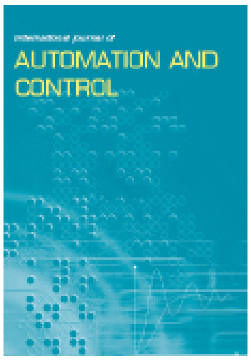


International Journal of Automation and Control

ISSN online: 1740-7524 - ISSN print: 1740-7516
https://www.inderscience.com/ijaac

Numerical integration-like algorithm for time-optimal trajectory optimisation of multi-axis motion system based on iterative learning

Tie Zhang, Cailei Liao, Yanbiao Zou, Zhongqiang Kang, Caicheng Wu

DOI: 10.1504/IJAAC.2023.10051796

Article History:

Received:	28 March 2021
Last revised:	14 August 2021
Accepted:	11 September 2021
Published online:	30 November 2022

Numerical integration-like algorithm for time-optimal trajectory optimisation of multi-axis motion system based on iterative learning

Tie Zhang*, Cailei Liao, Yanbiao Zou, Zhongqiang Kang and Caicheng Wu

School of Mechanical and Automotive Engineering,
South China University of Technology,
Guangzhou, China
Email: merobot@scut.edu.cn
Email: 1603321299@qq.com
Email: ybzou@scut.edu.cn
Email: kangzhongqiang@foxmail.com
Email: 971890417@qq.com
*Corresponding author

Abstract: In order to realise the high-velocity and high-precision motion of the multi-axis motion system, a numerical integration-like time-optimal trajectory optimisation algorithm combined with iterative learning is proposed. Based on the established dynamic model of the multi-axis motion system, the mathematical model with time optimisation as the objective function is derived under kinematics and dynamics constraints. The planned trajectory is discretised and the uniform acceleration equation (SUVAT) between any two adjacent discrete points is assumed so that pseudo-velocity planning of the phase plane is carried out by SUVAT equation instead of numerical integration method, after which the optimal solution satisfying the constraints can be obtained. In order to improve the dynamic model and reduce the errors between the calculated and the actual measured torques, a PD-type iterative learning method with forgetting factor is used to continuously update the dynamic model.

Keywords: high-velocity; high-precision; multi-axis motion system; time-optimal control; dynamic model; phase plane; numerical integration; iterative learning; forgetting factor.

Reference to this paper should be made as follows: Zhang, T., Liao, C., Zou, Y., Kang, Z. and Wu, C. (2023) 'Numerical integration-like algorithm for time-optimal trajectory optimisation of multi-axis motion system based on iterative learning', *Int. J. Automation and Control*, Vol. 17, No. 1, pp.91–115.

Biographical notes: Tie Zhang is a Professor in School of Mechanical Automotive Engineering at South China University of Technology. He specialises in the research of robots and automation equipment, especially kinematics and dynamics control.

Cailei Liao received his Master's in School of Mechanical Automotive Engineering at South China University of Technology. He studies trajectory optimisation based on dynamic model.

Yanbiao Zou is a Professor in School of Mechanical Automotive Engineering at South China University of Technology. He specialises in the research of robots, especially image recognition, and deep learning.

Zhongqiang Kang received his Master's degree in School of Mechanical Automotive Engineering at South China University of Technology. He studies trajectory optimisation based on vibration suppression.

Caicheng Wu received his Master's degree in School of Mechanical Automotive Engineering at South China University of Technology. He studies contour error control.

1 Introduction

After years of development, modern CNC systems such as turning and milling integrated machining centres have multi-axis simultaneous machining functions, which can achieve complex curve surface machining. However, there are always a lot of empty strokes in the processing of complex objects, which makes the control of processing velocity an urgent problem to be solved because faster processing velocity can improve the processing efficiency and shorten the production cycle of products. In terms of this problem, a large number of time-optimal trajectory planning methods have been studied and significant research progress has been made. Generally speaking, time-optimal trajectory planning can be divided into three stages: motion planning (Ziadi et al., 2021; Mohamed et al., 2018; Ouda et al., 2018), trajectory optimisation and trajectory tracking (Bellahcene et al., 2021; Guevara et al., 2019; Capito et al., 2016). This paper mainly focuses on trajectory optimisation, that is, each axis is required to work at the maximum allowable velocity under the critical constraints of a multi-axis motion system to achieve high velocity while ensuring high accuracy.

Time-optimal trajectory optimisation is a single-objective nonlinear optimisation problem with multiple constraints, which can be solved by two methods. Specifically, it has been proved to be effective in solving this problem through complex mathematical optimisation methods, such as the dynamic programming method (Kaserer et al., 2018), the sequential quadratic programming (SDP) method (Debrouwere et al., 2013). Although the mathematical optimisation method can be applied to nonlinear models, it requires too much calculation. Therefore, numerical integration method was proposed by Bobrow et al. (1985), which obtains the optimal velocity control curve satisfying the torque constraint by finding the conversion curve in the phase plane. On this basis, Constantinescu and Croft (2000) increased the constraint of the torque change rate to obtain a smoother planning curve. After that, Slotine and Yang (2002) proposed a new numerical integration method to make the planning time shorter. Mattmuller and Gisler (2009) proposed a pseudo-velocity optimisation method based on the phase plane iteration method. Although the numerical integration method has less calculation, it often needs to linearise the dynamic model (Reynosomora et al., 2013) to simplify the integration calculation, such as ignoring viscous friction, which leads to the inaccuracy of the dynamic model.

Therefore, a numerical integration-like method is proposed to improve the accuracy of the dynamic model by considering the nonlinear term of the dynamic model. The method not only retains the advantage of the small amount of calculation, but also can calculate the nonlinear model, which is realised by replacing the numerical integration method with the uniform acceleration equation (SUVAT) for pseudo-velocity planning on the phase plane.

In addition, although the dynamic model accuracy of the proposed method is improved compared with the traditional numerical integration method, it still has the possibility of further improvement for the multi-axis motion system will be subject to unavoidable disturbance forces, such as disturbances caused by mover current and stator magnetic field distortion, the change of mechanical parameters, time delay disturbance and so on. Therefore, considering the repetitive work characteristics of multi-axis motion systems such as advanced CNC machine tools, a compensation term is added to the dynamic model and then updated continuously by using iterative learning algorithm to further improve the dynamic model accuracy.

The rest of this paper is organised as follows. Section 2 studies the dynamic model of multi-axis motion system and derives the time-optimal mathematical model based on it. Section 3 proposes a time-optimal trajectory optimisation method similar to numerical integration and an iterative learning algorithm combined with it. Section 4 shows the experimental results of dynamic parameter identification and trajectory optimisation. Section 5 summarises this paper.

2 Time-optimal mathematical model

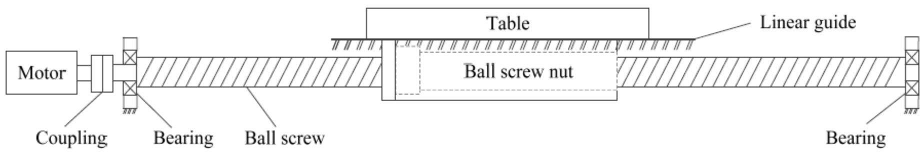
Time-optimal trajectory optimisation is essentially an optimisation problem, which can generally be solved by two steps. Initially, it is necessary to establish a mathematical model of practical problems, including selecting optimisation variables, determining the objective function and giving constraints, which are the main content of this section. In addition, appropriate optimisation methods should be applied to solve the mathematical model, which will be discussed in Section 3.

2.1 Dynamic model of multi-axis motion system

2.1.1 Dynamic modelling of multi-axis motion system

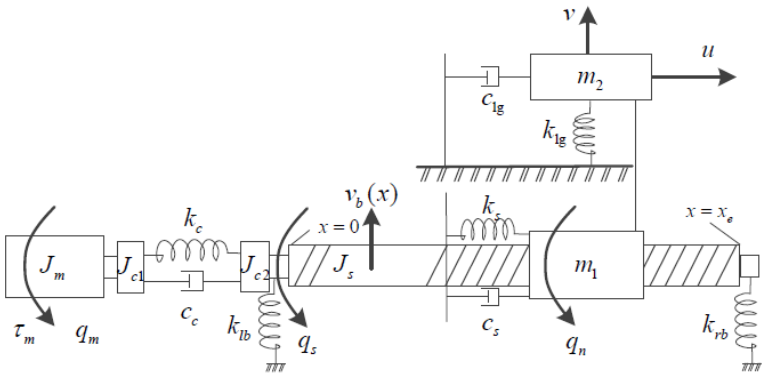
The schematic diagram of the single-axis transmission of the multi-axis motion system is shown in Figure 1. The motor output shaft is connected to the ball screw through a coupling, and the ball screw is installed on the bearings at both ends of the base. The ball screw nut or the slider is fixedly connected to the table with a bolt, so that the table can moves linearly along the linear guide.

Figure 1 Single-axis transmission diagram



The single-axis dynamic structure model of the multi-axis motion system (Zhang et al., 2017) is shown in Figure 2. The contact surfaces between the coupling, bearing, ball, table and linear guide are regarded as a spring damping system.

Figure 2 Single-axis dynamic structure model



Driven by the motor torque τ_m, the motor rotation angle is q_m and the screw rotation angle is q_s. After the ball transmission, the relative rotation angle of the screw nut is q_n, and the displacement of the table fixedly connected to the screw nut is u. In addition, considering the large shear and torsional stiffness, the screw can be regarded as a simply supported beam that does not deform and the radial displacement of the whole lead screw is consistent due to the bearing, that is, $v_b(x) = v_b$. The displacement in the vertical direction of the workbench in contact with the linear guide is v.

Because the established dynamic structure model has non-conservative forces such as damping force, motor driving force and Coulomb force on the table, the complete Lagrange's second equation can be applied for analysis.

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} + \frac{\partial R}{\partial q_i} = Q_i \tag{1}$$

where T is the system kinetic energy, U is the system potential energy, q_i is the generalised coordinate, R is the dissipation function, and Q_i is the generalised force in q_i.

The system kinetic energy includes the moment of inertia of the motor, the coupling and the screw, the kinetic energy of the ball screw nut and the table.

$$T = \frac{1}{2}(J_m + J_{c1})\dot{q}_m^2 + \frac{1}{2}(J_{c2} + J_s)\dot{q}_s^2 + \frac{1}{2}(m_1 + m_2)\dot{u}^2 + \frac{1}{2}(m_1 + m_2)\dot{v}^2 \tag{2}$$

where J_m, J_{c1} and J_{c2} are the moment of inertia of the motor, the coupling at the motor end and the coupling at the screw end respectively, m_1 and m_2 are the mass of the ball screw nut and the table respectively. $\cdot$ is the derivative of time t.

The system potential energy includes the elastic potential energy of the springs.

$$U = \frac{1}{2}k_c(q_m - q_s)^2 + \frac{1}{2}k_s(q_s - q_n)^2 + \frac{1}{2}(k_{lb} - k_{rb})v_b^2 + \frac{1}{2}k_{1g}v^2 \tag{3}$$

where k_c is the stiffness of the coupling, k_s and k_{lg} are the stiffness of the contact surface between the screw and the nut and the contact surface of the table and the linear guide respectively, k_{lb} and k_{rb} are the radial stiffness of the left and right bearings respectively.

The system dissipation function includes concentrated damping and structural damping of ball screws.

$$R=\frac{1}{2}C_c\left(\dot{q}_m-\dot{q}_s\right)^2+\frac{1}{2}C_s\left(\dot{q}_s-\dot{q}_n\right)^2+\frac{1}{2}C_{\mathrm{lg}}\dot{u}^2+f_c\cdot\mathrm{sgn}(\dot{u}) \tag{4}$$

where C_c is the damping coefficient of the coupling, C_s and C_{lg} are the damping coefficient of the contact surface between the screw and the nut and the contact surface between the table and the linear guide, f_c is the coulomb friction coefficient between the table and the linear guide.

From equations (1) to (4), the dynamic model can be obtained as:

$$\begin{aligned}\tau_m &= \left(J_m+J_{c1}\right)\ddot{q}_m+\left(J_{c2}+J_s\right)\ddot{q}_s+\left(m_1+m_2\right)\ddot{u}+\left(m_1+m_2\right)\ddot{v}\\ &\quad+\left(k_{lb}+k_{rb}\right)v_b+k_{\mathrm{lg}}v+C_{\mathrm{lg}}\dot{u}+f_c\cdot\mathrm{sgn}(\dot{u})\end{aligned} \tag{5}$$

In practical applications of multi-axis motion systems such as CNC machine tools, q_m is measurable while q_s, q_n and u are immeasurable. Therefore, q_s and q_n can be approximately substituted for q_m while u can be expressed by q_n as $u = q_n p_h/2\pi$ where p_h is the helical pitch of the screw. In addition, since the displacement of vibration v_b and v are too small to measure in practical application, they are approximately fitted by a constant term C. In summary, the dynamic model can be simplified as:

$$\tau_m = H_b\cdot\beta \tag{6}$$

where $H_b=\left[\ddot{q}_m \quad sign(\dot{q}_m) \quad \dot{q}_m \quad 1\right], \beta=\left[M \quad f_c \quad f_c \quad C\right]^T$, $f_v = p_h \cdot C_{\mathrm{lg}}/2\pi$, $M = J_m + J_{c1} + J_{c2} + J_s + p_h\,(m_1 + m_2)/2\pi$, $C=(m_1+m_2)\ddot{v}+(k_{lb}+k_{rb})v_b+k_{\mathrm{lg}}v$.

Next, s is defined as the path of the table of the multi-axis motion system, then the motor angle can be expressed as the function of s, i.e., $q(s)$. At the same time, s is the function of time t. Generally speaking, for any trajectory starting at $t = 0$ and ending at $t = T$, $s(0) = 0 \leq s(t) \leq 1 = s(T)$ (Xiao et al., 2012) is assumed. In order to facilitate the trajectory planning in phase plane $(s,\dot{s})$, the motor angular velocity $\dot{q}(t)$ and angular acceleration $\ddot{q}(t)$ are transformed into the function of the table path scalar s.

$$\dot{q}(t)=\dot{q}(s)=q'(s)\dot{s} \tag{7}$$

$$\ddot{q}(t)=\ddot{q}(s)=q'(s)\ddot{s}+q''(s)\dot{s}^2 \tag{8}$$

where

$$\dot{s}=ds/dt, \ddot{s}=d^2s/dt^2, q'(s)=\partial q(s), q''(s)=\partial^2 q(s)/\partial s^2.$$

By substituting equation (7) and (8) into equation (6), the path coordinate dynamic model of multi-axis motion system can be obtained as

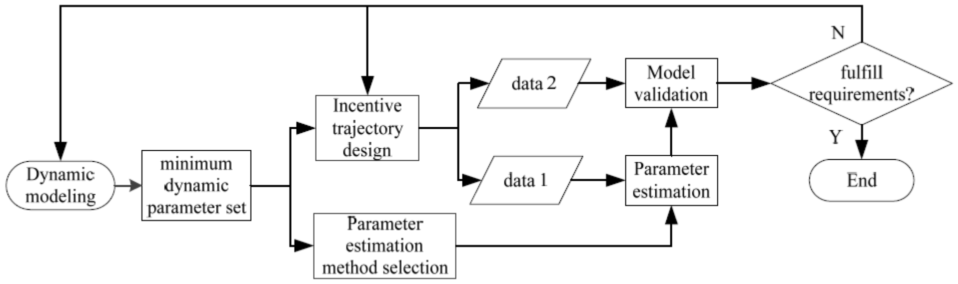
$$\tau=m(s)\cdot\ddot{s}+v(s)\cdot\ddot{s}^2+f(s)\cdot\dot{s}+c(s) \tag{9}$$

where $m(s)=M\cdot q'(s), v(s)=M\cdot q''(s), f(s)=f_v\cdot q'(s), c(s)=f_c\cdot\mathrm{sgn}[q'(s)]+C.$

2.1.2 Dynamic parameter identification of multi-axis motion system

The dynamic parameter identification process is shown in Figure 3. First, the established dynamic model is analysed to design two excitation trajectories and the parameter estimation method is selected based on the minimum dynamic parameter set. Then the motor moves along the designed excitation trajectories and the data including the motor angles and torques can be collected. One set of data is processed by the selected parameter estimation method to obtain the multi-axis motion system dynamic parameters. The other set of data is used to verify the estimated parameters, analyse the model errors, and judge whether the design requirements are met. If the requirements are met, the identification ends, otherwise, the dynamic model should be rebuilt or the excitation trajectory should be selected again.

Figure 3 Principle diagram of dynamic parameter identification


Dynamic modeling
minimum dynamic parameter set
Incentive trajectory design
Parameter estimation method selection
data 2
data 1
Model validation
Parameter estimation
fulfill requirements?
N
Y
End

In Section 2.1.1, the dynamic model has been established and the minimum dynamic parameter set has been determined. The excitation trajectory is designed by using the finite Fourier series with the basic frequency ω, which is related to the natural frequency of the system.

$$q_i(t) = \sum_{k=1}^{N_i} \frac{a_k^i}{\omega k}\sin(\omega kt) - \frac{b_k^i}{\omega k}\cos(\omega kt) + q_{i0} \tag{10}$$

Each Fourier series has $2 \times N_i + 1$ parameters, which include the amplitudes a_k^i and b_k^i ($k = 1, \ldots, N_i$) of the sine and cosine functions and the initial configuration of the motor q_{i0} when the excitation occurs. N_i is the order of Fourier series. Let $\delta = \{a_1^i, \cdots, a_{Ni}^i, b_1^i, \cdots, b_1^i \,|\, i = 1, \cdots, n\}$, where $n|$ is the number of axes of the motion system. The condition number of the observation matrix $A(\delta, \omega)$ is selected as the optimisation criterion so the design of the excitation trajectory can be regarded as a constraint optimisation problem.

$$\begin{aligned} &\hat{\delta} = \arg\min_{\delta} cond(A(\delta, \omega)) \\ &\text{with} \begin{cases} q_{\min} \le q(pT_s, \delta) \le q_{\max} \\ -\dot{q}_{\max} \le \dot{q}(pT_s, \delta) \le \dot{q}_{\max} \\ -\ddot{q}_{\max} \le \ddot{q}(pT_s, \delta) \le \ddot{q}_{\max} \end{cases} \\ &\text{for } 0 \le p \le T_f / T_s \end{aligned} \tag{11}$$

T_s is the sampling period of data collection in the experiment. The base frequency ω is fixed and its value depends on the required spectrum and data acquisition capabilities. $q_{\min}$, $q_{\max}$, $\dot{q}_{\max}$ and $\ddot{q}_{\max}$ are the vectors of the minimum and maximum values of the motor rotation angle, the maximum angular velocity and the maximum angular acceleration respectively.

The recursive least square parameter estimation algorithm (RLS) is used to estimate the dynamic parameters. Then the recurrence equation can be established according to equation (6) as

$$\begin{aligned} \hat{\beta}(k) &= \hat{\beta}(k-1) + K(k)\left[\tau_m(k) - H_b(k)\hat{\beta}(k-1)\right] \\ K(k) &= P(k-1) + H_b(k)\left[H_b^T(k)P(k-1)H_b(k)\right]^{-1} \\ P(k) &= \left[I - K(k)H_b^T(k)\right]P(k) - 1 \end{aligned} \tag{12}$$

The initial state $P(0)$ and $\hat{\beta}(0)$ can be obtained from a batch of data $q_m(1), \cdots, q_m(L_0)$ and $\tau_m(1), \cdots, \tau_m(L_0)$ by using a one-time completion algorithm.

$$\begin{aligned} P(0) &= [\Phi^T \Phi]^{-1} \\ \hat{\beta}(0) &= P(0)\Phi^T \Gamma \end{aligned} \tag{13}$$

$\Phi = [H_b(1), \cdots, H_b(L_0)]^T$, $\Gamma = [\tau_m(1), \cdots, \tau_m(L_0)]^T$, L_0 is the data length.

2.2 Time-optimal mathematical model

Given the motor velocity limits $\dot{q}_{\max}$ and $\dot{q}_{\min}$, the velocity limit inequality can be obtained as:

$$\dot{q}_{\min} \le \dot{q}(s) \le \dot{q}_{\max} \tag{14}$$

By substituting equation (7) into equation (14), the velocity limit inequality can be rewritten as:

$$\dot{q}_{\min} / q'(s) \le \dot{s} \le \dot{q}_{\max} / q'(s) \tag{15}$$

Given the motor acceleration limits max $\ddot{q}_{\max}$, and $\ddot{q}_{\min}$, the acceleration limit inequality can be obtained as:

$$\ddot{q}_{\min} \le \ddot{q}(s) \le \ddot{q}_{\max} \tag{16}$$

By substituting equation (8) into equation (16), the acceleration limit inequality can be rewritten as:

$$\left(\ddot{q}_{\min} - q''(s)\dot{s}^2\right) / q'(s) \le \ddot{s} \le \left(\ddot{q}_{\max} - q''(s)\dot{s}^2\right) / q'(s) \tag{17}$$

According to equation (9) and the given motor torque limits $\tau_{\max}$ and $\tau_{\min}$, the torque limit inequality can be obtained as:

$$\tau_{\min} \le m(s)\ddot{s} + v(s)\dot{s}^2 + f(s)\dot{s} + c(s) \le \tau_{\max} \tag{18}$$

The optimisation goal of time-optimal trajectory planning is to minimise the total execution time of the trajectory, so the objective function can be selected as:

$$\min T = \int_0^T 1dt \tag{19}$$

Replace the micro element *dt* with *ds*, the objective function can be expressed as:

$$\min T = \int_0^1 \frac{1}{\dot{s}} ds \tag{20}$$

Combining the objective function (20) and constraints (15), (17), and (18), the optimal objective mathematical model can be determined as:

$$\begin{aligned} &\min T = \int_0^1 \frac{1}{\dot{s}} ds \\ &\text{s.t.} \begin{cases} s(0) = 0 \quad s(T) = 1 \\ \dot{s}(0) = 0 \quad \dot{s}(T) = 1 \\ \dot{q}_{\min} / q'(s) \le \dot{s} \le \dot{q}_{\max} / q'(s) \\ \left(\dot{q}_{\min} - q''(s)s^2\right) / q'(s) \le \ddot{s} \le \left(\ddot{q}_{\max} - q''(s)\ddot{s}^2\right) / q'(s) \\ \tau_{\min} \le m(s)\ddot{s} + v(s)\dot{s}^2 + f(s)\dot{s} + c(s) \le \tau_{\max} \end{cases} \end{aligned} \tag{21}$$

3 Numerical integration-like optimisation method

In Section 2, the dynamic model of multi-axis motion system includes the nonlinear term of viscous friction. Compared with the linear dynamic model which ignores viscous friction, the nonlinear dynamic model undoubtedly increases the computational complexity of solving the time-optimal mathematical model. However, a more accurate dynamic model is conducive to get a better solution. In addition, using the torque calculated by the dynamic model as the feed-forward torque and combining with the feedback torque to design the control law can improve the control accuracy.

When it comes to solve the time-optimal mathematical model, the mathematical optimisation methods will lead to problems such as large amount of calculation while numerical integration methods are not suitable for nonlinear models. Therefore, in order to solve the nonlinear constrained time-optimal mathematical model and ensure the calculation efficiency, a numerical integration-like optimisation method is proposed. The method discretises the trajectory and assumes that the motion between any two adjacent discrete points is uniformly accelerated so that the uniform acceleration equation can be used to replace numerical integration method for planning.

Under the condition of ensuring the chord height error, the path of the table is discretised into *N* points.

$$\min T = \sum_{1}^{N} \frac{1}{\dot{s}_k}$$

$$\text{s.t.} \begin{cases} s_1 = 0 \quad s_n = 1 \\ \dot{s}_1 = 0 \quad \dot{s}_n = 1 \\ \underline{\dot{q}}(s_k)/q'(s_k) \le \dot{s}_k \le \overline{\dot{q}}(s_k)/q'(s_k) \\ \left(\underline{\ddot{q}}(s_k) - q''(s_k)\dot{s}_k^2\right)/q'(s_k) \le \ddot{s}_k \le \left(\overline{\ddot{q}}(s_k) - q''(s_k)\dot{s}_k^2\right)/q'(s_k) \\ \underline{\tau}(s_k) \le m(s_k)\ddot{s}_k + v(s_k)\dot{s}_k^2 + f(s_k)\dot{s}_k + c(s_k) \le \overline{\tau}(s_k) \\ \text{for } k = 1, \cdots, N \end{cases} \tag{22}$$

with $\underline{()}$ and $\overline{()}$ standing for the respective lower and upper limit. The first derivatives $q'(s_k)$ and second derivatives $q''(s_k)$ of the motor rotation angle at each discrete point can be obtained according to difference principle.

3.1 Maximum velocity limit curve

According to equation (9), the dynamic equation at any discrete point s_k after compensation can be expressed as:

$$\underline{\tau_i}(s_k) \le m_i(s_k)\ddot{s}_k + v_i(s_k)\dot{s}_k^2 + f_i(s_k)\dot{s}_k + c_i(s_k) + \delta_i(s_k) \le \overline{\tau_i}(s_k) \text{ for } i = 1, ..., n \tag{23}$$

where n is the degree of freedom of the multi-axis motion system, $\delta_i(s_k)$ is the compensation term of the dynamic model, which can be updated by iterative learning algorithm. The specific algorithm will be studied in Section 3.3.

When $m_i(s_k) \neq 0$, the upper limit $\alpha_{acc}(s_k, \dot{s}_k)$ and lower limit $\beta_{acc}(s_k, \dot{s}_k)$ of pseudo acceleration under kinematic constraint (acceleration constraint) can be obtained according to equation (17) while the upper limit $\alpha_{tor}(s_k, \dot{s}_k)$ and lower limit $\beta_{tor}(s_k, \dot{s}_k)$ of pseudo acceleration under dynamic constraint (torque constraint) can be obtained according to equation (18). Therefore, the upper and lower limits of pseudo acceleration under two kinds of constraints can be expressed as:

$$\alpha(s_k, \dot{s}_k) = \min\left(\alpha_{tor}(s_k, \dot{s}_k), \alpha_{acc}(s_k, \dot{s}_k)\right) \tag{24}$$

$$\beta(s_k, \dot{s}_k) = \max\left(\beta_{tor}(s_k, \dot{s}_k), \beta_{acc}(s_k, \dot{s}_k)\right) \tag{25}$$

When $m_i(s_k) = 0$, s_k is a zero inertia point. At this time the pseudo acceleration is only determined by the kinematic constraints, that is $\alpha(s_k, \dot{s}_k) = \alpha_{acc}(s_k, \dot{s}_k)$, $\beta(s_k, \dot{s}_k) = \beta_{acc}(s_k, \dot{s}_k)$.

It has been shown (Bobrow et al., 1985) that the maximum velocity limit curve under the acceleration and torque constraints is the curve composed of $\dot{s}_{\max,k}$ which satisfies equation (26) on the phase plane $(s, \dot{s})$.

$$\alpha(s_k, \dot{s}_k) = \beta(s_k, \dot{s}_k) \tag{26}$$

Equation (26) is usually nonlinear, which results in a lot of time to solve it. Therefore, the golden section method is considered to get the maximum velocity limit curve $MVC_{tor\&acc}$ under the acceleration and torque constraints. Then combining the maximum velocity limit curve MVC_{vel} obtained from the velocity constraint (15), the maximum velocity limit curve $MVC = \min(MVC_{tor\&acc}, MVC_{vel})$ satisfying the three constraints can be obtained finally.

3.2 Time-optimal trajectory optimisation algorithms

In order to improve the calculation efficiency, the uniform acceleration equation can be used instead of the numerical integration method to plan time-optimal trajectory of nonlinear dynamic model. The algorithm discretises the trajectories and assumes that any adjacent discrete points are uniformly accelerated with the maximum or the minimum pseudo-acceleration. According to the uniform acceleration equation, we can get the following equation

$$\dot{s}_k = \sqrt{2\ddot{s}_{\max/\min,k-1}\left(s_k - s_{k-1}\right) + \dot{s}_{k-1}^2} \tag{27}$$

where $\ddot{s}_{\max,k} = \alpha\left(s_k, \dot{s}_k\right)$, $\ddot{s}_{\min,k} = \beta\left(s_k, \dot{s}_k\right)$.

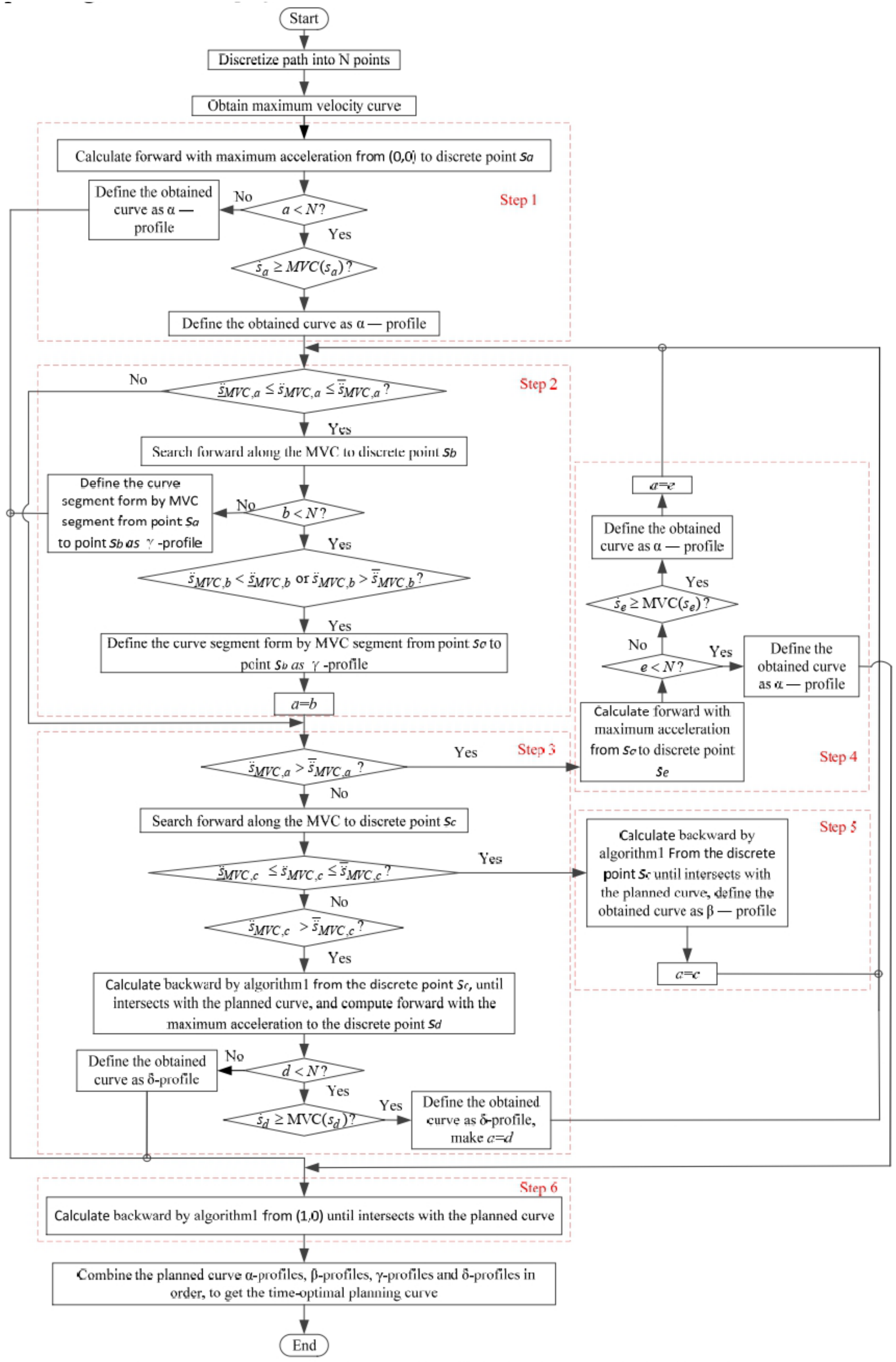
The process of time-optimal trajectory optimisation algorithm on the phase plane $(s, \dot{s})$ is shown in Figure 4, which consists of six steps.

Step 1 Calculate forwards, (i.e., increasing s) by the equation (27) from the initial point (0, 0) to the discrete point s_a with maximum pseudo acceleration $\ddot{s}_{\max,1}$. If $a = n$, go to step 6; if it intersects with MVC, that is, $\dot{s}_a \geq MVC(s_a)$, go to step 2. The obtained planning curve is $\alpha - profile$.

Step 2 Calculate $\ddot{s}_{MVC,a} = \left(\dot{s}_{MVC,a}^2 - \dot{s}_{MVC,a-1}^2\right)/2(s_a - s_{a-1})$, where $\dot{s}_{MVC}$ is the pseudo-velocity on MVC. If $\ddot{s}_{MVC,a} < \underline{\ddot{s}}_{MVC,a}$ or $\ddot{s}_{MVC,a} > \overline{\ddot{s}}_{MVC,a}$, go to step 3, where $\overline{\ddot{s}}_{MVC,a}$, and $\underline{\ddot{s}}_{MVC,a}$ represent the upper and lower limits of the pseudo acceleration obtained by equation (24) and (25) when the pseudo displacement is s_a and the pseudo velocity is $\dot{s}_{MVC,a}$; otherwise search forward along MVC to discrete point s_b. If $b = n$, go to step 6; if $\ddot{s}_{MVC,b} < \underline{\ddot{s}}_{MVC,b}$ or $\ddot{s}_{MVC,b} > \overline{\ddot{s}}_{MVC,b}$, interrupt search and let $a = b$. The obtained planning curve is $\gamma - profile$.

Step 3 Calculate $\ddot{s}_{MVC,a} = \left(\dot{s}_{MVC,a}^2 - \dot{s}_{MVC,a-1}^2\right)/2(s_a - s_{a-1})$. If $\ddot{s}_{MVC,a} > \overline{\ddot{s}}_{MVC,a}$, go to step 4; otherwise if $\ddot{s}_{MVC,a} < \underline{\ddot{s}}_{MVC,a}$, search forward along MVC to discrete point s_c. If $\underline{\ddot{s}}_{MVC,c} < \ddot{s}_{MVC,c} < \overline{\ddot{s}}_{MVC,c}$, go to step 5; otherwise if $\ddot{s}_{MVC,c} > \overline{\ddot{s}}_{MVC,c}$, use Algorithm 1 to calculate backwards from point s_c until it intersects the planned curve and calculate forward to point s_d with the maximum pseudo acceleration. If $d = n$, go to step 6; if $\dot{s}_d \geq MVC(s_d)$, let $a = d$ and go to step 2. The obtained planning curve is $\delta - profile$.

Figure 4 Flow diagram of time-optimal trajectory optimisation algorithm (see online version for colours)


Start
Discretize path into N points
Obtain maximum velocity curve
Calculate forward with maximum acceleration from (0,0) to discrete point s_a
Step 1
$a < N$?
No
Define the obtained curve as α — profile
Yes
$\dot{s}_a \geq MVC(s_a)$?
Define the obtained curve as α — profile
Step 2
$\ddot{s}_{MVC,a} \leq \ddot{s}_{MVC,a} \leq \bar{\ddot{s}}_{MVC,a}$?
No
Yes
Search forward along the MVC to discrete point s_b
$b < N$?
No
Define the curve segment form by MVC segment from point s_a to point s_b as γ -profile
Yes
$\ddot{s}_{MVC,b} < \ddot{s}_{MVC,b}$ or $\ddot{s}_{MVC,b} > \bar{\ddot{s}}_{MVC,b}$?
Yes
Define the curve segment form by MVC segment from point s_a to point s_b as γ -profile
$a=b$
Step 3
$\ddot{s}_{MVC,a} > \bar{\ddot{s}}_{MVC,a}$?
Yes
No
Search forward along the MVC to discrete point s_c
$\ddot{s}_{MVC,c} \leq \ddot{s}_{MVC,c} \leq \bar{\ddot{s}}_{MVC,c}$?
Yes
No
$\ddot{s}_{MVC,c} > \bar{\ddot{s}}_{MVC,c}$?
Yes
Calculate backward by algorithm1 from the discrete point s_c, until intersects with the planned curve, and compute forward with the maximum acceleration to the discrete point s_d
$d < N$?
No
Define the obtained curve as δ-profile
Yes
$\dot{s}_d \geq MVC(s_d)$?
Yes
Define the obtained curve as δ-profile, make $a=d$
Step 4
$a=e$
Define the obtained curve as α — profile
Yes
$\dot{s}_e \geq MVC(s_e)$?
No
$e < N$?
Yes
Define the obtained curve as α — profile
Calculate forward with maximum acceleration from s_a to discrete point s_e
Step 5
Calculate backward by algorithm1 From the discrete point s_c until intersects with the planned curve, define the obtained curve as β — profile
$a=c$
Step 6
Calculate backward by algorithm1 from (1,0) until intersects with the planned curve
Combine the planned curve α-profiles, β-profiles, γ-profiles and δ-profiles in order, to get the time-optimal planning curve
End

Step 4 Calculate forwards from point s_a to point s_e with the maximum pseudo acceleration $\ddot{s}_{\max,a}$. If $e = n$, go to step 6; if it intersects with MVC, that is $\dot{s}_e \geq MVC(s_e)$, let $a = e$ and go to step 2. The obtained planning curve is $\alpha - profile$.

Step 5 Calculate backwards, (i.e., decreasing s) from point s_c by using Algorithm 1 until the curve intersects with the planned curve. The obtained planning curve is $\beta - profile$.

Step 6 Calculate backwards, (i.e., decreasing s) from point (1, 0) by using Algorithm 1 until the curve intersects with the planned curve. The obtained planning curve is $\beta - profile$. Finally, the obtained curves $\alpha - profile$, $\beta - profile$, $\gamma - profile$ and $\delta - profile$ are combined into a curve in order, which is the time-optimal planning curve.

Algorithm 1 The pseudo velocity interval $(\dot{s}_{k-1}, \dot{s}_k)$ of any two discrete points s_{k-1} and s_k is equally divided into m parts, obtaining $m + 1$ points which are recorded as $\dot{s}_k(m), \dot{s}_k(m-1), ..., \dot{s}_k(0)$. Start searching from $\dot{s}_{k-1}(0)$ along the $\dot{s}$ axis until finding i that satisfies $\ddot{s}_{com}(s_k, \dot{s}_k(i-1)) < \beta(s_k, \dot{s}_k(i-1))$ and $\ddot{s}_{com}(s_k, \dot{s}_k(i)) > \beta(s_k, \dot{s}_k(i))$, i.e., finding the interval $(\dot{s}_k(i-1), \dot{s}_k(i))$ containing $\dot{s}_k(x)$ who satisfies $\ddot{s}_{com}(s_k, \dot{s}_k(x)) = \beta(s_k, \dot{s}_k(x))$, and then use the golden section method to obtain $\dot{s}_k(x)$.

3.3 Iterative learning algorithm

In order to further improve the dynamic model accuracy, iterative learning algorithm is applied to update the compensation term of the dynamic model. The control block diagram is shown in Figure 5. First of all, the desired positions and feed forward torques are obtained based on the dynamic model by using the time-optimal trajectory optimisation algorithm proposed in Section 3.2. Then the motor feedback positions and actual torques during the experiments are used in iterative learning algorithm to modify the dynamic model. After that, the trajectory is optimised again based on the revised model. Follow this process and the time-optimal trajectory is gradually close to the optimal solution.

When the bandwidth of the velocity loop is at least three times larger than that of the position loop and the step response does not oscillate, the velocity loop can be simplified as a unit gain (Shih et al., 2002). Therefore, the plant can be simplified as shown in Figure 5, where K_p is the position loop gain and T is the sampling period of the system. To sum up, the input-output relationship of the system can be written as:

$$s_{i,l,fb} = \frac{1}{Ts + K_p}\tau_{i,l} + \frac{K_p}{Ts + K_p}s_{i,l,r} \tag{28}$$

where i is the serial number of the axis, l is the number of iterations, $s_{i,l,fb}$ is the feedback position, $s_{i,l,r}$ is the desired position, $\tau_{i,l}$ is the calculated torque.

The open-closed loop PD type iterative learning method with forgetting factor is designed as shown in Figure 5, whose control law is:

$$\tau_{i,l+1} = Q\{[1-r(l)]\tau_{i,l} + r(l)\tau_{i,0} + L_p e_{i,l} + L_d \dot{e}_{i,l}\} \tag{29}$$

where $r(l)$ is the forgetting factor, Q, L_p and L_d are gains, $e_{i,l} = s_{i,l,fb} - s_{i,l,r}$, $\dot{e}_{i,l} = \frac{e_{i,l} - \dot{e}_{i,l-1}}{T}$.

By analysing the lifted matrix (Barton and Alleyne, 2007) of equation (28) and (29), the lifted expression of $s_{i,l,fb}$ and $\tau_{i,l+1}$ can be obtained as:

$$\hat{s}_{i,l,fb} = \hat{P}\hat{\tau}_{i,l} + \hat{d} \tag{30}$$

$$\hat{\tau}_{i,l+1} = Q\left(\hat{R}_l\hat{\tau}_{i,l} + \hat{L}\hat{e}_{i,l}\right) \tag{31}$$

where $\hat{s}_{i,l,fb} = \begin{bmatrix} s_{i,l,fb}(1) \\ s_{i,l,fb}(2) \\ \vdots \\ s_{i,l,fb}(n) \end{bmatrix}$, $\hat{\tau}_{i,l} = \begin{bmatrix} \tau_{i,l}(1) \\ \tau_{i,l}(2) \\ \vdots \\ \tau_{i,l}(n-1) \end{bmatrix}$, $\hat{P} = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_n & p_{n-1} & \cdots & p_1 \end{bmatrix}$,

$\hat{R}_l = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ r(l) & 1-r(l) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r(l) & p_{n-1} & \cdots & 1-r(l) \end{bmatrix}$, $\hat{L} = \begin{bmatrix} L_p + \frac{L_d}{T} & 0 & \cdots & 0 & 0 \\ -\frac{L_d}{T} & L_p + \frac{L_d}{T} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & L_p + \frac{L_d}{T} & 0 \\ 0 & 0 & \cdots & -\frac{L_d}{T} & L_p + \frac{L_d}{T} \end{bmatrix}$, $\hat{R}_l$ and $\hat{L}$

are $n \times n$ matrices.

The lifted expression of error signal is:

$$\hat{e}_{i,l} = \hat{s}_{i,l,r} - \hat{s}_{i,l,fb} = \hat{s}_{i,l,r} - \left(\hat{P}\hat{\tau}_{i,l} + \hat{d}\right) \tag{32}$$

Combine equations (30) ~ (32), the lifted expression of $\tau_{i,l+1}$ can be expressed as:

$$\hat{\tau}_{i,l+1} = Q\left(\hat{R}_l - \hat{L}\hat{P}\right)\hat{e}_{i.l} + \hat{W} \tag{33}$$

For any given initial control and initial state of each operation process, the necessary and sufficient condition for the iterative learning algorithm to converge uniformly is:

$$\left|\lambda_k\left(Q\left(\hat{R}_l - \hat{L}\hat{P}\right)\right)\right| < 1 \ \ \forall k \in [1, n] \tag{34}$$

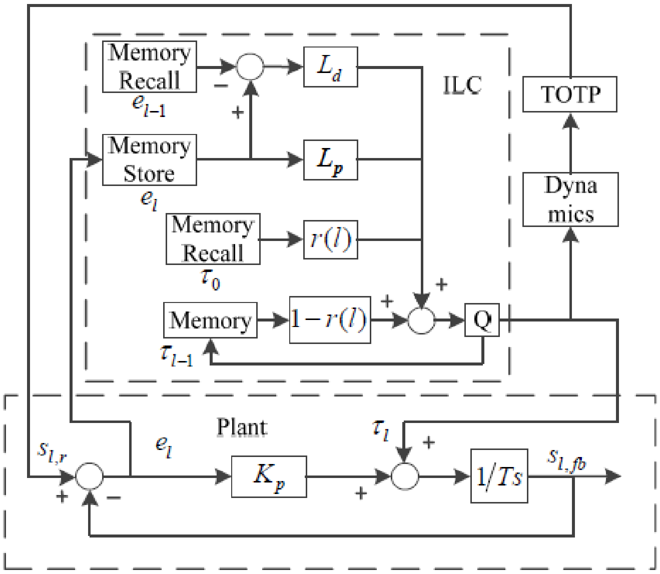
According to the path coordinate dynamic model (9), the dynamic equation of any point s_k on the trajectory after discretisation can be expressed as:

$$\tau_{i,l+1}(s_k) = m_{i,l+1}(s_k)\ddot{s}_k + v_{i,l+1}(s_k)\dot{s}_k^2 + f_{i,l+1}(s_k)\dot{s}_k + c_{i,l+1}(s_k) + \delta_{i,l+1}(s_k) \tag{35}$$

Further, iterative equation of the compensation term of the dynamic model can be obtained as:

$$\delta_{i,l+1}(s_k) = Q\{[1-r(l)]\delta_{i,l}(s_k) + r(l)\delta_{i,0}(s_k) - L_p(s_k)e_{i,l}(s_k) - L_d(s_k)\dot{e}_{i,l}(s_k)\} \quad (36)$$

Figure 5 Iterative learning algorithm control block diagram



4 Experimental verification and discussion

As shown in Figure 6, the experimental platform is a multi-axis motion system, which is composed of X and Z axes in series. The types of the motors for the two axes are Delta ECMA-CA0604RS, which are driven by Delta ASD-A2-0421-E AC motion driver, and then their rotary motion is converted into linear motion of the worktable through the ball screw. The experiments of dynamic parameter identification and time-optimal trajectory optimisation are carried out on this platform. The excitation trajectory optimisation, dynamic parameter identification, time-optimal trajectory optimisation and iterative learning algorithm of multi-axis motion system are calculated by MATLAB R2018b. The control system of the platform is built on Windows 7 64-bit system and uses EtherCAT for communication. Its control cycle and sampling cycle are both 1ms. Besides, the industrial computer model is DT-610P-ZQ170MA, the CPU is Intel Core i7-4770, the highest main frequency is 3.40GHz, and the operating memory is 8 GB.

4.1 Dynamic parameter identification experiment

Based on equation (11) in Section 2.1.2 and the SDP algorithm, a MATLAB program is designed to obtain the optimal excitation trajectory, whose parameters $\hat{\delta}$ are shown in Table 1.

Figure 6 Experimental platform (see online version for colours)

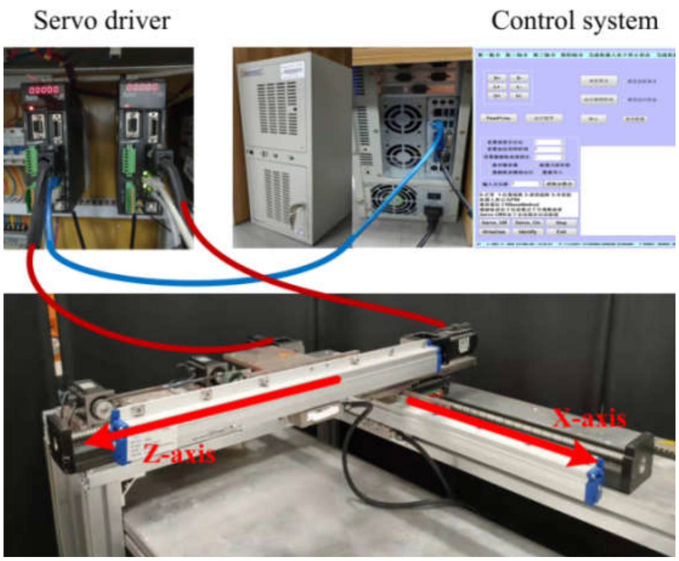


Table 1 Excitation trajectory parameters

Parameters	*X-axis*	*Z-axis*	*Parameters*	*X-axis*	*Z-axis*
q_{i0} (rad)	14.702	9.043	~	~	~
a_1^i	–12.412	16.861	b_1^i	–0.397	–7.044
a_2^i	–2.012	–9.419	b_2^i	3.680	–5.035
a_3^i	–2.288	–1.306	b_3^i	–20.433	3.770
a_4^i	26.881	11.015	b_4^i	–16.883	9.685
a_5^i	8.515	6.755	b_5^i	–4.404	–2.110
a_6^i	–1.159	–3.296	b_6^i	19.220	–6.619

Let the multi-axis motion system move along the excitation trajectory and then the collected motor angles and actual torques can be processed by using the Butterworth filter and central difference method. The initial states of dynamic parameters are calculated according to equation (13), and the dynamic parameters are calculated iteratively according to equation (12). The iterative processes are shown in Figure 7.

Table 2 Errors of the calculated and actual torques of the motors

	X-axis	*Z-axis*
IAE	7.2%	14.2%

For another verification trajectory, the calculated torques of each axes are determined by using the identified dynamic parameters, which are compared with the collected actual torques as shown in Figure 8. The blue curve represents the actual torques, the red curve represents the computed torques and the green curve represents the errors of the torques. It can be seen that the calculated torque curve of each axis is basically consistent with the actual torque curve, but the difference between them is obvious near the reverse points of the axis torques caused by the noise of acceleration signal, the inaccurate inertia matrix identification in the process of dynamic parameter identification and the reverse friction force. In general, the integral absolute errors (IAE) of the two axis torques are shown in Table 2, where the value of X axis is 7.2% and that of Z axis is 14.2%.

Figure 7 Iterative processes of dynamic parameters (see online version for colours)

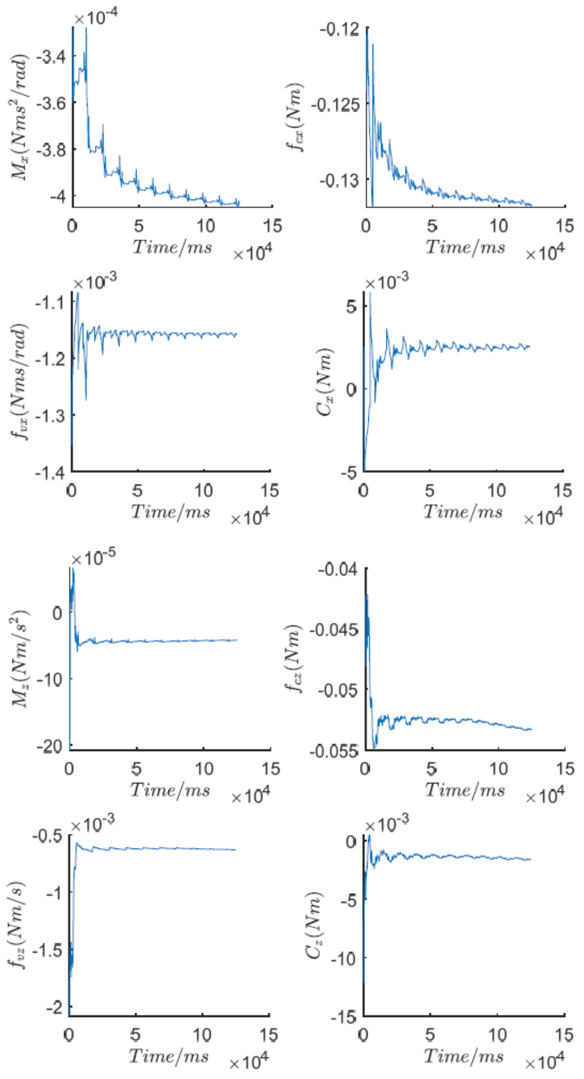
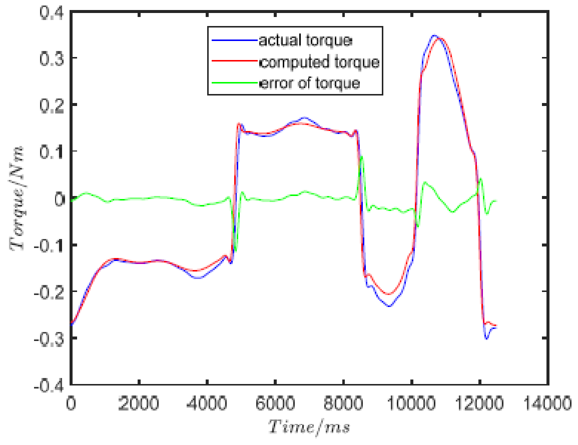
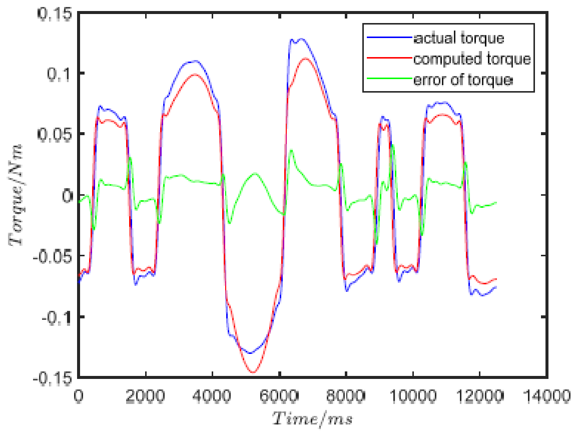


Figure 8 Comparison of torques of the axes in the identification experiment, (a) X-axis torques, (b) Z-axis torques (see online version for colours)



(a)



(b)

4.2 Time-optimal trajectory optimisation experiment

There are several ways to define geometric trajectory, including polynomial, the combination of straight line, arc and cycloid (Müller et al., 2007) and spline (Bobrow et al., 1985), among which spline is more suitable for describing the motion of platform (Oberherber et al., 2015). In the practical application of multi-axis motion system, machining paths are always determined by multi-purpose spline curves, most of which are NURBS curves. Considering the complexity and variability of the actual machining paths, the NURBS curve interpolator (Cheng et al., 2002) is used to generate a 'star shaped' curve path with variable curvature to simulate the machining path, which is used for verifying the effectiveness of the algorithm proposed. According to the effective stroke of the experimental platform, the selected parameters of the star curve are shown in Table 3, and the trajectory generated by the interpolation is shown in Figure 9.

Table 3 'Star curve' parameters

Number	*k = 2*				
Node vector	*U = {0, 0, 0, 1/9, 2/9, 3/9, 4/9, 5/9, 6/9, 7/9, 8/9, 1, 1, 1}*				
Serial number of vertex	*Control point coordinates (mm)*	*Weight factor*	*Serial number of vertex*	*Control point coordinates (mm)*	*Weight factor*
1	(0.0, 0.0)	1	6	(108.0, 0.0)	1
2	(48.0, 24.0)	1	7	(144.0, –40.0)	0.7
3	(48.0, 64.0)	1	8	(96.0, –32.0)	1
4	(96.0, 32.0)	1	9	(48.0, –64.0)	1
5	(144.0, 40.0)	0.7	10	(48.0, –24.0)	1

Table 4 Trajectory execution time (ms)

Sample size N	*1,005*	*2,079*	*3,014*	*4,018*	*4,920*	*6,028*	*7,008*	*7,930*	*8,909*	*10,045*
$c = 10$	1,735	1,733	1,735	1,736	1,735	1,733	1,735	1,737	1,736	1,736
$c = 5$	1,736	1,736	1,736	1,737	1,736	1,737	1,735	1,736	1,737	1,736
$c = 2$	1,738	1,738	1,738	1,738	1,739	1,739	1,739	1,738	1,739	1,739
$c = 1$	1,738	1,740	1,740	1,740	1,739	1,739	1,740	1,739	1,739	1,739
$c = 0.5$	1,739	1,741	1,740	1,740	1,740	1,740	1,740	1,740	1,740	1,740
$c = 0.2$	1,739	1,741	1,740	1,740	1,740	1,741	1,740	1,740	1,740	1,740
$c = 0.1$	1,739	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741
$c = 0.05$	1,739	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741
$c = 0.02$	1,739	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741
$c = 0.01$	1,739	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741	1,741

It should be noted that the parameters such as the sample size and trajectory optimisation accuracy might influence the solution accuracy. Therefore, robustness of the optimisation algorithm should be analysed and a sensitivity research (Chai et al., 2020) of the sample size N and trajectory optimisation accuracy on the numerical integration-like (NI-like) method has been carried out. As is known, trajectory optimisation accuracy is related to curvature change rate c. For the trajectory shown in Figure 9, the trajectory execution time is generated by correspondingly selecting the values of N and c from the sets (1,005, 2,079, 3,014, 4,018, 4,920, 6,028, 7,008, 7,930, 8,909, 10,045) and (10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01), which is shown in Table 4. It can be observed that the trajectory execution time does not vary significantly as the sample number N and curvature change rate c increase. Besides, the algorithm will converge gradually with the decrease of the curvature change rate of the trajectory when the sample size is given. This is because when the curvature change rate of the trajectory is small to a certain value which depends on the given trajectory, the discretised trajectory can completely describe the original trajectory.

The optimal mathematical model (21) can actually be regarded as a convex-concave problem, which can be solved in several ways. An intuitive approach to solve the problem as a NLP (Steinhauser and Swevers, 2005) that can be solved by utilising, e.g., an interior point solver like IPOPT, which is used for comparing with NI-like algorithm by solving the optimal path tracking problem. The calculation time of the two methods is shown in Figure 10 as the sample number N increases, where for every number of discretisation points and each method, the problem is solved ten times to take variations into account.

Figure 9 Star curve (see online version for colours)

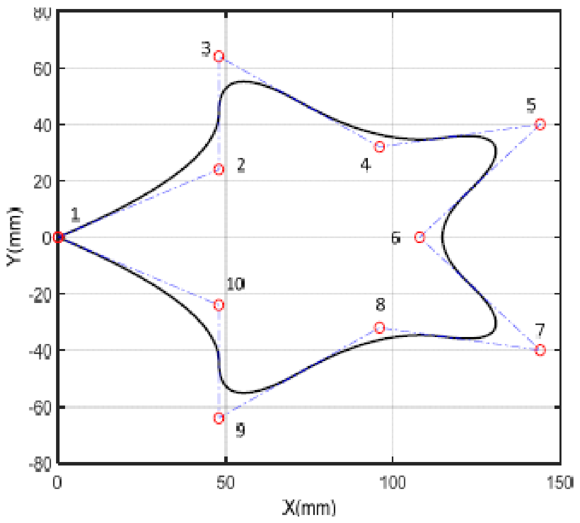
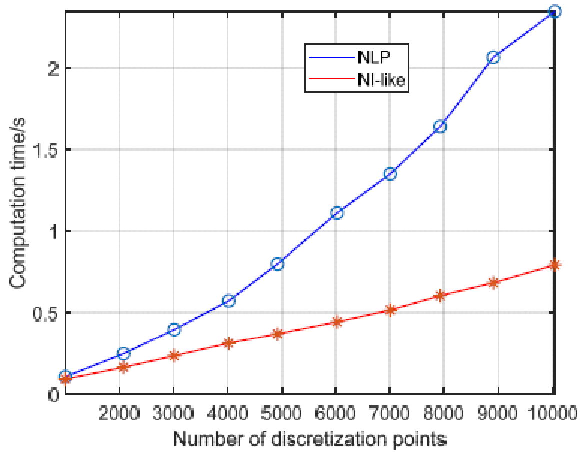


Figure 10 Comparison of the calculation time of the NLP and the NI-like algorithm (see online version for colours)



In either case, the calculation time scales approximately linearly with the number of discretisation points. However, compared with NLP method, the NI-like method has less calculation time which is crucial in case the problem size increases.

As mentioned above, the star curve track with 6028 discrete points is selected for analysis. Figure 11 and Figure 12 show the planning results of NI-like algorithm and NLP algorithm in phase plane $(s, \dot{s})$ respectively. The velocity, acceleration and torque curves obtained by NI-like algorithm and NLP algorithm are shown in Figure 12, where the black dotted lines are upper and lower limit. It can be seen from Figure 13 that the velocity, acceleration and torque of the planning trajectory obtained by NI-like algorithm are all under the limited conditions, which verifies the effectiveness of the algorithm. However, since the NLP algorithm has no mandatory constraints, the velocity and torque of X axis and the velocity and acceleration of Z axis obtained by the algorithm exceed the limits. The same results can be seen from the phase planes of Figure 11 and Figure 12: some points obtained by NLP algorithm exceed the limit while the points obtained by NI-like algorithm are all under the limit curve and can fit it well.

Figure 11 Trajectory planning results of NI-like algorithm in phase plane (see online version for colours)

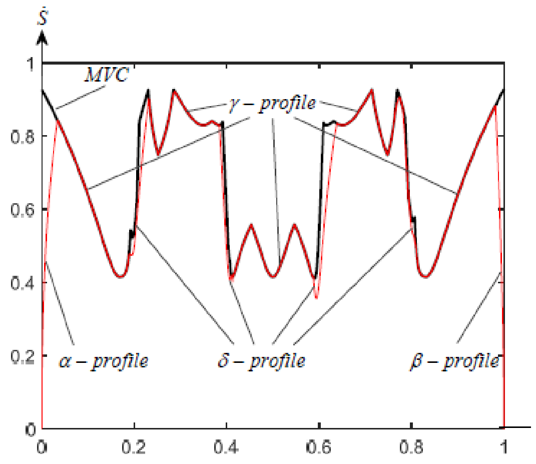


Figure 12 Trajectory planning results of NLP algorithm in phase plane (see online version for colours)

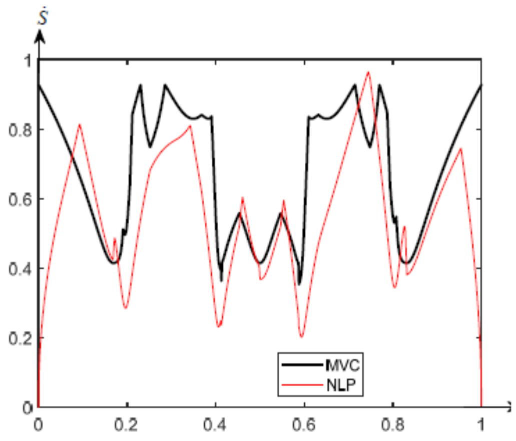
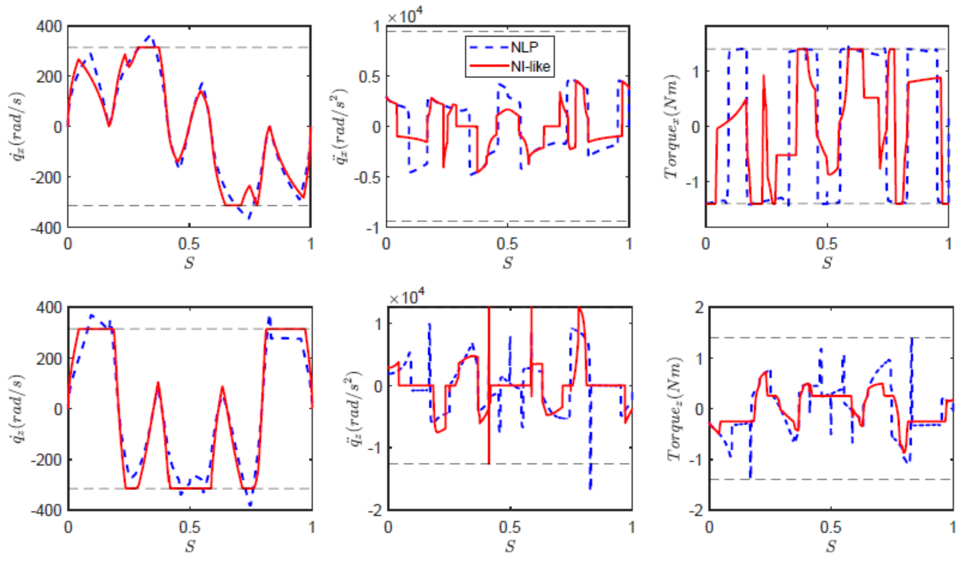


Figure 13 Velocity, acceleration and torque of planning track (see online version for colours)



4.3 *Iterative learning experiment*

For the star curve trajectory given in Section 4.2, the iterative learning algorithm in Section 3.3 is applied to continuously modify the dynamic model to optimise the time-optimal trajectory. The comparison between the optimisation results of the proposed iterative learning algorithm and the open-closed loop PD-type iterative learning method is shown in Figures 14, 15, and 16.

Figure 14 Trajectory execution time (see online version for colours)

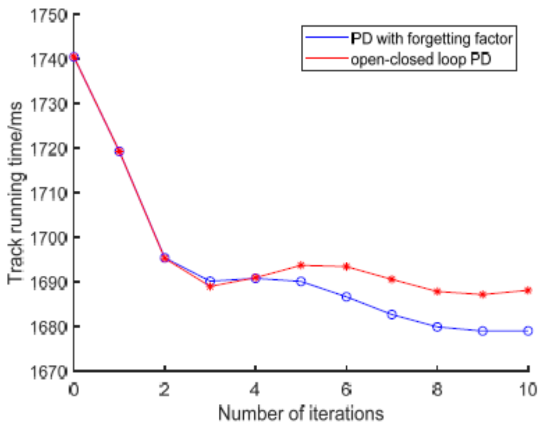
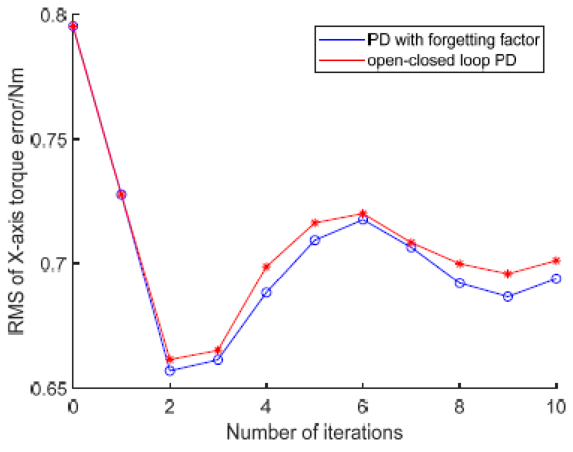


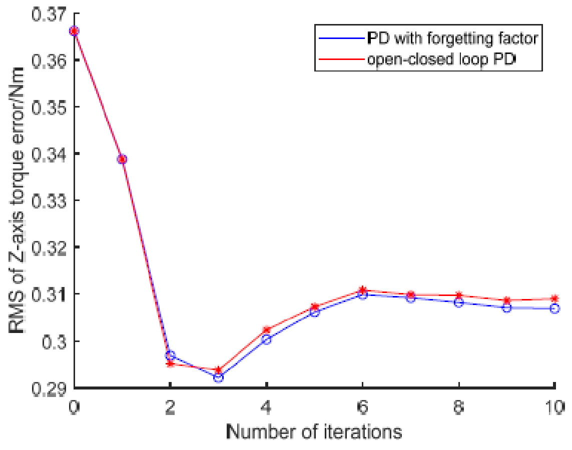
Figure 14 shows the change of the trajectory execution time after ten iterations. It can be seen from the figure that compared with the 0th iteration; the trajectory execution time of PD type iterative learning method with forgetting factor is reduced by 3.5% while that of open-closed loop PD type iterative learning method is reduced by 3.0% after ten iterations. Furthermore, both algorithms are effective, and the PD-type iterative learning method with forgetting factor is slightly better than the open-closed PD-type iterative learning method.

Figure 15 shows the change of root mean square (RMS) of the errors between the calculated torques and the actual torques after ten iterations. It can be seen from the figure that after ten iterations, the RMS of X and Z-axis torque errors of PD type iterative learning method with forgetting factor are reduced by 23.9% and 29.7% respectively while those of open-closed loop PD type iterative learning method are reduced by 22.3% and 28.8% respectively. In addition, the iterative effect of X-axis is not as ideal as that of Z-axis. This is because the Z-axis is mounted on the X-axis, and the movement of the Z-axis will have a mechanical effect on the X-axis, thus making the actual dynamic model of the X-axis is more complicated than that of the Z-axis.

Figure 15 RMS of torque error, (a) RMS of X-axis torque error, (b) RMS of Z-axis torque error (see online version for colours)



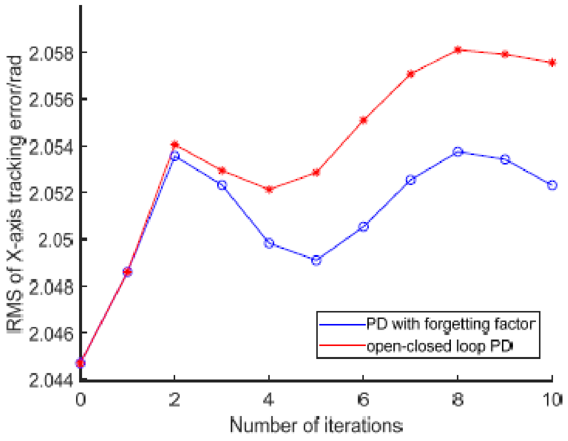
(a)



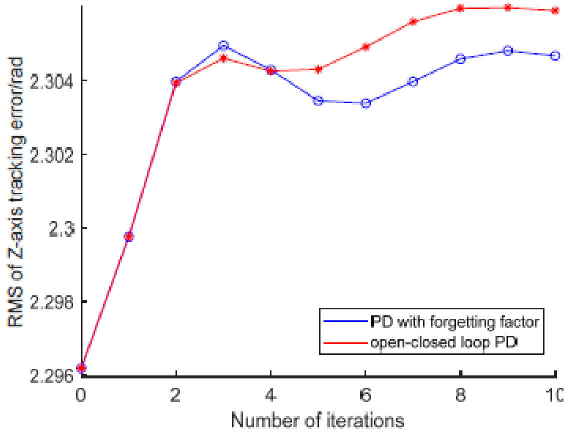
(b)

Figure 16 shows the changes of RMS of tracking error of each axis in the process of ten iterations. It can be seen from the figure that the fluctuation ranges of the RMS of the X and Z-axis tracking errors by using PD type iterative learning method with forgetting factor are within 0.89% and 0.76% respectively while those by using open-closed loop PD type iterative learning method are within 1.32% and 0.85% respectively, which shows that both methods can guarantee the tracking errors.

Figure 16 RMS of tracking error, (a) RMS of X-axis tracking error RMS of Z-axis tracking error/rad, (b) RMS of Z-axis tracking error (see online version for colours)



(a)



(b)

As shown in Figure 15, it is clear that the torque errors on each axis of the first three iterations are decreased independently due to the improved feed forward torque accuracy, which means that the dynamic model accuracy is improved. However, because of interference factors such as the environment's noise on the experimental platform, the model has been overcompensated since the fourth iteration, resulting in torque error fluctuations. In addition, the execution time of the trajectory optimised by proposed

NI-like algorithm is decreased as shown in Figure 14 while the tracking error fluctuates as shown in Figure 16 in actual operation process, which is a normal phenomenon of motor control: the motor tracking error will fluctuate as its velocity fluctuates.

5 Conclusions

A numerical integration-like time-optimal trajectory optimisation algorithm is proposed, which minimises the execution time of the motion along the geometric path while taking the motor limitations into account to ensure feasibility. The proposed algorithm not only retains the advantages of small amount of calculation of the numerical integration method, but also considers the nonlinear terms of the dynamic model. The PD-type iterative learning method with forgetting factor is used to continuously update the dynamic model and then performs trajectory optimisation by the proposed time-optimal trajectory optimisation algorithm. Experiments show that after ten iterations, the execution time of the selected star curve is reduced by 3.5%, and the torque errors of X-axis and Z-axis are reduced by 23.9% and 29.7% respectively. In the process of ten iterations, the fluctuation ranges of RMS of the X and Z-axis tracking errors are within 0.89% and 0.76% respectively. Therefore, it is obvious that the proposed method can improve the working efficiency while ensuring tracking accuracy for repetitive operations of multi-axis motion system.

References

Barton, K.L. and Alleyne, A.G. (2007) 'Cross-coupled ILC for improved precision motion control: design and implementation', *7th American Control Conference*.

Bellahcene, Z., Bouhamida, M., Denai, M. and Khaled, A. (2021) 'Adaptive neural network-based robust h∞ tracking control of a quadrotor uav under wind disturbances', *International Journal of Automation and Control*, Vol. 15, No. 1, p.28.

Bobrow, J., Dubowsky, S. and Gibision, J. (1985) 'Time-optimal control of robotic manipulators along specified paths', *The International Journal of robotics Research*, Vol. 4, No. 3, pp.3–17.

Capito, L., Proao, P., Camacho, O., Rosales, A. and Scaglia, G. (2016) 'Experimental comparison of control strategies for trajectory tracking for mobile robots', *International Journal of Automation and Control*, Vol. 10, No. 3, p.308.

Chai, R., Savvaris, A., Tsourdos, A., Chai, S. and Xia, Y. (2020) 'Stochastic spacecraft trajectory optimization with the consideration of chance constraints', *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 4, pp.1550–1559.

Cheng, M.Y., Tsai, M.C. and Kuo, J.C. (2002) 'Real-time NURBS command generators for CNC servo controllers', *International Journal of Machine Tools & Manufacture*, Vol. 42, No. 7, pp.801–813.

Constantinescu, D. and Croft, E.A. (2000) 'Smooth and time-optimal trajectory planning for industrial manipulators along specified path', *Journal of Robotic Systems*, Vol. 17 No. 5, pp.233-249.

Debrouwere, F., Loock, W.V., Pipeleers, G. and Dinh, Q.T. (2013) 'Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming', *IEEE International Conference on Robotics & Automation*.

Guevara, L., Camacho, O., Rosales, A., Guevara, J. and Scaglia, G. (2019) 'A linear algebra controller based on reduced order models applied to trajectory tracking for mobile robots: an experimental validation', *International Journal of Automation and Control*, Vol. 13, No. 2, p.176.

Kaserer, D., Gattringer, H. and Mueller, A. (2018) 'Nearly optimal path following with jerk and torque rate limits using dynamic programming', *IEEE Transactions on Robotics*, Vol. 35, No. 2, pp.521–528.

Mattmuller, J. and Gisler, D. (2009) 'Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path', *International Journal of Advanced Manufacturing Technology*, Vol. 45, Nos. 9–10, pp.1007–1016.

Mohamed, A., Ren, J., Lang, H. and El-Gindy, M. (2018) 'Optimal path planning for an autonomous articulated vehicle with two trailers', *International Journal of Automation and Control*, Vol. 12, No. 3, pp.449–465.

Müller, B., Deutscher, J. and Grodde, S. (2007) 'Continuous curvature trajectory design and feedforward control of parking a car', *IEEE Transactions on Control Systems Technology*, Vol. 15, No. 3, pp.541–553.

Oberherber, M., Gattringer, H. and Müller, A. (2015) 'Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking', *Mechanical Sciences*, Vol. 6, No. 2, pp.245–254.

Ouda, A.N., Lang, H., Gindy, M.E., Jing, R. and Mohamed, A. (2018) 'Literature survey for autonomous vehicles: sensor fusion, computer vision, system identification and fault tolerance', *International Journal of Automation and Control*, Vol. 12, No. 4, p.555.

Reynosomora, P., Chen, W. and Tomizuka, M. (2013) 'On the time-optimal trajectory planning and control of robotic manipulators along predefined paths', *American Control Conference*.

Shih, Y.T., Chen, C. and Lee, A.C. (2002) 'A novel cross-coupling control design for bi-axis motion', *International Journal of Machine Tools and Manufacture*, Vol. 42, No. 14, pp.1539–1548.

Slotine, J.J.E. and Yang, H.S. (2002) 'Improving the efficiency of time-optimal path-following algorithms', *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 1, pp.118–124.

Steinhauser, A. and Swevers, J. (2005) 'An efficient iterative learning approach to time-optimal path tracking for industrial robots', *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 11, pp.5200–5207.

Xiao, Y., Du, Z. and Dong, W. (2012) 'Smooth and near time-optimal trajectory planning of industrial robots for online applications'. *Industrial Robot: An International Journal*, Vol. 39, No. 2, pp.169–177.

Zhang, L., Wang, T., Wang, G. and Tian, S. (2017) 'Hybrid dynamic modeling and analysis of a ball-screw-drive spindle system', *Journal of Mechanical Science and Technology*, Vol. 31, No. 10, pp.4611–4618.

Ziadi, S., Njah, M. and Chtourou, M. (2021) 'Autonomous PSO-DVSF2: an optimised force field mobile robot motion planning approach for unknown dynamic environments', *International Journal of Automation and Control*, Vol. 15, No 3, pp.318–339.