Tracking Error Reduction in CNC Machining by Reshaping the Kinematic Trajectory *

Jian-Xin Guo, Qiang Zhang, Xiao-Shan Gao
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, Beijing 100190, China

Abstract

In this paper, a method of reducing the tracking error in CNC machining is proposed. The structured neural network is used to approximate the discontinuous friction in CNC machining, which has jump points and uncertainties. With the estimated nonlinear friction function, the reshaped trajectory can be computed from the desired one by solving a second order ODE such that when the reshaped trajectory is fed into the CNC controller, the output is the desired trajectory and the tracking error is eliminated in certain sense. The proposed reshape method is also shown to be robust with respect to certain parameters of the dynamic system.

Keywords. CNC controller, tracking error, structured neural network, robustness.

1 Introduction

Traditionally, the CNC machine programming is divided into three major levels: path planning, trajectory planning, and trajectory control. The path planning phase aims to define the geometric cutting path without timing information [1]. The trajectory planning considers the kinematic information, such as velocity, acceleration, jerk, and jounce, to generate a velocity profile along the specified geometric path (Jamhour and Andr [2], Dong and Stori [3], and Smith et al. [4], Fan et al [5]). Time optimal trajectory generation process plays an important role in maximizing productivity in CNC machining. The trajectory control is in the closed-loop real-time process, which aims to respond swiftly to ensure the actuators moving along the specified trajectory accurately.

In the trajectory control phase, due to the inherent dynamics limits, it is impossible to respond instantaneously to variations in the commanded path. Moreover, high feedback gains cannot be used to eliminate all tracking errors, for that will prohibit compliant control and make the system less safe for the environment. As a result, the actual machine motion deviates from the desired motion inevitably and the contour error may emerge as well.

Hence, whatever algorithm is adopted for trajectory planning, tracking errors exist in each axis. A number of different approaches have been proposed to ensure high tracking accuracy. Renton [6] developed a method to reduce cycle time and path error, which uses the axis performance envelope as well as instantaneous position, velocity, and acceleration information of the target path to improve servo performance in the presence of disturbances. Dong [7] brought the dynamic information into the trajectory planning level by reducing the tracking error to a linear combination of velocity and acceleration approximately. Lo [8] presented a new servo control method for axis machining which conducts a direct elimination of the deviation error, the orientation error, and the tracking

* Partially supported by a National Key Basic Research Project of China and by a USA NSF grant CCR-0201253.

error. Dong [9] presented a synchronization approach to trajectory tracking of multiple mobile robots. The main idea is to control each robot tracking its desired trajectory while synchronizing its motion with other robots to keep the relative kinematics relationship. Farouki [10] solved the problem of compensating for inertia and damping of the machine axes by a priori reshaping the commanded path geometry for CNC machines governed by typical feedback controllers.

In this paper, the idea of reshaping the original cutter path with kinematic information is used to reduce tracking errors. This idea was first proposed in the robot community [11] and recently used in CNC controlling [10]. The principle of these work is the same which aims to modify the input signal to the CNC controller so that when the modified signal is fed into the controller, the output is exactly the desired cutter path and thus there exists no tracking error comparing to the original cutter path. The contributions of this paper are as follows. A more general and practical model with the PID controller and nonlinear uncertainty frictions is considered, while previous work only considers the PD controller and simpler friction models. The nonlinear friction is learned through an SNN (Structured Neural Network), which can approximate discontinuous functions with jump points nicely. Each continuous piece of the friction is approximated by the powerful BPNN (Back Propagation Neural Network) model. Compared with the method of approximating the friction with a given explicit expression, the SNN/BPNN model can cope with a broad range of unknown disturbances regardless of their specific structure. After the friction is estimated with SNN, the reshaped trajectory can be computed from the original one by solving a second ODE. We also show that the proposed method is robust with respect to certain parameters of the dynamic system. Finally, a practical case from CNC machining is used to verify the approach by Matlab/Simulink.

The paper is organized as follows. Section 2 describes the basic CNC controller and dynamic models. Section 3 introduces the SNN model used to estimate the friction. Section 4 presents the reshaping method and the robustness analysis of the method. Section 5 presents the simulation results. Section 6 concludes the paper.

2 Machine dynamics

In this section, we will introduce the machine dynamics, the controller, and the friction model used in this paper. Suppose that the two axes are decoupled and the dynamic parameters for each axis are the same for brevity. Here, we mainly discuss the dynamics for the x-axis and similar principles apply to the y-axis.

2.1 Actuator model and PID control

The control system is composed of two main parts: the actuator an the controller. The actuator is assumed to be of the universal form motor which is shown in Fig.1 as the part from the actuating signal u to the axis location x. The actuator works as follows. The current amplifier k_a converts the actuating signal u into current i to control the motor, which produces a torque T through the motor torque gain k_t. The torque T determines the angular speed through the system inertia J and damping B. The motor shaft angle speed θ obtained by integration of ω, determines the axis linear position x through the transmission ratio r_g. The relation between ω and x is

$$\omega(t) = \frac{\dot{x}(t)}{r_g} \tag{1}$$

where t is the time and $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$. For brevity, set $K = k_a k_t r_g$ since these three parameters often occur in the form of this product.

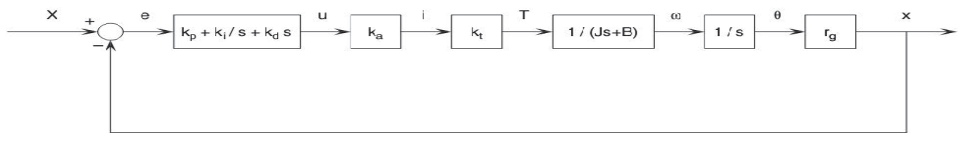

Figure 1: Closed loop control system with PID controller

The PID controller is widely used in CNC systems, whose parameters are the proportional, integral, and derivative gains k_p, k_d, and k_i as shown in Fig.1. The tracking error $e = X - x$ is the difference between the commanded and actual axis locations, which satisfies the following differential equation in the time domain.

$$k_d \ddot{e} + k_p \dot{e} + k_i e = \dot{u} \tag{2}$$

2.2 Overall system with friction model

In practice, the controller incurs the outside disturbance T_f. Thus the overall system can be more precisely described as the one given in Fig. 2. Denote T_n to be the nominal control torque worn by the external disturbance T_f and T_d to be the desired output control voltage generated from the controller.

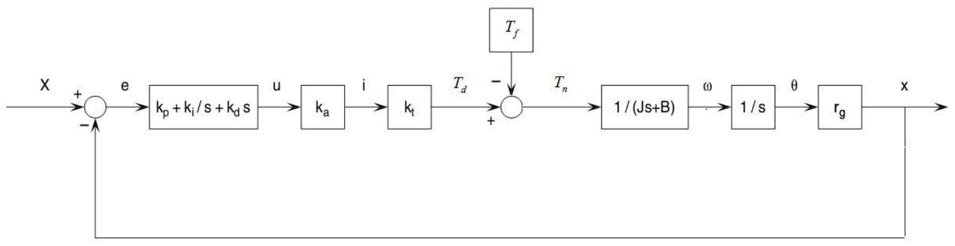

Figure 2: Closed loop control system with PID controller and outside disturbance

$$\ddot{x} = -\frac{B\dot{x}}{J} + \frac{r_g T_n}{J} \tag{3}$$

$$T_d = T_n + T_f \tag{4}$$

$$T_d = k_a k_t u \tag{5}$$

3 Friction estimation via SNN

In this section, we will introduce the principle of SNN which can be used to provide robust estimation for nonlinear functions with jump points. in the following reshaping procedure.

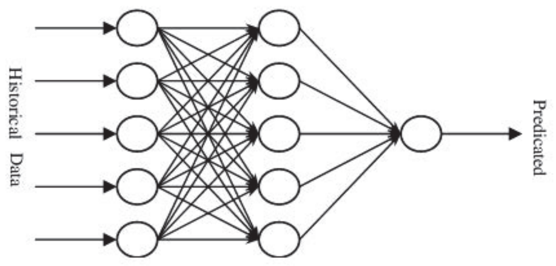

Figure 3: BPNN model

3.1 Principle of BPNN

The BPNN (Back Propagation Neural Network) is widely employed to estimate smooth nonlinear functions in an uncertain system [12].

Let us assume that the function to be estimated is the following real valued function

$$F : \mathbf{x} \to y,$$

where vector $\mathbf{x} \in \mathbb{R}^I$ and $y \in \mathbb{R}$. Supposing a set of sampling points $\mathbf{x_k} = (x_{k1}, x_{k2}, \ldots, x_{kI}), k = 1, \ldots, N$ and the corresponding values $y_k, k = 1, \ldots, N$ are given, the task is to construct a BPNN to approximate the function F. All the subscript in this section denoted with k stands for the sampled data.

The BPNN consists of three layers: the input layer, the hidden layer, and the output layer. For the input $\mathbf{x_k}$, the j-th hidden layer neuron $h_j (j = 1, 2, ..., H)$ can be calculated by

$$h_{kj} = f(\sum_{i=1}^{I} w_{ij}^1 \cdot x_{ki} + \theta_j^1)$$

where $f(x)$ denotes following sigmoid function:

$$f(x) = \frac{1}{1 + \exp(-x)},$$

w_{ij}^1 is the weight between the i-th input and h_j, and the θ_j^1 is the bias value in the j-th hidden layer. The output value $\hat{y}_k$ can be obtained as

$$\hat{y}_k = f(\sum_{j=1}^{H} w_j^2 \cdot h_{kj} + \theta^2) \tag{6}$$

where w_j^2 and θ^2 are the corresponding weights and bias.

To construct a BPNN, we need to “train” the parameters $w_{ji}^1, \theta_j^1, w_j^2, \theta^2$ through the following learning process. We start with a set of initial values for these parameters. The learning procedure works as follows: compute a set of values $\hat{y}_k, k = 1, \ldots, N$ using (6), update the parameters as follows:

$$\begin{aligned}
w_j^2 &= w_j^2 + \eta \cdot (\hat{y}_k - y_k) \cdot \hat{y}_k \cdot (1 - \hat{y}_k), \\
w_{ji}^1 &= w_{ji}^1 + \eta \cdot (\hat{y}_k - y_k) \cdot w_j^2 \cdot h_j \cdot (1 - h_j) \cdot x_{ki}, \\
\theta^2 &= \theta^2 + \eta \cdot (\hat{y}_k - y_k) \cdot \hat{y}_k \cdot (1 - \hat{y}_k), \\
\theta_j^1 &= \theta_j^1 + \eta \cdot (\hat{y}_k - y_k) \cdot w_j^2 \cdot h_j \cdot (1 - h_j),
\end{aligned}$$

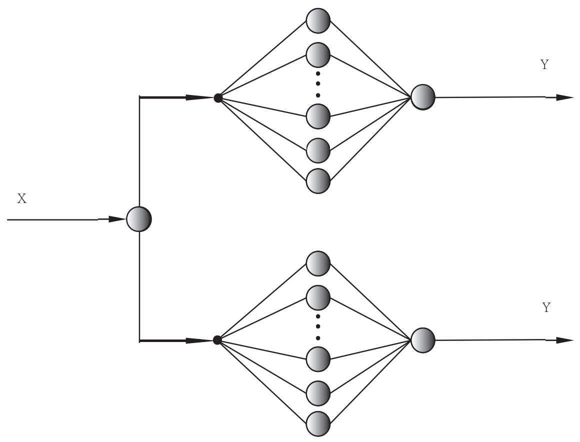

Figure 4: SNN model

where η is the learning parameter chosen between 0.5-0.7. The updated parameters are used to repeat the learning procedure. The procedure is repeated for 1000 times unless the optimal index

$$J = \frac{1}{2}\sum_{k=1}^{N}(\hat{y}_k - y_k)^2$$

is less than the given accuracy.

3.2 Principle of SNN

Although a three-layered BPNN can approximate any nonlinear smooth function adequately according to the artificial intelligence theory, it cannot be used to estimate functions with jump points nicely as shown in Fig.5, where the dotted line is the original function and he solid line is the estimation function. It is known that disturbances such as the Strikbek friction (see Section 5.1) contains jump points. So, we will use the so-called SNN (Structured Neural Network) to give a better estimation for functions with jump points as shown by Figure 6 [13].

Suppose that a function F has a finite number of jump points at $c_1, c_2, \ldots, c_N$, that is, it is piecewise continuous in the intervals $I_i = [c_{i-1}, c_i], i = 2, \ldots, N$. Then it can be estimated using the following SNN:

$$F(x) = \sum_{i=1}^{N+1} I_i(x) \cdot \tilde{F}_i(x),$$

where $I_i(x)$ is the i interval set function defined as

$$I_i(x) = \begin{cases} 1 & x \in I_i \\ 0 & x \notin I_i \end{cases}$$

and $\tilde{F}_i(x)$ is the BPNN estimation model for $F(x)$ in the interval I_i.

4 The reshaping method

A plane curve $(x(u), y(u)), u \in [0, 1]$ is used here to illustrate the procedure. After bounds for the chord error, feedrate, and axis accelerations are given, a sequence of interpolation points $(x_i, y_i), i = 1, \ldots, N$ can be computed efficiently with the method given in [17], which will be feeded into the

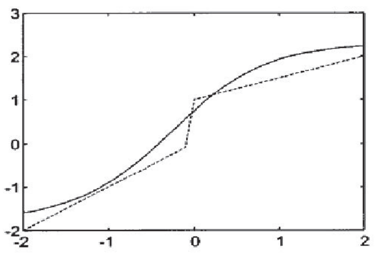
Figure 5: BPNN estimation

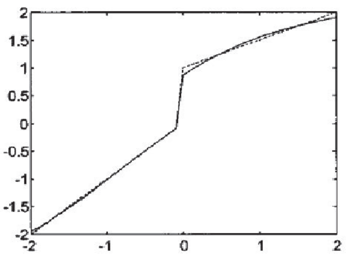
Figure 6: SNN estimation

CNC controller. Due to the existence of tracking errors, the output will deviate from the desired trajectory.

In this section, we will give a method to compute a sequence of reshaped interpolation points $(X_i, Y_i), i = 1, \ldots, N$ such that when these reshaped interpolation points are feeded to the CNC controller, the output is exactly the desired trajectory $(x_i, y_i), i = 1, \ldots, N$. This process is composed of three main steps to be given below.

4.1 Friction estimation via SNN

In this section, we show how to construct an SNN to estimate the unknown disturbance T_f from the angle velocity ω (See Fig. 2). In some complicated situation, the disturbance depends upon not only the velocity but also the other kinematic quantities. In this paper, it is assumed that the disturbance bears relationship only with velocity. More complicated situations can be treated similarly.

Take the x-axis as an example. In order to use SNN to approximate the function $T_f = f(\omega)$, we first need to collect the sample data used for calculating the input and target values in the SNN. From (3), (4), and (5), we have

$$T_f = T_d - T_n = k_a k_t u - J\ddot{x}/r_g - B\dot{x}/r_g. \tag{7}$$

Also note $\omega(t) = \frac{\dot{x}(t)}{r_g}$. As a consequence, if the control voltage u and the output x are known, it is able to calculate the data pair $(\omega_k, T_{fk}), k = 1, \ldots, N$.

In practice, values for u and x can be collected through the sensors mounted upon the CNC machine. In the simulation to be given in Section 5, we use the oscilloscope in the Simulink of Matlab to collect values for u. In order to guarantee the prediction accuracy of the SNN, we need to ensure the range of the collected data to be as large as possible. A practical way is to feed the interpolation points $(x_k, y_k), k = 1, \ldots, N$ into the CNC controller and use the corresponding values for u as the sample data set.

Let T be the interpolation period. Since the interpolation points $(x_k, y_k), k = 1, \ldots, N$ are dense enough, $\dot{x}_k = \dot{x}(kT)$ and $\ddot{x}_k = \ddot{x}(kT)$, $k = 1, \ldots, N$ can be approximated with finite differences for $k = 2, \ldots, N-1$:

$$\dot{x}_k \approx \frac{x_{k+1} - x_{k-1}}{2T} = \frac{x(k(T+1)) - x(k(T-1))}{2T} \tag{8}$$

$$\ddot{x}_k \approx \frac{x_{k+1} + x_{k-1} - 2x_k}{T^2} = \frac{x(k(T+1)) + x(k(T-1)) - 2x(k(T))}{T^2} \tag{9}$$

As usual, we assume that $\dot{x}_1 = \ddot{x}_N = 0$. When feeding the interpolation points $(x_k, y_k), k = 1, \ldots, N$ to the Simulink of Matlab, with the oscilloscope, we can collect the corresponding values for the voltage $u_k, k = 1, \ldots, N$. Using (7), the correspondence $(w_k, T_{fk}), k = 1, \ldots, N$ can be computed as follows

$$\begin{aligned} T_{fk} &= k_a k_t u_k - J\ddot{x}_k/r_g - B\dot{x}_k/r_g \\ \omega_k &= \dot{x}_k/r_g \end{aligned} \tag{10}$$

for $k = 1, \ldots, N$.

Now the sampled data $(w_k, T_{fk}), k = 1, \ldots, N$ can be used to construct an SNN to represent the function $T_f = f(\omega)$. We make the assumption that the unknown disturbance T_f has a unique jump point at $\omega = 0$. Therefore, a two parts SNN is used to approximate T_f, and the two nonlinear parts of the SNN are smooth and can be estimated using BPNN separately.

To ensure the accuracy of the prediction, the data pair (ω_k, T_{fk}) is normalized. The main objective here is to ensure that the statistical distribution of the values for the network input and output is roughly uniform. The sampled data are scaled so that they have zero mean and unitary variance. Precisely, let

$$\bar{\omega}_k = \frac{\omega_k - \omega_{\min}}{\omega_{\max} - \omega_{\min}}, \quad \bar{T}_{fk} = \frac{T_{fk} - T_{f_{\min}}}{T_{f_{\max}} - T_{f_{\min}}}, k = 1, \ldots, N \tag{11}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of ω_k, $T_{f_{\max}}$ and $T_{f_{\min}}$ are the maximum and minimum values of T_{fk}. The data pairs $(\bar{\omega}_k, \bar{T}_{fk}), k = 1, \ldots, N$ will be used to construct the SNN, whose range are $(0, 1)$.

Finally, let the SNN thus constructed be $\widehat{T_f}$ which maps an ω to the disturbance T_f.

4.2 Desired torque calculation

Take the x-axis as an example. In Figure 2, the nominal control torque T_n satisfies (3). Since the interpolation points $(x_i, y_i), i = 1, \ldots, N$ are dense enough, $\dot{x}$ and $\ddot{x}$ can be approximately computed through (8) and (9). From equation (3), $T_{n_i} = T_n(iT)$ can be computed through (12).

$$T_{n_k} = T_n(kT) = \frac{J\ddot{x}_k + B\dot{x}_k}{r_g}. \tag{12}$$

Hence the desired torque can be computed as $T_{di} = T_{ni} + \widehat{T}_{f_i}, i = 1, \ldots, N$ by (4), where $\widehat{T}_{f_i}$ is the torque obtained via the SNN method given in Section 4.1 with input $\omega_i = \frac{\dot{x}_i}{r_g}$.

4.3 Compute the reshaped trajectory

Substituting equation (5) into (2), we have

$$k_d\ddot{e} + k_p\dot{e} + k_i e = \frac{\dot{T}_d}{k_a k_t}. \tag{13}$$

In order to get the new tracking error, we just need to solve the ODE (13) with the initial values $e(0) = \dot{e}(0) = 0$. The fixed step fourth order Runge-Kutta method can be used to solve the second order ODE (13) to obtain the tracking error $e_i = e(iT), i = 1, \ldots, N$. Nevertheless, function values $\dot{T}_d(t)$ at places other than $iT, i = 1, \ldots, N$ are needed in this procedure. In order to provide these values, we interpolate T_d using cubic splines which will be used to calculate its derivative at any points.

Since T_d is piecewise continuous, we need to solve (13) piecewisely. Take the first interval as an example. Suppose the interval is divided into $k-1$ subintervals by the knots $t_1, t_2, \dots, t_k$. The following cubic polynomial splines are used to approximate T_d

$$S(t) = \begin{cases} S_1(t), t \in [t_1, t_2] \\ S_2(t), t \in [t_2, t_3] \\ \cdots \\ S_{n-1}(t), t \in [t_{k-1}, t_k], \end{cases}$$

where $S(t)$ is defined as

$$S_i(t) = \frac{z_{i+1}(t-t_i)^3 + z_i(t_{i+1}-t)^3}{6h_i} + (\frac{T_{d_{i+1}}}{h_i} - \frac{h_i}{6} z_{i+1})(t - t_i) + (\frac{T_{d_i}}{h_i} - \frac{h_i}{6} z_i)(t_{i+1} - t) \tag{14}$$

$t_i = iT$ and $h_i = t_{i+1} - t_i$. The value z_i can be obtained by solving the following linear equations:

$$\begin{cases} z_1 = 0 \\ h_{i-1} z_{i-1} + 2(h_{i-1} + h_i) z_i + h_i z_{i+1} = 6(\frac{T_{d_{i+1}} - T_{d_i}}{h_i} - \frac{T_{d_i} - T_{d_{i-1}}}{h_{i-1}}) \\ z_k = 0. \end{cases}$$

With the newly constructed function $T_d(t)$, the fixed step fourth order Runge-Kutta method will be used to solve the second order ODE to obtain $e_i, i = 1, \dots, k$. For the second interval, the initial values of e and $\dot{e}$ are chosen to be their end values in the first interval. This is rational since e and $\dot{e}$ are continuous even if the system may be subjected to some discontinuous or unknown disturbance.

Repeat the procedure, all $e_i, i = 1, \dots, N$ will be computed. Finally, the reshaped inputs are

$$X_i = x_i + e_i, i = 1, \dots, N.$$

The same procedure can be implemented to the y axis as well. Thus the reshaped signals $(X_i, Y_i), i = 1, \dots, N$ are acquired.

4.4 Robustness analysis

In this section, we will briefly discuss the robustness of the reshaping method, that is, when the mechanism parameters are not accurate but very close to the exact ones, the reshaped curve obtained with these parameters are also very close to the exact reshaped curve.

To begin with, the following system equation is crucial for us to analyze the robustness, which is deduced from (2), (3), (4), and (5)

$$\frac{J}{K}\dddot{x} + (\frac{B + Kk_d}{K})\ddot{x} + k_p\dot{x} + k_i x + \frac{1}{k_a k_t}\frac{dT_f}{dt} = k_d\ddot{X} + k_p\dot{X} + k_i X. \tag{15}$$

To avoid analyzing the derivative $\frac{dT_f}{dt}$, we make the substitutions $y = \int_0^t x(\tau)d\tau$, $Y = \int_0^t X(\tau)d\tau$. Since the initial values of x, $\dot{x}$, $\ddot{x}$ and $\dddot{x}$, X, $\dot{X}$, $\ddot{X}$ and $\dddot{X}$ are zero, equation (15) becomes:

$$\frac{J}{K}\dddot{y} + (\frac{B + Kk_d}{K})\ddot{y} + k_p\dot{y} + k_i y + \frac{T_f}{k_a k_t} = k_d\ddot{Y} + k_p\dot{Y} + k_i Y. \tag{16}$$

In the reshaping process, the following equation will be used

$$\frac{J}{K}\dddot{y} + (\frac{B + Kk_d}{K})\ddot{y} + k_p\dot{y} + k_i y + \frac{\widehat{T}_f}{k_a k_t} = k_d\ddot{Y} + k_p\dot{Y} + k_i Y. \tag{17}$$

where $\widehat{T_f}$ is the disturbance estimation learned through SNN. In this phase, the left hand side including y (desired output) is known and Y on the right hand side is unknown. As discussed above, in practice, the parameters are not accurate. For instance, suppose J and B are estimated wrongly as $\tilde{J}$ and $\tilde{B}$. Thus we actually use the follow equation to compute the reshaping coordinate Y:

$$\frac{\tilde{J}}{K}\dddot{y} + (\frac{\tilde{B}+Kk_d}{K})\ddot{y} + k_p\dot{y} + k_i y + \frac{\widetilde{T_f}}{k_a k_t} = k_d\ddot{Y} + k_p\dot{Y} + k_i Y. \tag{18}$$

where $\widetilde{T_f}$ is the disturbance estimated through SNN using parameters $\tilde{J}$ and $\tilde{B}$.

Hence we will obtain the inaccurate reshaping value $\tilde{Y}$ with (18), while the accurate reshaped value Y is obtained from the reverse equation (17).

When implementing the reshaping process, only the right hand side of (17) contains variables. Thus defining the left side of equation (17) to be $\theta(t)$, (17) becomes:

$$k_d\ddot{Y} + k_p\dot{Y} + k_i Y = \theta(t), \tag{19}$$

where

$$\theta(t) = \frac{J}{K}\dddot{y} + (\frac{B+Kk_d}{K})\ddot{y} + k_p\dot{y} + k_i y + \frac{\widehat{T_f}}{k_a k_t}.$$

Besides, denoting the left hand side of equation (18) to be $\tilde{\theta}(t)$ correspondingly. In order to analyze the relationship between $Y, \dot{Y}, \ddot{Y}$ and $\tilde{Y}, \dot{\tilde{Y}}, \ddot{\tilde{Y}}$, we transform equation (19) into the following matrix form by defining $\xi = (Y, \dot{Y})^T$:

$$\dot{\boldsymbol{\xi}} = A\boldsymbol{\xi} + \boldsymbol{\eta}(t), \boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0 \tag{20}$$

where

$$A = \begin{pmatrix} 0 & 1 \\ -\frac{k_i}{k_d} & -\frac{k_p}{k_d} \end{pmatrix}, \quad \boldsymbol{\eta}(t) = \begin{pmatrix} 0 \\ \theta(t) \end{pmatrix}.$$

Although the right hand side of this ODE could be discontinuous, we can discuss its solution piecewisely. Suppose there are k intervals $I_1, \ldots, I_k$ and in each I_k $\theta(t)$ is continuous. We now consider ODE (20) over the first interval I_1. Suppose that the solution $\tilde{\boldsymbol{\xi}}$ of (20) with inaccurate parameters satisfies

$$\dot{\tilde{\boldsymbol{\xi}}} = A\tilde{\boldsymbol{\xi}} + \tilde{\boldsymbol{\eta}}(t), \tilde{\boldsymbol{\xi}}(t_0) = \tilde{\boldsymbol{\xi}}_{\mathbf{0}}, \tag{21}$$

where $\|\boldsymbol{\eta}(t) - \tilde{\boldsymbol{\eta}}(t)\|_1 \leq \varepsilon$ and $\|\cdot\|_1$ denote the 1-norm which is the sum of all the absolute values of the elements in the matrix.

The solutions of equations (20) and (21) can be written respectively as

$$\begin{aligned} \boldsymbol{\xi}(t) &= \boldsymbol{\xi}_0 + \textstyle\int_0^t \boldsymbol{A}\boldsymbol{\xi}(\tau) + \boldsymbol{\eta}(\tau)d\tau \\ \tilde{\boldsymbol{\xi}}(t) &= \tilde{\boldsymbol{\xi}}_0 + \textstyle\int_0^t \boldsymbol{A}\tilde{\boldsymbol{\xi}}(\tau) + \tilde{\boldsymbol{\eta}}(\tau)d\tau. \end{aligned}$$

We thus have

$$\begin{aligned} &\left\|\boldsymbol{\xi}(t) - \tilde{\boldsymbol{\xi}}(t)\right\|_1 \leq \left\|\boldsymbol{\xi}_0 - \tilde{\boldsymbol{\xi}}_0\right\|_1 + \textstyle\int_0^t \left\|A\boldsymbol{\xi}(\tau) - A\tilde{\boldsymbol{\xi}}(\tau)\right\|_1 d\tau + \textstyle\int_0^t \|\boldsymbol{\eta}(\tau) - \tilde{\boldsymbol{\eta}}(\tau)\|_1 d\tau \\ &\leq \varepsilon_0 + \|A\|_1 \textstyle\int_0^t \left\|\boldsymbol{\xi}(\tau) - \tilde{\boldsymbol{\xi}}(\tau)\right\|_1 d\tau + \textstyle\int_0^{T_1} \|\boldsymbol{\eta}(\tau) - \tilde{\boldsymbol{\eta}}(\tau)\|_1 d\tau \\ &\leq \varepsilon_0 + \|A\|_1 \textstyle\int_0^t \left\|\boldsymbol{\xi}(\tau) - \tilde{\boldsymbol{\xi}}(\tau)\right\|_1 d\tau + \varepsilon_1 \\ &= \varepsilon + \|A\|_1 \textstyle\int_0^t \left\|\boldsymbol{\xi}(\tau) - \tilde{\boldsymbol{\xi}}(\tau)\right\|_1 d\tau, \end{aligned}$$

where $\varepsilon = \varepsilon_0 + \varepsilon_1$ and T_1 is the length of the first interval I_1.

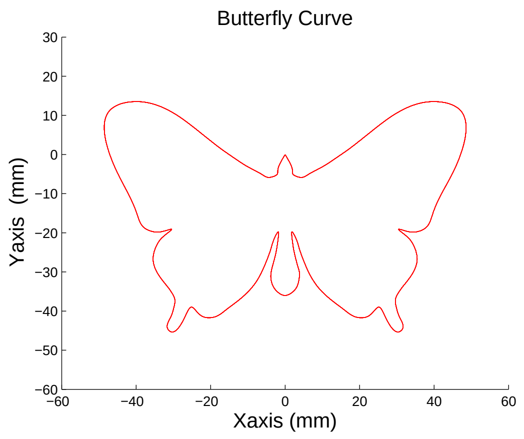

Figure 7: Butterfly Curve

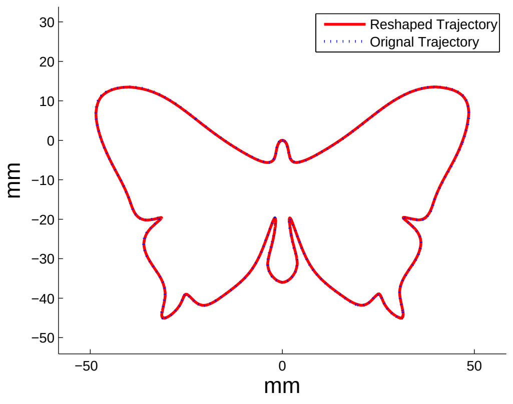

Figure 8: Comparison in Test 1

According to Grownwall inequality, we have:

$$\left\|\boldsymbol{\xi}(t) - \tilde{\boldsymbol{\xi}}(t)\right\|_1 \leq \varepsilon e^{\|\boldsymbol{A}\|_1 t} \leq \varepsilon e^{\|\boldsymbol{A}\|_1 T_1}.$$

Thus as long as the change of the initial values $\boldsymbol{\xi}_0$ and $\boldsymbol{\eta}$ are small, the new solution cannot deviate from the original solution largely in the whole interval I_1. The rest of intervals $I_2, \ldots, I_k$ can be analyzed similarly.

Therefore, in order to show the robustness of the reshaping procedure, we need only to show that small changes in J and B will lead to small changes of $\boldsymbol{\eta}$, or equivalently small changes of $\theta(t)$. From (19), $\theta(t)$ depends on B and J linearly except the part $\widehat{T_f}$. So, we need to show that small changes of J and B will lead to small changes of $\widehat{T_f}$. For that purpose, we denote $\widehat{T_f}$ as $\widehat{T_f}(J, B)$.

Recall that $\widehat{T_f}(J, B)$ is learned from the map

$$F : \omega \to T_f,$$

and T_f is calculated through the observation u and x by

$$T_f = T_d - T_n = k_a k_t u - \frac{J\ddot{x} + B\dot{x}}{r_g}.$$

Obviously, T_f depends upon J and B continuously. Thus when small changes ΔJ and ΔB to J, B occur, the new map is

$$\tilde{F} : \omega \to T_f + \varepsilon, \tag{22}$$

where $\varepsilon = \frac{\Delta J\ddot{x} + \Delta B\dot{x}}{r_g}$. Hence, $\widehat{T_f}$ and $\widetilde{T_f}$ are sufficiently close when small changes occur to J and B.

Based on the above analysis, we can conclude that the reshaped curve obtained with inaccurate parameters is close to the accurate reshaped curve if the inaccurate parameters are close to the exact ones.

5 Simulation results

5.1 System setup

A plane curve from [14] shown in Fig.7 is used to perform the simulation. The bounds of the velocity and acceleration are set to be $V_m = 150mm/s$ and $A_{x\max} = A_{y\max} = 3000mm/s^2$. The

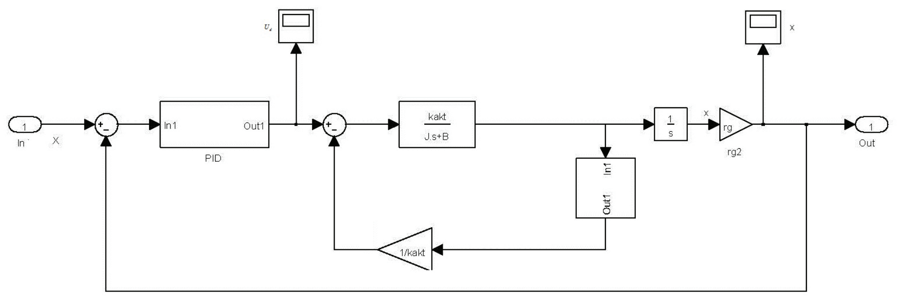

Figure 9: System mounted under the environment in Simulink

sampling period is $1ms$. The methods proposed in [17] is used to compute the velocity and the method given in [5] is used to compute the interpolation points $(x_k, y_k), k = 1, \dots, N$ where N is set to be 100.

In the first step of the method, we need to train the SNN off-line to obtain a disturbance model T_f following the method given in Section 4.1. Again, the x-axis is used to illustrate the procedure.

Since x_k are known, from equation (10), what we need to know are $u_k, k = 1, \dots, N$, which will be obtained via Matlab/Simulink. We observe the control voltage u via sensors in Matlab/Simulink, which is shown in Fig. 9. The system parameters are given in Table 1.

Table 1: Parameters used in the simulation

Meaning	Parameter	Unit	Value
current amplifier	k_a	A/V	5
Armature resistance	k_t	Nm/A	2
Transmission ratio	r_g	rad/m	0.005
Transmission gain	K	Nm^2/V	0.05
Minimum level of coulomb friction	f_c	Nm	500
Static friction	f_s	Nm	100
Viscous friction	f_v	Nm/rad	0.05

Aiming to approximate the real environment as near as possible in the Matlab/Simulink simulation, the following Stribeck friction [15] shown in Fig. 10 is used as the disturbance force

$$T_f = (f_c + f_s e^{-(\frac{\omega}{\omega_s})^2} + f_v\,|\omega|)\mathrm{sign}(\omega), \tag{23}$$

where f_c is the minimum level of coulomb friction, f_s is the level of static friction parameter, f_v is the viscous friction parameter, and ω is the motor angle velocity. The values of these parameters are given in Table 1.

Now, $u_k, k = 1, \dots, N$ can be obtained by feeding $x_k, k = 1, \dots, N$ into the Matlab/Simulink with the Stribeck friction (23) as the disturbance force, and the disturbance model $\widehat{T_f}$ can be obtained with the method given in Section 4.1.

Following the method given in Sections 4.2 and 4.3, the reshaped x-coordinates $X_k, k = 1, \dots, N$ can be computed.

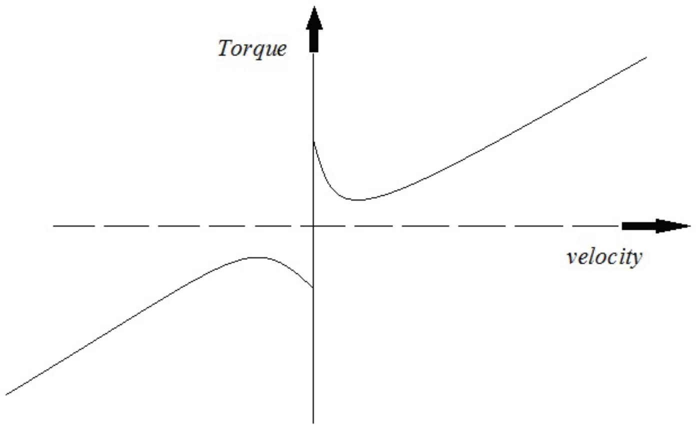

Figure 10: Stribeck effect, static, Coulomb, and viscous friction model

5.2 Simulation results

Three tests will be given in this section to illustrate the effectiveness of the method. In the first two tests, different values of rotary inertia are used to compare the effect on the reshaped tool path, where parameters of PID are the same. The third test is used to show the robustness of the method.

Table 2: Parameters used in the test

Parameters	Test 1	Test 2
J	$0.003kgm^2$	$0.01kgm^2$
B	$0.006kgm^2/s$	$0.02kgm^2/s$
K_p	$1000V/mm$	$1000V/mm$
K_i	$1000V/(mm \cdot s)$	$1000V/(mm \cdot s)$
K_d	$10V/(mm/s)$	$10V/(mm/s)$

Test 1: Smaller inertia J and damping B.

The parameters used in test 1 are given in Table 2. The original and reshaped curves are shown in Fig. 8. Due to the small magnitude of the changes, we cannot see the difference clearly in this figure. In Fig.11-Fig.14, four "sharp corners" of the reshaped tool path and the commanded path are given. We can see that in these four sharp corners, the modification is evident. In such corners, the controller can not respond to the input signal swiftly, so the lagging effect is conspicuous. The figures revel that the new input signal takes an advanced turn compared with the origin one in order to output the correct result. From Fig.15-Fig.16, we can see that with these system parameters, the tracking error (the red curve) is between -200um-250um for the x-axis and -300um-250um for the y-axis. For the reshaped tool path, the tracking error diminishes to near zero, which valid the goal of the method. The fluctuation of the tracking error near zero is mainly due to numerical calculation errors. Moreover, in Fig.17-Fig.18 we can see that the SNN approximates the nonlinear part of the system accurately, which will render accurate compensations for reshaping procedures.

Test 2: Larger inertia J and damping B.

The parameters used in test 2 are in Table 2 and the results are given in Fig.19-Fig.26. Compared with test 1, the differences between the original and shaped tool pathes are bigger. In Fig.21, the reshaped path contains a loop which is required to compensate the inability of the controller dynamics in tracking the sharp corner of the original tool path. Correspondingly, the tracking error of the unshaped tool path is bigger as shown in Fig.23 and Fig.24.

Test 3: Robustness.

Parameters in Test 1 are used to check the robustness of the method. Suppose that the mechanism parameters J and B are wrongly estimated. The accurate values are $J = 0.003, B = 0.006$ and the estimated values are $\tilde{J} = 0.00375, \tilde{B} = 0.0045$, where the estimation errors are within 25% of the accurate value. The rest parameters are assumed to be accurate.

Fig.27 and Fig.28 illustrate the tracking errors for original tool path and the shaped tool path with the contaminated J ad B. We can see that the tracking error for the reshaped tool path becomes larger, but is in an acceptable level. In particular, it is still much smaller than the original tracking error.

The force disturbance estimated through SNN is another crucial factor for the robustness. Fig.29 and Fig.30 are the figures for the function $T_f = f(w)$ obtained with the accurate parameters and the contaminated parameters. In Fig.30, a “tunnel” is generated, which is actually the fluctuating noise around the original curve.

This phenomenon can be comprehended through (22). The noise magnitude is actually the same as that of ε which depends upon the deviation of J and B. Fortunately, due to the linear relationship between the right hand side of the map (22) with J and B, this fluctuation can not be too large. Thus the learning accuracy can be guaranteed to certain degree. Thus, no matter which curve in Fig.29 or Fig.30 is learned through SNN, the learned curve will be included in the “tunnel”. Hence as long as the diameter of the tunnel is small the wrongly learned map curve by SNN is deviated from the real estimation in Fig.29 by no more than ε uniformly.

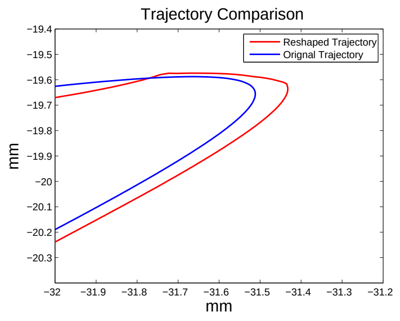

Figure 11: Left corner in Test 1

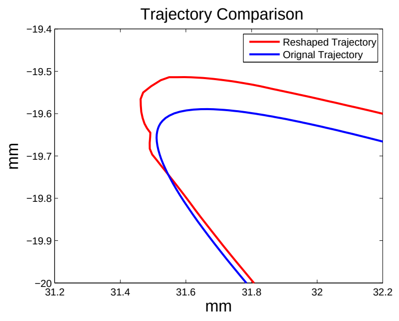

Figure 12: Right corner in Test 1

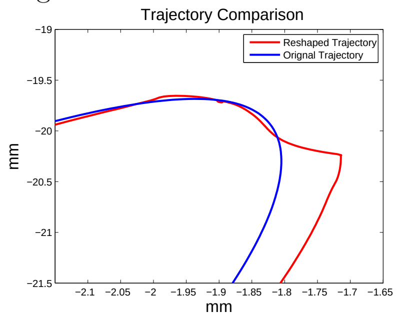

Figure 13: Middle left corner in Test 1

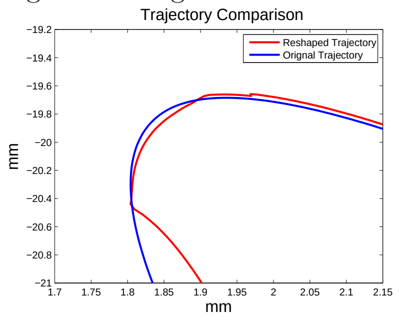

Figure 14: Middle right corner in Test 1

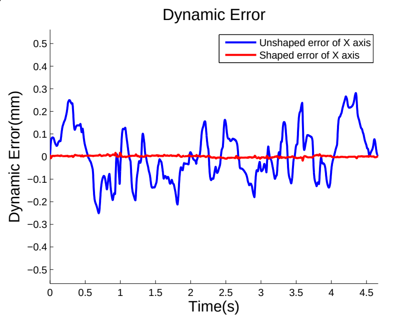

Figure 15: Tracking error of X axis in Test 1

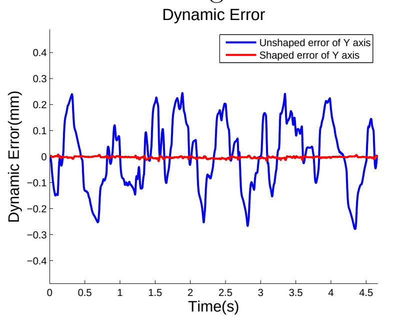

Figure 16: Tracking error of Y axis in Test 1

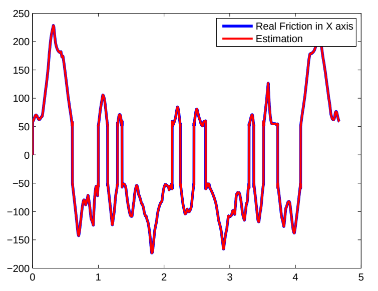

Figure 17: Friction torque of X axis in test 1

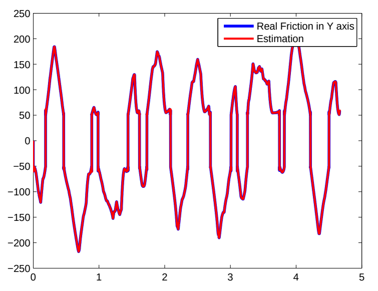

Figure 18: Friction torque of Y axis in test 1

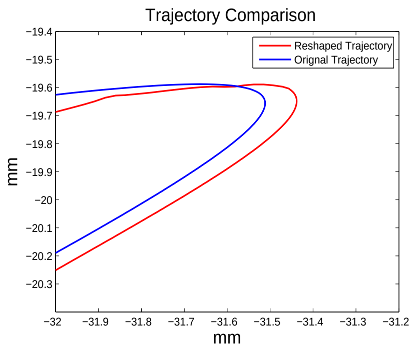

Figure 19: Left corner in Test 2

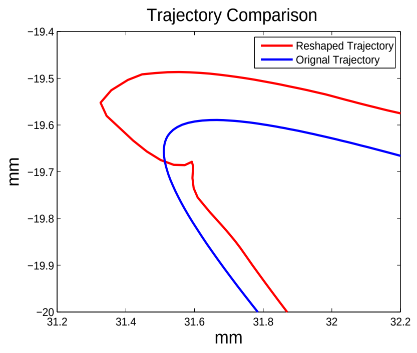

Figure 20: Right corner in Test 2

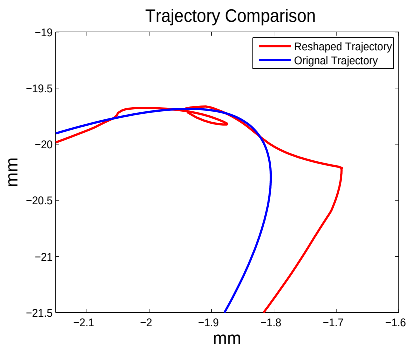

Figure 21: Middle left corner in Test 2

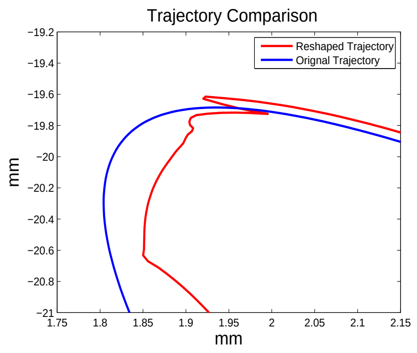

Figure 22: Middle right corner in Test 2

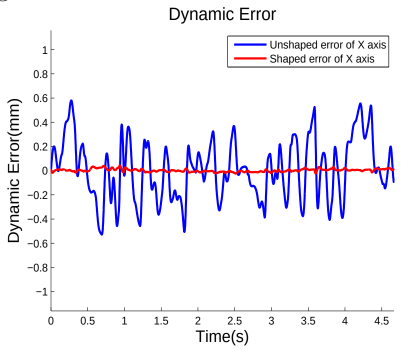

Figure 23: Tracking error of X axis in Test 2

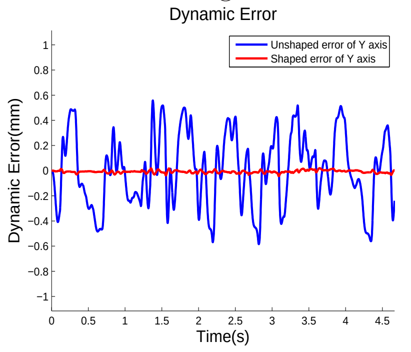

Figure 24: Tracking error of Y axis in Test 2

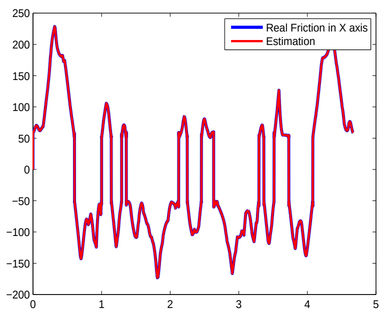

Figure 25: Friction torque of X axis in test 2

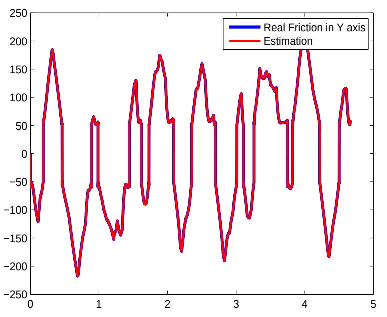

Figure 26: Friction torque of Y axis in test 2

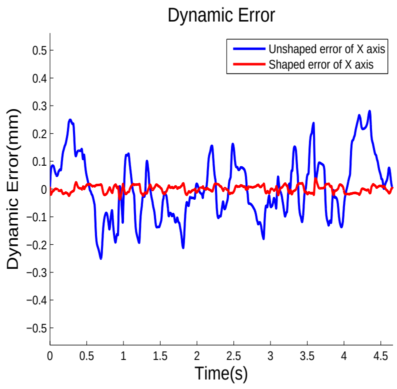

Figure 27: Tracking error of X axis in Test 3

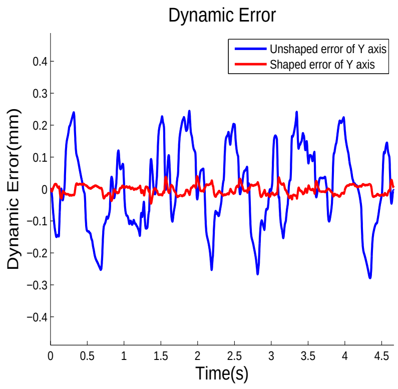

Figure 28: Tracking error of Y axis in Test 3

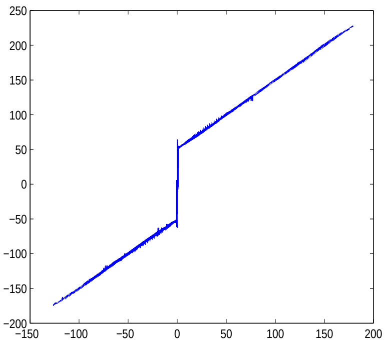

Figure 29: Uncontaminated figure for $T_f(\omega)$

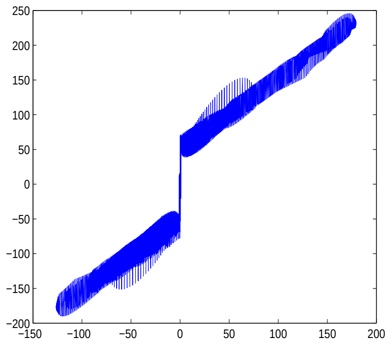

Figure 30: Contaminated for $T_f(\omega)$

6 Conclusion

A method is proposed to reduce the inherent tracking error of high-speed CNC machining. In order to be practical, the CNC dynamic system is assumed to have a discontinuous disturbance and the controller is assumed to be the widely used PID controller. The method mainly consists of three steps: the unknown disturbance estimation, the compensation of the desired control voltage, and the computation of the reshaped command signal. An advantage of this approach is that it is implemented outside the control loop, requiring no access to the control system.

The disturbance is learnt with an SNN model via data collected from CNC machine sensors. The SNN model is known to be a nice way to provide robust estimation for nonlinear functions with jump points. With the SNN based disturbance function, the reshaped signal is obtained by solving a linear differential equation. Moreover the procedure is shown to be robust with respect to certain parameters of the dynamic system. The method is illustrated with a practical example, where the tracking error is reduced to almost zero.

References

[1] Castelino K, D'Souza R, Wright PK. Tool path optimization for minimizing airtime during machining. *Journal of manufacturing systems*, 2003; 22(3): 173-180.

[2] Jamhour E, Andr PJ. Planning smooth trajectories along parametric paths. *Mathematics and computers in simulation* , 1996; 41(5-6): 615-626.

[3] Dong J, Stori JA. A generalized time-optimal bi-directional scan algorithm for constrained feedrate optimization. *ASME journal of dynamic systems, measurement, and control*, 2006; 128: 379-390.

[4] Smith TS, Farouki RT, Timar SD, Boyadjieff GL. Algorithms for time-optimal control of CNC machines along curved tool paths. *Robotics and computer integrated manufacturing* , 2005; 21: 37-53.

[5] Fan W, Gao XS, Yan W, Yuan CM. Interpolation of parametric CNC machining path under confined jounce. *Int J Adv Manuf Technol.* 62, 719C739, 2012

[6] Renton D, Elbestawi MA. High speed servo control of multi-axis machine tools. *International journal of machine tools and manufacture* , 2000; 40:539-559.

[7] Dong JY, Stori JA. Optimal feed-rate scheduling for high-speed contouring. *Journal of manufacturing science and engineering* , 2007; 129(1):63-76.

[8] Lo CC. A tool-path control scheme for five-axis machine tools. *International journal of machine tools and manufacture*, 2002;42 (1):37-53.

[9] Dong SY, Can W, Wen S, Gang F. A synchronization approach to trajectory tracking of multiple mobile robots while maintaining time-varying formations. *IEEE transactions on robotics*, 2009;5(5): 1074 - 1086.

[10] Ernesto CA, Farouki RT. Solution of inverse dynamics problems for contour error minimization in CNC machines. *International journal of advanced manufacturing technology* , 2010; 49:589-604.

[11] Shiller Z, Chang H. Trajectory preshaping for high-speed articulated systems. *Journal of dynamic systems, measurement, and control*, 1995; 117(304-310).

[12] Cheng F, Fan KC, Miao JW, Li BK, Wang HY. A BPNN-PID based long-stroke nanopositioning control scheme driven by ultrasonic motor., *Precision engineering*, 2012; 36(3):485C493.

[13] Rastko RS, Frank LL. Neural-Network approximation of piecewise continuous functions: application to friction compensation. *IEEE Trans on Neural Networks*, 2002; 13(3):745 - 751.

[14] Yau HT, Lin MT, Tsai MS. Real-time NURBS interpolation using FPGA for high speed motion control., *Computer-Aided Design*, 2006;38:1123-1133.

[15] Huang SN, Lee TH. Robust adaptive numerical compensation for friction and force ripple in permanent-magnet. *IEEE transactions on magnetic*, 2002;38(1):221-228.

[16] Lischinsky P, Canudas-de-Wit C, Morel G. Friction compensation for an industrial hydraulic robot. *Control systems, IEEE*, 1999;19(1):25-32.

[17] Zhang K, Yuan CM, Gao XS. Efficient algorithm for time-optimal feedrate planning and smoothing with confined chord error and acceleration. *Int. J. Adv. Manuf. Technol.*. DOI 10.1007/s00170-012-4450-3.