ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2026.0054

Learning-based data-driven control for micro-nano free-floating space robots in Cartesian space

Renhao MAO[1], Tao MENG[1,2]✉, Kun WANG[3], Zhonglin ZUO[4], Hang ZHOU[1], Shujian SUN[1,2]

[1]*School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China*
[2]*Huanjiang Laboratory, Zhejiang University, Zhuji 311899, China*
[3]*Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China*
[4]*College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*

Abstract: This paper addresses the problem of end-effector position-tracking control for micro-nano free-floating space robots in Cartesian space without relying on explicit analytical kinematic or dynamic models. To address this challenge, we develop a two-layer learning architecture. In the first layer, a deep neural network is used for kinematic learning to capture the nonlinear mapping from end-effector Cartesian coordinates to joint angular velocities and to generate reference joint trajectories. In the second layer, a Koopman-operator-based network is employed to construct an approximately linearized representation of the joint-space dynamics of free-floating space robots. Based on this model, we propose a terminal fractional-order model predictive control scheme that incorporates the Grünwald–Letnikov fractional-order operator, thereby enhancing online control performance and improving tracking speed and accuracy relative to conventional model predictive control. Simulation results verify the effectiveness of the proposed method, demonstrating accurate and rapid end-effector trajectory tracking, all without requiring explicit analytical kinematic and dynamic models in the controller design, while the training pipeline relies solely on input–output trajectories.

Key words: Free-floating space robot; Deep neural networks; Model learning for control; Model predictive control

1 Introduction

Free-floating space robots (FFSRs) have been widely employed in space missions, including tasks such as space station maintenance and the retrieval of uncontrolled satellites (Fallahiarezoodar and Zhu, 2025). In particular, micro-nano space robots, which generally weigh between a few kilograms and several hundred kilograms, are attracting growing interest owing to their cost-effectiveness and deployment flexibility (Tu et al., 2024). Nevertheless, the development of accurate dynamic models for micro-nano space robots is particularly challenging due to the significant dynamic coupling effects arising from the higher inertia and mass ratios between their manipulators and base platforms, compared with larger robotic systems.

Certain space missions performed by space robots are formulated in task space, such as Cartesian space (Jin RY et al., 2021). One of the primary challenges in end-effector position control for micro-nano FFSRs lies in mapping the desired end-effector position to the corresponding joint angles. Unlike fixed-base manipulators, the end-effector position of FFSRs is influenced by both the base satellite and the manipulator. Additionally, FFSRs are subject to non-holonomic constraints due to the unactuated nature of the base satellite (Liu et al., 2025). As a result, the relationship between the end-effector position and the joint angles in FFSRs is governed by velocity-level constraints. The generalized Jacobian matrix describing this relationship depends not only on the joint angles but also on the state of the base satellite, together with the mass and inertia properties of the constituent components. Traditional end-effector trajectory planning methods for FFSRs rely on accurate models (Rybus et al., 2022), yet such models are difficult to obtain in on-orbit scenarios because of unmodeled dynamics and base–manipulator coupling effects. With the advancement of artificial intelligence (AI), some researchers have addressed this issue using deep reinforcement learning methods (Lei et al.,

✉ Tao MENG, mengtao@zju.edu.cn
Renhao MAO, https://orcid.org/0009-0005-4218-9731
Tao MENG, https://orcid.org/0000-0001-7871-0457
CLC number: TP242; V448.2

Received: Feb. 25, 2026; Revision accepted: Apr. 20, 2026;
Crosschecked: Apr. 30, 2026; Published online: June 10, 2026

2023; Al Ali et al., 2024). However, the direct application of deep reinforcement learning is hindered by the limited interpretability of its decision-making processes (Shao YC et al., 2024), resulting in black-box controllers that may compromise stability analysis in safety-critical space missions.

Another challenge lies in designing a high-performance tracking controller at the dynamic level (Huang et al., 2024; Tang et al., 2025). Significant research efforts have been devoted to addressing the control challenges of FFSRs. A variety of control strategies have been proposed, including sliding mode control (Lavín-Delgado et al., 2023) and adaptive control (Prakash et al., 2022), aiming to achieve precise and robust control performance in the presence of model uncertainties and external disturbances. To meet the demand for rapid response in space missions, Tao et al. (2025) proposed a predefined-time control scheme, which significantly reduces the convergence time. Additionally, to ensure consistent performance across transient and steady-state phases, Gong (2025) developed a prescribed performance control scheme. While these methods effectively address diverse control challenges, their inherent reliance on accurate physical models requires substantial domain-specific expertise for model construction and entails rigorous parameter identification procedures. Moreover, the resulting nonlinear models complicate controller optimization and frequently lead to suboptimal energy efficiency. For micro-nano space robotic systems, computationally efficient and energy-efficient control strategies are essential for sustainable operation.

The Koopman operator (Koopman, 1931) has emerged as a powerful tool for extracting globally coherent linear representations from nonlinear systems. Its core principle lies in transforming nonlinear dynamic systems into linear representations within an infinite-dimensional space. Unlike traditional data-driven approaches, the global linearization property of the Koopman operator simplifies the design of subsequent controllers. However, since infinite-dimensional spaces are impractical for applications, finite-dimensional lifted observables are employed to approximate the system effectively. Historically, researchers have used the Koopman operator for control system design, leveraging radial basis functions or polynomial functions to approximate nonlinear models in finite dimensions (Korda and Mezić, 2018). Nevertheless, the manual selection of suitable lifted observables for complex nonlinear systems remains a significant challenge. With advancements in AI, the use of trained deep neural networks (DNNs) to derive observable functions has supplanted traditional manual selection and has become the predominant approach (Jin A et al., 2024). Building on the linearized models derived from the Koopman operator, linear-system control techniques, such as the linear quadratic regulator and model predictive control (MPC), have been developed and successfully applied to various nonlinear dynamical systems (Chen et al., 2024). Despite these advancements, the inherent approximation errors resulting from mapping low-dimensional nonlinear systems to high-dimensional linear models in the dynamic framework underscore the need for further enhancements in control precision.

To address control issues arising from inevitable modeling errors in deep Koopman (DK) models, as explored by Rahmani and Redkar (2024a, 2024b), fractional-order sliding mode control has been introduced. By exploiting the memory effect of fractional-order operators, this method improves control robustness, enhances tracking accuracy, and mitigates the impact of uncertainties in the DK model. However, it does not explicitly account for energy consumption optimization and suffers from slow tracking of time-varying trajectories. In our previous work, we developed a joint-space DK model for FFSRs and applied MPC to achieve optimized joint-angle control (Mao et al., 2024). However, generating future sequences of desired joint angles and angular velocities for end-effector position control in Cartesian space remains challenging because of the non-holonomic constraints of FFSRs. Consequently, directly applying MPC to the DK model in this scenario results in slower control responses and lower accuracy, since forecasting future trajectories over the prediction horizon is difficult.

In this paper, we collect input–output data from the system and train two DNNs separately. One DNN is used to map end-effector Cartesian coordinates to the corresponding joint angles and angular velocities, thereby generating the desired tracking signals. The other DNN, in combination with the Koopman operator, is employed to establish a high-dimensional linearized model in joint space. Together, these two networks form the foundation for controller design. Subsequently, to achieve high-precision and rapid tracking control while optimizing energy consumption, we introduce a terminal fractional-order (TFO) variable into the MPC framework. This approach leverages the memory effect of fractional-order variables and the rapid response of the terminal variables, while exploiting the receding-horizon optimization mechanism of MPC, thereby improving control performance without violating input constraints.

The main contributions of this paper are as follows:

1. Unlike the work of Bruder et al. (2021), which directly constructs an end-to-torque data-driven model, this paper proposes a more natural and structured two-layer data-driven modeling approach for micro-nano FFSRs. Based solely on input–output data from the FFSR, a kinematic model is learned using a DNN, while the dynamic model is constructed via the Koopman operator with the aid of another DNN. In this paper, the input–output trajectories are generated using a high-fidelity simulator for evaluation; however, the proposed learning-and-control pipeline itself assumes only access to measured trajectories and does not require explicit analytical model derivations during online control.

2. Building upon the memory property of the TFO variable, a TFO-based transformation is introduced to reformulate the error dynamics within the Koopman space. Compared with MPC applied to DK models (Chen et al., 2024) and fractional-order sliding mode control applied to DK models (Rahmani and Redkar, 2024a, 2024b), this method improves the overall closed-loop tracking performance.

3. Building on the DK-based linearization of the joint-space dynamic model, the TFO variable is integrated into the MPC framework for online control while respecting input constraints. This approach achieves lower energy consumption than DK-based sliding mode control, thereby contributing to more sustainable and energy-efficient control performance.

Notations used are as follows: For positive integers n and m, $\mathbf{0}_n \in \mathbb{R}^n$ denotes a vector whose entries are all

zero. $\mathbf{1}_n \in \mathbb{R}^n$ denotes a vector whose entries are all one. $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ denotes the $n \times n$ identity matrix. $\mathbb{R}^+$ denotes the set of positive real numbers. $\mathbb{N}$ denotes the set of natural numbers. The unit 3-sphere $\mathbb{S}^3 \subset \mathbb{R}^4$ is characterized as $\{\boldsymbol{q} \in \mathbb{R}^4 \mid \|\boldsymbol{q}\|_2 = 1\}$, corresponding to unit quaternions. For an arbitrary vector $\boldsymbol{a} \in \mathbb{R}^n$, a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and a positive-definite matrix $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, $\boldsymbol{a}^{\mathrm{T}}$ denotes the transpose of $\boldsymbol{a}$, $\boldsymbol{a}^{\dagger}$ denotes the Moore–Penrose pseudoinverse of $\boldsymbol{a}$, $\|\boldsymbol{a}\|_2^2 = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}$, $\|\boldsymbol{a}\|_{\boldsymbol{W}}^2 = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{a}$, and $\lceil \boldsymbol{a} \rfloor^{\lambda} = \left[|a_1|^{\lambda}\mathrm{sign}(a_1),\ |a_2|^{\lambda}\mathrm{sign}(a_2), \ldots,\ |a_n|^{\lambda}\mathrm{sign}(a_n)\right]^{\mathrm{T}}$. $\boldsymbol{a}(k+i|k)$ denotes the value of $\boldsymbol{a}$ at time step $k+i$ ($k \in \mathbb{N}$), predicted at time step k; if $i = 0$, $\boldsymbol{a}(k|k) = \boldsymbol{a}(k)$. For a square matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$, $\|\boldsymbol{B}\| = \sigma_{\max}(\boldsymbol{B})$ and $\|\boldsymbol{B}\|^j = (\sigma_{\max}(\boldsymbol{B}))^j$, where $\sigma_{\max}(\boldsymbol{B})$ denotes the largest singular value of $\boldsymbol{B}$.

2 Preliminaries

2.1 Koopman operator with control inputs

Consider a nonlinear dynamical system governed by the following discrete-time dynamics:

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ and $\boldsymbol{u}(k) \in \mathbb{R}^m$ represent the states and control inputs of the system at time step k, respectively. $\boldsymbol{f} : \mathbb{R}^{n+m} \to \mathbb{R}^n$ is a C^∞ smooth function.

The Koopman operator $\mathcal{K}$ is an infinite-dimensional linear operator that evolves the observable function $\boldsymbol{g}$ associated with the state-input pair:

$$\boldsymbol{g}(\boldsymbol{x}(k+1), \boldsymbol{u}(k+1)) = \mathcal{K}\boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{2}$$

where $\boldsymbol{g}$ is a C^∞ smooth function that maps the original state into an infinite-dimensional space.

Fig. 1 illustrates the action of the Koopman operator on a nonlinear system.

The observable function $\boldsymbol{g}$ can be further decomposed into components associated with the state and the control input:

$$\boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k)) = \begin{bmatrix} \boldsymbol{g}_{\mathrm{x}}(\boldsymbol{x}(k)) \\ \boldsymbol{u}(k) \end{bmatrix}, \tag{3}$$

where $\boldsymbol{g}_{\mathrm{x}}$ represents the lifted observable function.

Because the infinite-dimensional $\mathcal{K}$ is intractable in practice, it is approximated by a finite-dimensional truncation.

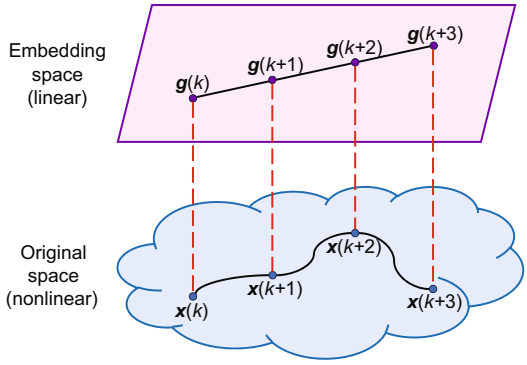

Fig. 1 Action of the Koopman operator on a nonlinear system

Accordingly, Eq. (3) can be written in the following finite-dimensional form:

$$\begin{bmatrix} \boldsymbol{z}(k+1) \\ \boldsymbol{u}(k+1) \end{bmatrix} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{z}(k) \\ \boldsymbol{u}(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_{\mathrm{xx}} & \boldsymbol{K}_{\mathrm{xu}} \\ \boldsymbol{K}_{\mathrm{ux}} & \boldsymbol{K}_{\mathrm{uu}} \end{bmatrix}\begin{bmatrix} \boldsymbol{z}(k) \\ \boldsymbol{u}(k) \end{bmatrix}, \tag{4}$$

where $\boldsymbol{z}(k) \in \mathbb{R}^d$ is a finite-dimensional approximation of the lifted observable $\boldsymbol{g}_{\mathrm{x}}$ and d is the Koopman dimension. $\boldsymbol{K} \in \mathbb{R}^{(d+m)\times(d+m)}$ is the finite-dimensional approximation of $\mathcal{K}$. $\boldsymbol{K}_{\mathrm{xx}} \in \mathbb{R}^{d\times d}$, $\boldsymbol{K}_{\mathrm{xu}} \in \mathbb{R}^{d\times m}$, $\boldsymbol{K}_{\mathrm{ux}} \in \mathbb{R}^{m\times d}$, and $\boldsymbol{K}_{\mathrm{uu}} \in \mathbb{R}^{m\times m}$ are block matrices of $\boldsymbol{K}$.

Following Eq. (4), let $\boldsymbol{K}_{\mathrm{xx}} = \boldsymbol{A}$ and $\boldsymbol{K}_{\mathrm{xu}} = \boldsymbol{B}$. Then the evolution of the system in the lifted space is described as follows:

$$\boldsymbol{z}(k+1) = \boldsymbol{A}\boldsymbol{z}(k) + \boldsymbol{B}\boldsymbol{u}(k). \tag{5}$$

To reconstruct the original state from the lifted space, we define $\boldsymbol{z}(k)$ as follows:

$$\boldsymbol{z}(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{\Phi}(\boldsymbol{x}(k)) \end{bmatrix}, \tag{6}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}(k)) = \left[\varphi_1(\boldsymbol{x}(k)), \varphi_2(\boldsymbol{x}(k)), \ldots, \varphi_{d-n}(\boldsymbol{x}(k))\right]^{\mathrm{T}} \in \mathbb{R}^{d-n}$ is the basis function vector of the lifted space. Consequently, the original state space is embedded in the lifted space, allowing the state to be reconstructed exactly by projection from $\boldsymbol{z}(k)$.

2.2 Grünwald–Letnikov fractional-order operator

In this work, we adopt the Grünwald–Letnikov definition of the fractional-order operator because our control scheme is implemented in discrete time and solved in a receding-horizon framework. The Grünwald–Letnikov operator admits a native discrete-time difference form, which can be evaluated directly from sampled signals and is therefore convenient for real-time implementation in an MPC setting. In contrast, the Caputo and Riemann–Liouville derivatives (Li et al., 2011) are more commonly formulated in continuous time. However, using them in a sampled-data predictive controller typically requires an additional discretization step for the fractional derivatives, thereby increasing implementation complexity.

The Grünwald–Letnikov fractional-order difference for a system y with an arbitrary order $\alpha \in \mathbb{R}$ is defined as follows:

$$\mathcal{D}^{\alpha} y(k) = \frac{1}{(\Delta T)^{\alpha}} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} y(k-j), \tag{7}$$

where ΔT denotes the sampling period, and $\binom{\alpha}{j}$ denotes the binomial coefficient, given by

$$\binom{\alpha}{j} = \begin{cases} 1, & j = 0, \\ \frac{\alpha(\alpha-1)\cdots(\alpha-j+1)}{j!}, & j = 1, 2, \ldots, k. \end{cases} \tag{8}$$

Since storing and processing the entire history of the sampled data is impractical in real-world applications, a finite memory length L is introduced into the definition of the fractional-order difference, yielding

$$\mathcal{D}^{\alpha} y(k) = \frac{1}{(\Delta T)^{\alpha}} \sum_{j=0}^{L} (-1)^j \binom{\alpha}{j} y(k-j). \tag{9}$$

3 Problem formulation

This paper considers a space robot consisting of a base satellite and an n-degree-of-freedom (DOF) manipulator, as shown in Fig. 2. The symbols and variables used to describe the space robot are defined in Table 1.

The kinematic relationship between the Cartesian space and joint space of the space robot can be expressed by the Jacobian matrix as follows:

$$\dot{\boldsymbol{X}}_{\mathrm{E}} = \boldsymbol{J}_{\mathrm{be}}\dot{\boldsymbol{X}}_{\mathrm{b}} + \boldsymbol{J}_{\mathrm{me}}\dot{\boldsymbol{\theta}}, \tag{10}$$

where $\dot{\boldsymbol{X}}_{\mathrm{E}} = \left[\boldsymbol{v}_{\mathrm{E}}^{\mathrm{T}},\ \boldsymbol{\omega}_{\mathrm{E}}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^6$ is the velocity vector of the end-effector, including the linear velocity $\boldsymbol{v}_{\mathrm{E}} \in \mathbb{R}^3$ and the angular velocity $\boldsymbol{\omega}_{\mathrm{E}} \in \mathbb{R}^3$. $\dot{\boldsymbol{X}}_{\mathrm{b}} = \left[\boldsymbol{v}_0^{\mathrm{T}},\ \boldsymbol{\omega}_0^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^6$ is the velocity vector of the base satellite, including the linear velocity $\boldsymbol{v}_0 \in \mathbb{R}^3$ and the angular velocity $\boldsymbol{\omega}_0 \in \mathbb{R}^3$. $\boldsymbol{\theta} = \left[\theta_1, \theta_2, \ldots, \theta_n\right]^{\mathrm{T}} \in \mathbb{R}^n$ is the vector of manipulator joint angles, $\theta_1, \theta_2, \ldots, \theta_n$ denote the angles of the individual joints, and $\dot{\boldsymbol{\theta}}$ is the joint angular velocities. $\boldsymbol{J}_{\mathrm{be}} \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{J}_{\mathrm{me}} \in \mathbb{R}^{6\times n}$ are the Jacobian matrices for the base satellite and the manipulator, respectively.

For an FFSR, the control forces and moments exerted on the base satellite and the external forces and moments exerted on the end-effector are zero. Consequently, the FFSR system obeys the principle of momentum conservation. Without loss of generality, we assume that the initial linear and angular momenta of the system are zero. The system momentum can be expressed as follows (Dou and Yue, 2023):

$$\boldsymbol{L} = \boldsymbol{H}_{\mathrm{b}}\dot{\boldsymbol{X}}_{\mathrm{b}} + \boldsymbol{H}_{\mathrm{bm}}\dot{\boldsymbol{\theta}} = \boldsymbol{0}_6, \tag{11}$$

where $\boldsymbol{H}_{\mathrm{b}} \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{H}_{\mathrm{bm}} \in \mathbb{R}^{6\times n}$ are the base satellite inertia matrix and the coupling inertia matrix between the base satellite and the manipulator, respectively.

Combining Eqs. (10) and (11), we obtain

$$\dot{\boldsymbol{X}}_{\mathrm{E}} = \boldsymbol{J}_{\mathrm{g}}\dot{\boldsymbol{\theta}}, \tag{12}$$

where $\boldsymbol{J}_{\mathrm{g}} = \boldsymbol{J}_{\mathrm{me}} - \boldsymbol{J}_{\mathrm{be}}\boldsymbol{H}_{\mathrm{b}}^{-1}\boldsymbol{H}_{\mathrm{bm}}$ is known as the generalized Jacobian matrix. The joint-space dynamics of the FFSR can be expressed as follows (Nekoo, 2022):

$$\begin{bmatrix} \boldsymbol{H}_{\mathrm{b}} & \boldsymbol{H}_{\mathrm{bm}} \\ \boldsymbol{H}_{\mathrm{bm}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{m}} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{X}}_{\mathrm{b}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_{\mathrm{b}} \\ \boldsymbol{c}_{\mathrm{m}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_6 \\ \boldsymbol{u} \end{bmatrix}, \tag{13}$$

where $\boldsymbol{H}_{\mathrm{m}} \in \mathbb{R}^{n\times n}$ represents the manipulator inertia matrix and $\boldsymbol{c}_{\mathrm{b}} \in \mathbb{R}^6$ and $\boldsymbol{c}_{\mathrm{m}} \in \mathbb{R}^n$ encapsulate the nonlinear components. $\boldsymbol{u} \in \mathbb{R}^n$ corresponds to the control torque applied to the manipulator.

In practice, FFSRs are influenced by external disturbances. To account for these effects, the dynamic model can be modified as

$$\begin{bmatrix} \boldsymbol{H}_{\mathrm{b}} & \boldsymbol{H}_{\mathrm{bm}} \\ \boldsymbol{H}_{\mathrm{bm}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{m}} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{X}}_{\mathrm{b}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_{\mathrm{b}} \\ \boldsymbol{c}_{\mathrm{m}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_6 \\ \boldsymbol{u} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Delta}_{\mathrm{b}} \\ \boldsymbol{\Delta}_{\mathrm{m}} \end{bmatrix}, \tag{14}$$

where $\boldsymbol{\Delta}_{\mathrm{b}} \in \mathbb{R}^6$ and $\boldsymbol{\Delta}_{\mathrm{m}} \in \mathbb{R}^n$ represent the disturbance vectors acting on the base satellite and the manipulator, respectively.

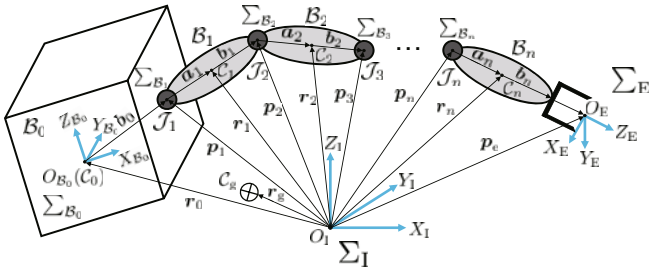

Fig. 2 Model of an FFSR. $O_{\mathcal{B}_0}/O_{\mathrm{I}}/O_{\mathrm{E}}$: origin of the corresponding coordinate frame $\sum_{\mathcal{B}_0}/\sum_{\mathrm{I}}/\sum_{\mathrm{E}}$

Table 1 FFSR symbols and variables

Symbol	Definition
$\mathcal{B}_0$	Base satellite
$\mathcal{B}_i$	Link i
$\mathcal{J}_i$	Joint i
$\mathcal{C}_0$	Center of mass (CM) of the base satellite
$\mathcal{C}_i$	CM of $\mathcal{B}_i$
$\mathcal{C}_{\mathrm{g}}$	CM of the system
$\sum_{\mathrm{I}}$	Inertial coordinate frame
$\sum_{\mathcal{B}_0}$	Base satellite coordinate frame
$\sum_{\mathcal{B}_i}$	$\mathcal{J}_i$ coordinate frame
$\sum_{\mathrm{E}}$	End-effector coordinate frame
$\boldsymbol{r}_{\mathrm{g}} \in \mathbb{R}^3$	Position vector of the space robot
$\boldsymbol{r}_0 \in \mathbb{R}^3$	Position vector of the base satellite
$\boldsymbol{r}_i \in \mathbb{R}^3$	Position vector of $\mathcal{B}_i$
$\boldsymbol{p}_i \in \mathbb{R}^3$	Position vector of $\mathcal{J}_i$
$\boldsymbol{p}_{\mathrm{e}} \in \mathbb{R}^3$	Position vector of the end-effector
$m_0 \in \mathbb{R}^1$	Mass of the base satellite
$m_i \in \mathbb{R}^1$	Mass of $\mathcal{B}_i$
$\boldsymbol{I}_0 \in \mathbb{R}^{3\times 3}$	Inertia matrix of the base satellite
$\boldsymbol{I}_i \in \mathbb{R}^{3\times 3}$	Inertia matrix of $\mathcal{B}_i$
$\boldsymbol{a}_i \in \mathbb{R}^3$	Length from $\mathcal{J}_i$ to $\mathcal{C}_i$
$\boldsymbol{b}_{i-1} \in \mathbb{R}^3$	Length from $\mathcal{C}_{i-1}$ to $\mathcal{J}_i$
$\boldsymbol{b}_n \in \mathbb{R}^3$	Length from $\mathcal{C}_n$ to the end-effector

$i = 1, 2, \ldots, n$

4 Control-oriented model learning

This section is divided into two parts and focuses on constructing models using only the system's input–output data as a basis for subsequent controller design. First, a DNN (referred to as DEN) is trained to capture the relationship between the end-effector position and the joint angles of the FFSR. Then, a high-dimensional linearized model is established in the joint space of the FFSR based on the Koopman operator, supported by another DNN (referred to as DKN).

In the aerospace field, it is common and practically reasonable to use high-fidelity simulation as the first step for generating training data and evaluating learning-based modeling methods (Shenwai et al., 2025). Real on-orbit experiments and ground-based hardware-in-the-loop tests are often constrained by cost, safety considerations, and experimental conditions, making it difficult to obtain rich datasets at the early stage. Therefore, this study begins with a high-fidelity MATLAB simulator to provide consistent input–output trajectories for model learning, while noting that the proposed framework relies only on measured input–output data and can be extended to hardware-generated data once such data become available.

In this work, the datasets are created from multiple trajectories, each spanning 5000 time steps (50 s). During data acquisition, the desired joint angular velocities are randomly assigned within predefined limits: $[-1, 1]$ rad/s for the first

three joints and $[-0.5, 0.5]$ rad/s for the last three joints. A proportional-like controller generates the control inputs, enabling random motion of the FFSR. A total of 20 simulations are performed to generate training trajectories for the FFSR. The resulting datasets are used jointly to train the DEN and DKN models.

4.1 Training of the DEN model

Unlike fixed-base manipulators, where the mapping between end-effector position and joint angles can be established directly, FFSRs are affected by nonholonomic constraints, requiring additional transformations to establish the relationship between the end-effector position and joint angles.

Based on Eq. (12), the relationship between the end-effector position and joint angles is characterized at the velocity level. Accordingly, we collect the corresponding data to train DEN: joint angular velocities $\dot{\boldsymbol{\theta}}$, end-effector position $\boldsymbol{p}_{\mathrm{e}}$, joint angles $\boldsymbol{\theta}$, and the attitude of the base satellite. We represent the attitude of the base satellite using quaternions $\boldsymbol{Q} \in \mathbb{S}^3$.

Fig. 3 illustrates the training process. The structure of the DEN model with fully connected layers used in this paper is outlined below:

$$\begin{cases} \boldsymbol{y}_1^*(k) = \boldsymbol{\sigma}_1^*\left(\boldsymbol{W}_1^*\begin{bmatrix}\boldsymbol{Q}(k)\\ \boldsymbol{\theta}(k)\end{bmatrix} + \boldsymbol{b}_1^*\right), \\ \boldsymbol{y}_2^*(k) = \boldsymbol{\sigma}_2^*\left(\boldsymbol{W}_2^*\boldsymbol{y}_1^*(k) + \boldsymbol{b}_2^*\right), \\ \vdots \\ \boldsymbol{y}_l^*(k) = \boldsymbol{\sigma}_l^*\left(\boldsymbol{W}_l^*\boldsymbol{y}_{l-1}^*(k) + \boldsymbol{b}_l^*\right), \\ \boldsymbol{\Psi}(k) = \boldsymbol{W}_{l+1}^*\boldsymbol{y}_l^*(k) + \boldsymbol{b}_{l+1}^*, \end{cases} \tag{15}$$

where $l \in \mathbb{N}$ denotes the number of hidden layers within the DEN architecture. $\{\boldsymbol{\sigma}_i^*\}_{i=1}^{l}$, $\{\boldsymbol{W}_i^*\}_{i=1}^{l+1}$, and $\{\boldsymbol{b}_i^*\}_{i=1}^{l+1}$ denote the activation functions, weights, and biases associated with DEN, respectively. $\{\boldsymbol{y}_i^*(k)\}_{i=1}^{l}$ denotes the outputs corresponding to each transformation stage. $\boldsymbol{\Psi}(k) \in \mathbb{R}^{3n}$ is the output of DEN, which is then reshaped into $\boldsymbol{\Psi}(k) \in \mathbb{R}^{3\times n}$.

The training objective of DEN is to minimize the following loss function:

$$\mathcal{L}^* = \left\| \boldsymbol{p}_{\mathrm{e}}(k+1) - \left(\boldsymbol{p}_{\mathrm{e}}(k) + \boldsymbol{\Psi}(k)\dot{\boldsymbol{\theta}}(k)\Delta T\right)\right\|_2^2. \tag{16}$$

4.2 Training of the DKN model

To train the DKN model of the joint-space dynamics, it is essential to collect data that include joint angles $\boldsymbol{\theta}$, joint angular velocities $\dot{\boldsymbol{\theta}}$, and random control inputs $\boldsymbol{u}$.

Fig. 4 illustrates the training process. Matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are implemented as bias-free linear transformations within the network architecture. Let $\boldsymbol{x} = \left[\boldsymbol{\theta}^{\mathrm{T}}, \dot{\boldsymbol{\theta}}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{2n}$. The structure of DKN used to train the DK model with a residual network (ResNet) architecture is outlined below:

$$\begin{cases} \bar{\boldsymbol{y}}_0(k) = \bar{\boldsymbol{\sigma}}_0\bar{\boldsymbol{W}}_0\boldsymbol{x}(k), \\ \bar{\boldsymbol{y}}_1(k) = \bar{\boldsymbol{\sigma}}_2\bar{\boldsymbol{W}}_2\left(\bar{\boldsymbol{\sigma}}_1\bar{\boldsymbol{W}}_1\bar{\boldsymbol{y}}_0(k) + \bar{\boldsymbol{y}}_0(k)\right), \\ \vdots \\ \bar{\boldsymbol{y}}_{\bar{l}}(k) = \bar{\boldsymbol{\sigma}}_{2\bar{l}}\bar{\boldsymbol{W}}_{2\bar{l}}\left(\bar{\boldsymbol{\sigma}}_{2\bar{l}-1}\bar{\boldsymbol{W}}_{2\bar{l}-1}\bar{\boldsymbol{y}}_{\bar{l}-1}(k) + \bar{\boldsymbol{y}}_{\bar{l}-1}(k)\right), \\ \boldsymbol{\Phi}(k) = \bar{\boldsymbol{W}}_{2\bar{l}+1}\bar{\boldsymbol{y}}_{\bar{l}}(k), \end{cases} \tag{17}$$

where $\bar{l} \in \mathbb{N}$ represents the number of residual blocks in the DKN structure. The weights and activation functions for DKN are denoted by $\{\bar{\boldsymbol{W}}_i\}_{i=0}^{2\bar{l}+1}$ and $\{\bar{\boldsymbol{\sigma}}_i\}_{i=0}^{2\bar{l}}$, respectively. Additionally, $\{\bar{\boldsymbol{y}}_i(k)\}_{i=0}^{\bar{l}}$ denotes the intermediate outputs of the residual blocks.

Suppose that the system's state at time step k is described by $\boldsymbol{x}(k)$, where $\boldsymbol{u}(k)$ represents the control inputs and $\boldsymbol{z}(k)$ corresponds to the lifted-space state. The state prediction in the lifted space after $S \in \mathbb{N}$ time steps can be expressed as

$$\hat{\boldsymbol{z}}(k+S) = \boldsymbol{A}^S\boldsymbol{z}(k) + \sum_{i=1}^{S}\boldsymbol{A}^{S-i}\boldsymbol{B}\boldsymbol{u}(k+i-1). \tag{18}$$

The loss function selected in this paper mainly consists of the prediction error in the lifted space and the prediction error of the original system states:

$$\begin{aligned} \bar{\mathcal{L}} =& \bar{\alpha}_1\sum_{i=1}^{S}\|\boldsymbol{z}(k+i) - \hat{\boldsymbol{z}}(k+i)\|_2^2 \\ &+ \bar{\alpha}_2\sum_{i=1}^{S}\|\boldsymbol{x}(k+i) - \boldsymbol{C}\hat{\boldsymbol{z}}(k+i)\|_2^2, \end{aligned} \tag{19}$$

where $\boldsymbol{C} = \left[\boldsymbol{I}_{2n}^{\mathrm{T}}, \boldsymbol{0}_{2n\times(d-2n)}^{\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{2n\times d}$, $\bar{\alpha}_1 \in \mathbb{R}^+$, and $\bar{\alpha}_2 \in \mathbb{R}^+$ are trade-off parameters corresponding to different parts of the loss function.

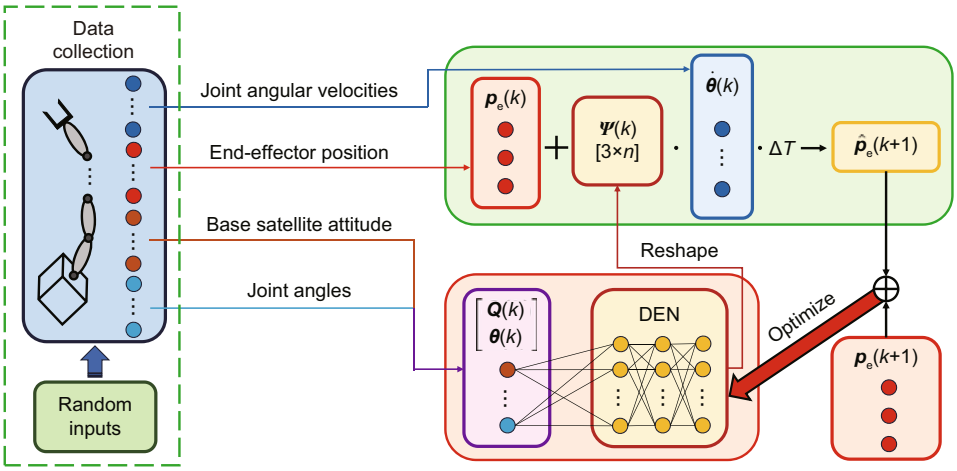

Fig. 3 Process of training DEN for the DNN-based end-effector position model. ⊕: the difference between the true value and the predicted value

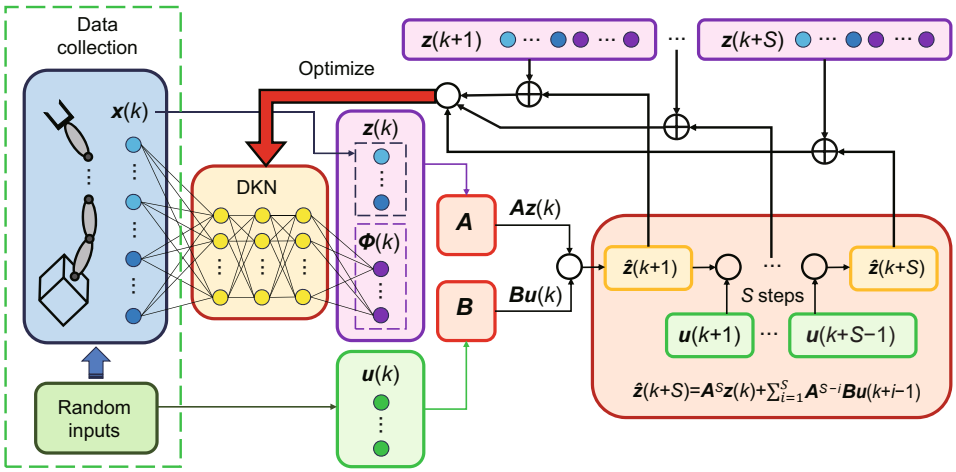

Fig. 4 Process of training DKN for the DK model. ◯: the additive combination of incoming signal components

After completing the training of DKN, a linearized representation of the FFSR dynamics in the lifted space can be derived as follows:

$$\begin{cases} \boldsymbol{z}(k+1) = \boldsymbol{A}\boldsymbol{z}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{\delta}(k) + \boldsymbol{d}(k) \\ \qquad\quad = \boldsymbol{A}\boldsymbol{z}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{D}(k), \\ \boldsymbol{x}(k) = \boldsymbol{C}\boldsymbol{z}(k), \end{cases} \tag{20}$$

where $\boldsymbol{d}(k) \in \mathbb{R}^d$ represents external disturbances in the lifted space, mapped from the original dynamical system, while $\boldsymbol{\delta}(k) \in \mathbb{R}^d$ captures the approximation errors introduced when reducing the original nonlinear system to a finite-dimensional linear representation during the training of DKN. $\boldsymbol{D}(k) \in \mathbb{R}^d$ combines these effects, representing the aggregated disturbances.

The controller design is based on the following nominal system model:

$$\begin{cases} \hat{\boldsymbol{z}}(k+1) = \boldsymbol{A}\boldsymbol{z}(k) + \boldsymbol{B}\boldsymbol{u}(k), \\ \boldsymbol{x}(k) = \boldsymbol{C}\boldsymbol{z}(k), \end{cases} \tag{21}$$

where $\hat{\boldsymbol{z}}(k+1)$ denotes the predicted approximation of $\boldsymbol{z}(k+1)$.

Assumption 1 (Mao et al., 2024) For the system described in Eq. (20), the lumped disturbance $\boldsymbol{D}(k)$ is assumed to be bounded.

5 Controller design

This section presents the controller design based on the model derived in Section 4. First, DEN is employed to generate the desired states in the joint space, given the expected end-effector position in Cartesian space and the current relevant states. Subsequently, a TFO variable is introduced to transform the error variable. Finally, this variable is incorporated into the MPC framework to derive the controller.

The structure of the proposed method is illustrated in Fig. 5. First, input–output data are collected from the FFSR. Next, these datasets are employed to train the DEN model and the DKN model offline. After extensive training, during the online control phase, DEN is employed to map the desired end-effector position and the current relevant states to the desired state in the joint space. Then, the error variable is transformed into the TFO variable and incorporated into TFO-MPC for online control based on the linearized DK model learned via DKN. Before proceeding, the following assumption is introduced:

Assumption 2 The end-effector of the FFSR moves within a path-independent workspace, where the pseudoinverse of $\boldsymbol{\Psi}(k)$ always exists.

The workspace of the FFSR is generally divided into a path-independent workspace and a path-dependent workspace. The path-independent workspace refers to the region where, no matter what trajectory is chosen, no dynamic singularities occur during the kinematic solution (Papadopoulos et al., 2021). Therefore, the desired trajectory must be chosen within the path-independent workspace to avoid dynamic singularities that could result in control failure. This ensures that Assumption 5 is satisfied. Given that the desired trajectory for the end-effector position is $\boldsymbol{p}_{\mathrm{ref}}$ and satisfies Assumption 5, the desired joint angular velocities in the joint space, determined in conjunction with DEN, can be expressed as

$$\dot{\boldsymbol{\theta}}_{\mathrm{d}}(k) = k_{\mathrm{p}}\boldsymbol{\Psi}^{\dagger}(k)\left(\boldsymbol{p}_{\mathrm{ref}}(k) - \boldsymbol{p}_{\mathrm{e}}(k)\right), \tag{22}$$

where $\dot{\boldsymbol{\theta}}_{\mathrm{d}}(k) \in \mathbb{R}^n$ denotes the vector of desired joint angular velocities, while $k_{\mathrm{p}} \in \mathbb{R}^+$ denotes the gain parameter.

Thus, the desired state vector $\boldsymbol{x}_{\mathrm{d}}(k) \in \mathbb{R}^{2n}$ in the joint space is given by

$$\boldsymbol{x}_{\mathrm{d}}(k) = \begin{bmatrix} \boldsymbol{\theta}(k) + \dot{\boldsymbol{\theta}}_{\mathrm{d}}(k)\Delta T \\ \dot{\boldsymbol{\theta}}_{\mathrm{d}}(k) \end{bmatrix}. \tag{23}$$

In the joint space, the error variable $\boldsymbol{e}(k) \in \mathbb{R}^{2n}$ is defined as

$$\boldsymbol{e}(k) = \boldsymbol{x}_{\mathrm{d}}(k) - \boldsymbol{C}\boldsymbol{z}(k). \tag{24}$$

Next, a TFO variable $\boldsymbol{s}[\boldsymbol{z}(k)] \in \mathbb{R}^{2n}$ is introduced to transform the error variable, facilitating the subsequent controller design:

$$\boldsymbol{s}[\boldsymbol{z}(k)] = \boldsymbol{e}(k) + \boldsymbol{\xi}\mathcal{D}^{\alpha-1}\lceil \boldsymbol{e}(k-1)\rfloor^{\lambda}, \tag{25}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{2n\times 2n}$, $\alpha \in \mathbb{R}^+$, and $0 < \lambda < 1$ are coefficients to be designed.

Within the MPC framework, the stage cost and the terminal cost of the proposed TFO-MPC are defined as

$$\begin{aligned} L\{\boldsymbol{s}[\hat{\boldsymbol{z}}(k+i|k)], \boldsymbol{u}(k+i|k)\} =& \|\boldsymbol{s}[\hat{\boldsymbol{z}}(k+i|k)]\|^2_{\boldsymbol{Q}_z} \\ &+ \|\boldsymbol{u}(k+i|k)\|^2_{\boldsymbol{R}}, \end{aligned} \tag{26}$$

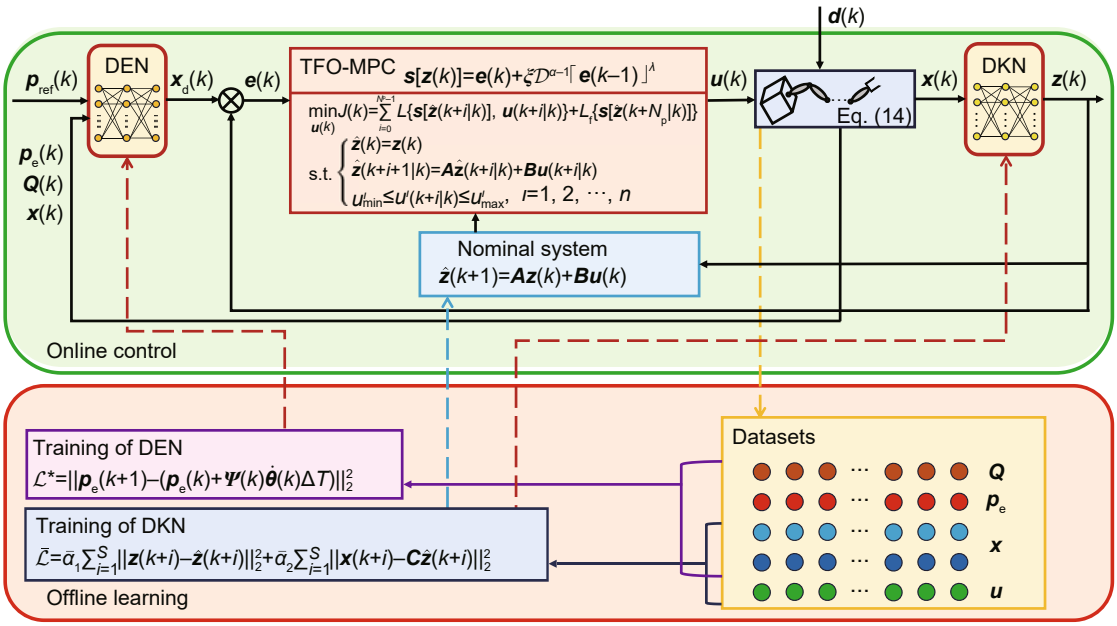

Fig. 5 Structure of the proposed method. ⊗: the tracking error between the desired state and the feedback state

$$L_{\mathrm{f}}\{\boldsymbol{s}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\}=\|\boldsymbol{s}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\|^2_{\boldsymbol{Q}_{\mathrm{N}}}, \tag{27}$$

where N_{p} denotes the prediction horizon. $\boldsymbol{Q}_{\mathrm{z}}\in\mathbb{R}^{2n\times 2n}$, $\boldsymbol{Q}_{\mathrm{N}}\in\mathbb{R}^{2n\times 2n}$, and $\boldsymbol{R}\in\mathbb{R}^{n\times n}$ denote the weighting matrices.

Considering the nominal system augmented with the TFO variable, along with the stage cost and terminal cost, the optimization problem of the TFO-MPC can be formulated as follows:

$$\begin{aligned}\min_{\boldsymbol{u}(k)} J(k)=&\sum_{i=0}^{N_{\mathrm{p}}-1} L\{\boldsymbol{s}[\hat{\boldsymbol{z}}(k+i|k)],\boldsymbol{u}(k+i|k)\}\\&+L_{\mathrm{f}}\{\boldsymbol{s}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\}\\ \text{s.t.}\ &\begin{cases}\hat{\boldsymbol{z}}(k)=\boldsymbol{z}(k),\\ \hat{\boldsymbol{z}}(k+i+1|k)=\boldsymbol{A}\hat{\boldsymbol{z}}(k+i|k)+\boldsymbol{B}\boldsymbol{u}(k+i|k),\\ u^{\iota}_{\min}\le u^{\iota}(k+i|k)\le u^{\iota}_{\max},\ \iota=1,2,\ldots,n,\end{cases}\end{aligned} \tag{28}$$

where $u^{\iota}_{\min}\in\mathbb{R}^{+}$ and $u^{\iota}_{\max}\in\mathbb{R}^{+}$ denote the lower and upper bounds of the control inputs for each joint, respectively, and $u^{\iota}\in\mathbb{R}^{+}$ $(\iota=1,2,\ldots,n)$ denotes the control input of the ι^{th} joint.

Theorem 1 The norm of the TFO prediction error in the optimization problem, arising from the discrepancy between the true system and the nominal system, remains bounded.

Theorem 2 For the true system under the proposed TFO-MPC strategy, the TFO variable $\boldsymbol{s}[\boldsymbol{z}(k)]$ in Eq. (25) is input-to-state stable.

Theorem 3 If the TFO variable $\boldsymbol{s}[\boldsymbol{z}(k)]$ in Eq. (25) converges, then the error variable $\boldsymbol{e}(k)$ in Eq. (24) converges within a finite number of steps.

Details of relevant proofs are provided in the supplementary materials.

6 Simulations

6.1 Simulation settings

This subsection evaluates the performance of the proposed approach using a 6-DOF micro-nano FFSR, as depicted in Fig. 6. The mass and inertia entries define the dynamic properties of the base satellite and each manipulator link, and $I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}$, and I_{yz} are the components of the inertia tensor expressed in the corresponding body-fixed frame. The entries DH.d, DH.α, DH.a, and DH.θ denote the standard Denavit–Hartenberg (DH) parameters used to describe the manipulator kinematic chain. The quantities b_i^x, b_i^y, and b_i^z are the components of $\boldsymbol{b}_i=[b_i^x,b_i^y,b_i^z]^{\mathrm{T}}$, which represents the vector from the corresponding center of mass to the adjacent joint, as defined in Table 1. The system comprises a 6-DOF manipulator mounted on a satellite base. The DH parameters, as well as the kinematic and dynamic properties, are summarized in Table 2. These parameters are used only to build high-fidelity

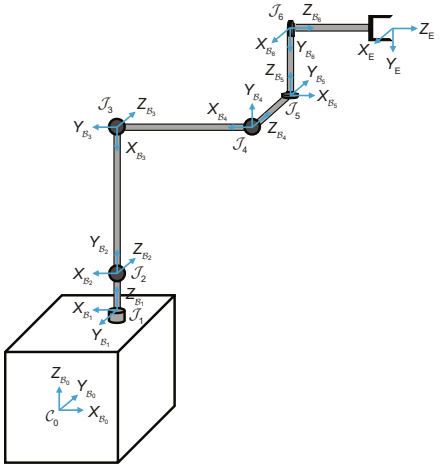

Fig. 6 Simulation model

Table 2 Parameters of the space robot

Parameter	$\mathcal{B}_0$	$\mathcal{J}_1$	$\mathcal{J}_2$	$\mathcal{J}_3$	$\mathcal{J}_4$	$\mathcal{J}_5$	$\mathcal{J}_6$
Mass (kg)	40	3.7	8.393	2.275	1.219	1.219	2.1879
Inertia I_{xx} (kg · m^2)	1.5	0.0067	0.0149	0.0025	0.0012	0.0012	0.04
Inertia I_{yy} (kg · m^2)	1	0.0064	0.3564	0.0551	0.0012	0.0012	0.04
Inertia I_{zz} (kg · m^2)	1.2	0.0067	0.3553	0.0546	0.0009	0.0009	0.01
Inertia I_{xy} (kg · m^2)	0	0	0	0	0	0	0
Inertia I_{xz} (kg · m^2)	0	0	0	0.0034	0	0	0
Inertia I_{yz} (kg · m^2)	0	0	0	0	0	0	0
DH.d (m)	–	0.089 159	0	0	0.093	0.094 65	0.2423
DH.α (rad)	–	$\pi/2$	0	0	$\pi/2$	$-\pi/2$	0
DH.a (m)	–	0	−0.425	−0.392 25	0	0	0
DH.θ (rad)	–	0	$-\pi/2$	$\pi/2$	π	$-\pi/2$	0
b_i^x (m)	0.15	0	0.2125	0.119 93	0	0	0
b_i^y (m)	0	−0.025 61	0	0	−0.0018	0.0018	0
b_i^z (m)	0.29	0.001 93	0.113 36	0.0265	0.016 34	0.016 34	−0.111 59

"–" indicates that the data are not available

simulators for evaluation. The inertial frame $\sum_{\mathrm{I}}$ is aligned with the initial base satellite coordinate frame $\sum_{\mathcal{B}_0}$. The joint torque limits are set to $u_{\min}^{\iota} = -5$ N·m and $u_{\max}^{\iota} = 5$ N·m. The sampling interval is fixed at $\Delta T = 0.01$ s. Offline training is implemented in Python 3.8.18, while the online control is executed in MATLAB R2021b on a workstation equipped with an Intel® Core™ i9-12900H@2.50 GHz processor and 16.0 GB of RAM. The hyperparameters selected for the simulations are as follows: For the DEN model (Eq. (15)), the number of hidden layers is $l = 3$, the activation functions $\{\boldsymbol{\sigma}_i^*\}_{i=1}^{l}$ are exponential linear units (ELUs), the number of neurons in each hidden layer is 256, the structure of DEN is $[10, 256, \ldots, 256, 18]$, and the learning rate is 10^{-3}; for the DKN model (Eqs. (17)–(20)), the number of residual blocks is $\bar{l} = 3$, the activation functions $\{\bar{\boldsymbol{\sigma}}_i\}_{i=0}^{2\bar{l}}$ are rectified linear units (ReLUs), the number of neurons in each hidden layer is 256, the Koopman dimension is $d = 30$, the structure of DKN is $[12, 256, \ldots, 256, 18]$, the multi-step prediction length and loss-function parameters are $S = 9$, $\bar{\alpha}_1 = 1$, and $\bar{\alpha}_2 = 1$, and the learning rate is 10^{-3}; for the controller design (Eqs. (22)–(28)), the gain parameter is $k_{\mathrm{p}} = 1$, the coefficients are $\boldsymbol{\xi} = \mathrm{diag}\{5\boldsymbol{I}_6, 0.05\boldsymbol{I}_6\}$, $\alpha = 0.5$, and $\lambda = 0.6$, the weights are $\boldsymbol{Q}_{\mathrm{z}} = \mathrm{diag}\{20\boldsymbol{I}_6, 0.2\boldsymbol{I}_6\}$, $\boldsymbol{Q}_{\mathrm{N}} = 10\boldsymbol{Q}_{\mathrm{z}}$, and $\boldsymbol{R} = \mathrm{diag}\{0.001\boldsymbol{I}_3, 0.01\boldsymbol{I}_2, 0.1\}$, and the prediction horizon is $N_{\mathrm{p}} = 9$.

6.2 Training and prediction evaluation

In this subsection, a two-layer data-driven model for the FFSR shown in Fig. 6 is developed using DNNs.

After the training is completed, we compare the learned data-driven models with the high-fidelity simulator. Fig. 7 presents the one-step prediction performance of the end-effector position in Cartesian space for the FFSR after training DEN, while Fig. 8 shows the ten-step prediction performance of the DK model in the joint space of FFSR after training with DKN. The proposed two-layer data-driven modeling framework exhibits favorable prediction performance in Cartesian space and joint space, providing a reliable basis for subsequent controller design.

Subsequently, we define the root mean square error

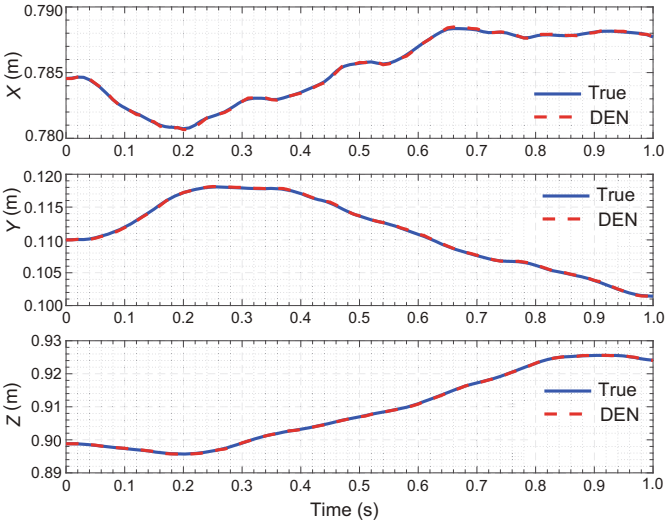

Fig. 7 Prediction performance of DEN

(RMSE) as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{k=1}^{N} \|\boldsymbol{\varepsilon}(k)\|_2^2}{N}}, \tag{29}$$

where N denotes the number of sampling points and $\boldsymbol{\varepsilon}(k)$ denotes the prediction error at time step k.

Tables 3 and 4 report the one-step prediction performance of $\boldsymbol{p}_{\mathrm{e}}$ for different numbers of DEN hidden layers l and the ten-step prediction performance of $\boldsymbol{x}$ for different Koopman dimensions d, respectively, as evaluated on the test set. For the DEN model, the one-step prediction RMSE generally decreases with increasing hidden layers, achieving its minimum at $l = 3$ in this work. A slight rebound in error is observed when the layer number is further increased to $l = 4$. This suggests that while greater depth enhances prediction accuracy within a certain range, a deeper architecture may lead to overfitting, resulting in a marginal performance degradation. For the DKN model, increasing the Koopman dimension yields only marginal improvements in prediction accuracy beyond $d = 30$ in this work. However, for the subsequent MPC controller, a higher dimension typically leads to an increased computational burden. Therefore, a practical trade-off should

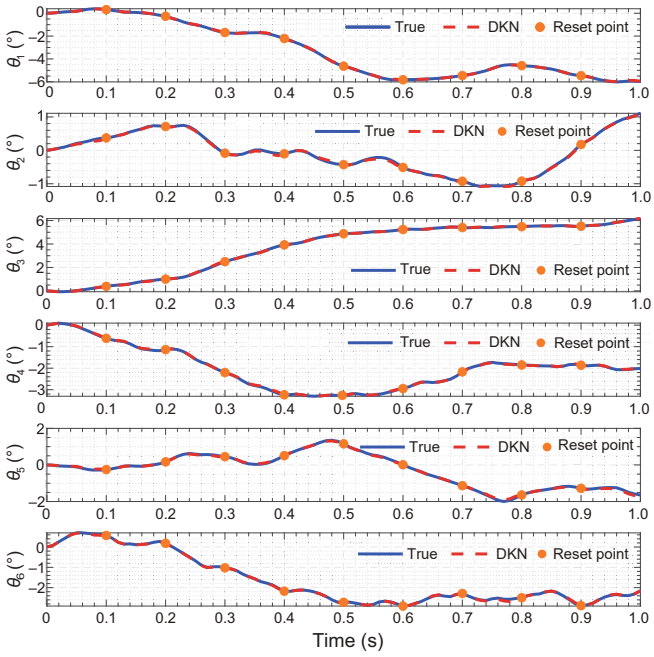

Fig. 8 Prediction performance of DKN

Table 3 One-step prediction RMSE of p_e under different l values

l	RMSE
1	9.87×10^{-5}
2	8.45×10^{-5}
3	6.65×10^{-5}
4	6.82×10^{-5}

Table 4 Ten-step prediction RMSE of x under different d values

d	RMSE
20	12.10×10^{-2}
30	8.74×10^{-2}
40	8.39×10^{-2}
50	8.31×10^{-2}

be made between prediction accuracy and computational cost.

6.3 Tracking performance evaluation (case 1)

In this subsection, we evaluate the performance of the proposed method in tracking a fixed desired position (case 1).

The target position of the end-effector is given by $\boldsymbol{p}_{\mathrm{ref}} = [0.5,\ 0,\ 0.5]^{\mathrm{T}}$ m. In addition, the base satellite and each manipulator component are subject to 20% uncertainties in mass and inertia properties. Moreover, each element of unknown disturbances $\Delta_m^i(t) = 0.01 \times (\sin(20t) + \cos(10t) - \mathrm{e}^{-0.1t})$ N·m $(i = 1, 2, \ldots, 6)$ is applied to each joint.

We compare the proposed method with the following approaches: a conventional MPC method based on the DK model (DK-MPC) (Chen et al., 2024), a fractional-order sliding mode control method based on the DK model (DK-FOSMC) (Rahmani and Redkar 2024a, 2024b), and a computed-torque fractional-order sliding mode control method based on the explicit model (Exp-CTFOSMC) (Shao XY et al., 2021).

The main distinction between the cost function of DK-MPC and the proposed method lies in the absence of the TFO-based transformation of the error variable. The cost function of DK-MPC is as follows:

$$\begin{aligned}\min_{\boldsymbol{u}(k)} J_{\mathrm{MPC}}(k) = &\sum_{i=0}^{N_{\mathrm{p}}-1} L\{\boldsymbol{e}[\hat{\boldsymbol{z}}(k+i|k)], \boldsymbol{u}(k+i|k)\} \\ &+ L_{\mathrm{f}}\{\boldsymbol{e}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\} \\ \text{s.t.} \begin{cases} \hat{\boldsymbol{z}}(k) = \boldsymbol{z}(k), \\ \hat{\boldsymbol{z}}(k+i+1|k) = \boldsymbol{A}\hat{\boldsymbol{z}}(k+i|k) + \boldsymbol{B}\boldsymbol{u}(k+i|k), \\ u_{\min}^{\iota} \le u^{\iota}(k+i|k) \le u_{\max}^{\iota},\ \iota = 1, 2, \ldots, 6, \end{cases}\end{aligned} \tag{30}$$

where $L\{\boldsymbol{e}[\hat{\boldsymbol{z}}(k+i|k)], \boldsymbol{u}(k+i|k)\} = \|\boldsymbol{e}[\hat{\boldsymbol{z}}(k+i|k)]\|_{\boldsymbol{Q}_{\mathrm{z}}}^2 + \|\boldsymbol{u}(k+i|k)\|_{\boldsymbol{R}}^2$ and $L_{\mathrm{f}}\{\boldsymbol{e}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\} = \|\boldsymbol{e}[\hat{\boldsymbol{z}}(k+N_{\mathrm{p}}|k)]\|_{\boldsymbol{Q}_{\mathrm{N}}}^2$.

DK-FOSMC employs a fractional-order sliding mode control approach, with its control law given by

$$\begin{aligned}\boldsymbol{u}(k) = &(\boldsymbol{C}\boldsymbol{B})^{\dagger} \Big[\boldsymbol{e}(k) + \boldsymbol{\xi}\mathcal{D}^{\alpha-1}\lceil \boldsymbol{e}(k)\rfloor^{\lambda} \\ &- (\boldsymbol{I}_{12} - \boldsymbol{K})\left(\boldsymbol{e}(k) + \boldsymbol{\xi}\mathcal{D}^{\alpha-1}\lceil \boldsymbol{e}(k-1)\rfloor^{\lambda}\right)\Big],\end{aligned} \tag{31}$$

where $\boldsymbol{K} = \mathrm{diag}\{200\boldsymbol{I}_6, 0.2\boldsymbol{I}_6\}$, with all other hyperparameters remaining identical to those in the proposed method.

Exp-CTFOSMC employs the fractional-order sliding mode term at the acceleration level as the virtual control input, which is then mapped to the control torque via a computed-torque scheme based on an explicit dynamic model. The virtual control input is given by

$$\begin{aligned}\boldsymbol{v}(k) = &\begin{bmatrix} 0.5\Delta T^2 \boldsymbol{I}_6 \\ \Delta T \boldsymbol{I}_6 \end{bmatrix}^{\dagger} \Bigg[\boldsymbol{K}_{\mathrm{v}}\boldsymbol{e}(k) - \begin{bmatrix} \Delta T \boldsymbol{I}_6 \\ \boldsymbol{0}_{6\times 6} \end{bmatrix} \dot{\boldsymbol{\theta}}(k) \\ &+ \boldsymbol{\xi}\mathcal{D}^{\alpha-1}\lceil \boldsymbol{e}(k)\rfloor^{\lambda} - (\boldsymbol{I}_{12} - \boldsymbol{K})\,\boldsymbol{\xi}\mathcal{D}^{\alpha-1}\lceil \boldsymbol{e}(k-1)\rfloor^{\lambda}\Bigg],\end{aligned} \tag{32}$$

where $\boldsymbol{K}_{\mathrm{v}} = \mathrm{diag}\{20000\boldsymbol{I}_6, 20\boldsymbol{I}_6\}$, with all other hyperparameters remaining identical to those in the proposed method. Subsequently, the control law is formulated as

$$\boldsymbol{u} = \left(\boldsymbol{H}_{\mathrm{m}} - \boldsymbol{H}_{\mathrm{bm}}^{\mathrm{T}}\boldsymbol{H}_{\mathrm{b}}^{-1}\boldsymbol{H}_{\mathrm{bm}}\right)\boldsymbol{v}(k) + \boldsymbol{c}_{\mathrm{m}} - \boldsymbol{H}_{\mathrm{bm}}^{\mathrm{T}}\boldsymbol{H}_{\mathrm{b}}^{-1}\boldsymbol{c}_{\mathrm{b}}. \tag{33}$$

To evaluate the performance of the four methods, we use three metrics: RMSE, integral of time-weighted square error (ITSE), and integral of squared control input (ISCI). These metrics are chosen to assess the control accuracy and energy efficiency. Specifically, RMSE and ITSE are employed to quantify the tracking error, while ISCI is used to reflect the control cost. RMSE represents the mean square magnitude of the error and thus primarily characterizes the average error level. In contrast, ITSE introduces time weighting, emphasizing the time-accumulated effect of the error and the suppression of late-stage residual errors. Consequently, it is more sensitive to the convergence rate and tailing behavior of the error trajectory.

The definitions of ITSE and ISCI are as follows:

$$\begin{cases} \mathrm{ITSE} = \displaystyle\sum_{k=1}^{T/\Delta T} T\|\boldsymbol{p}_{\mathrm{ref}} - \boldsymbol{p}_{\mathrm{e}}(k)\|_2^2, \\ \mathrm{ISCI} = \displaystyle\sum_{k=1}^{T/\Delta T} \|\boldsymbol{u}(k)\|_2^2. \end{cases} \tag{34}$$

Fig. 9 demonstrates that all four control methods effectively control the end-effector of the FFSR to reach the desired

position, thereby indirectly validating the accuracy of the two-layer data-driven modeling approach developed in the previous subsection. Fig. 10 presents the error curves over time for the four control methods in case 1. As shown in this figure, DK-MPC exhibits relatively slow tracking because it must follow the desired joint angles generated by DEN within each short prediction horizon. This phenomenon will be further demonstrated in the next subsection when tracking a time-varying trajectory. The proposed method improves tracking speed and accuracy by converting the error variable into a TFO variable. Fig. 11 shows the control torques over time for the four control methods in case 1. At the initial stage of control, DK-FOSMC and Exp-CTFOSMC exhibit relatively large control efforts. In contrast, DK-MPC and the proposed method optimize energy consumption, achieving better performance in terms of required control energy compared to DK-FOSMC and Exp-CTFOSMC.

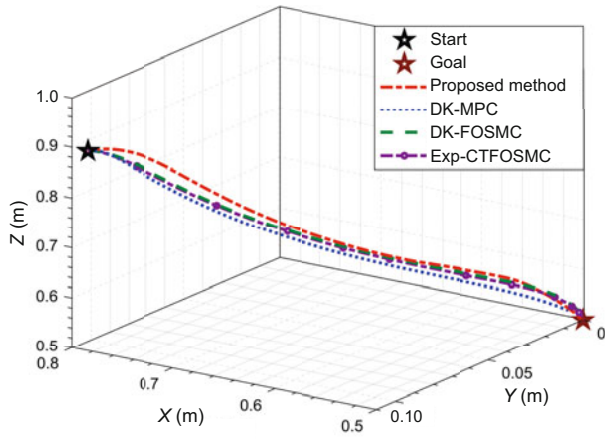

Fig. 9 Trajectories of the four methods (case 1)

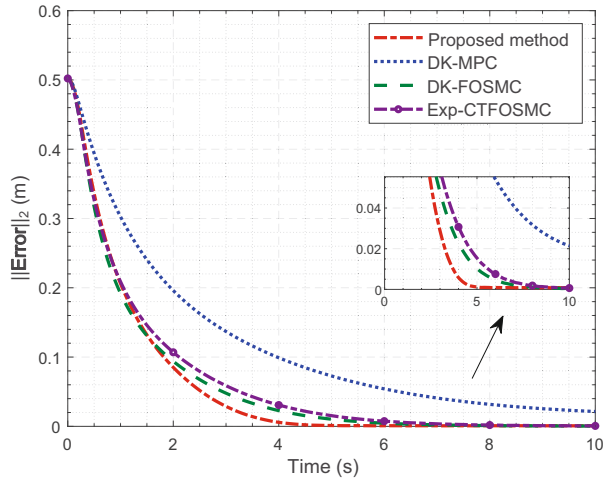

Fig. 10 Error curves of the four methods (case 1), where $\|\mathbf{Error}\|_2 = \sqrt{\|\mathbf{Error}\|_2^2}$

Table 5 presents a comparison of the four control methods using quantitative metrics. Although the DK-MPC method demonstrates the lowest ISCI, it suffers from the slowest response and lowest precision, resulting in the worst RMSE and ITSE metrics. In contrast, the proposed method, which

Table 5 Tracking performance comparison (case 1)

Method	RMSE	ITSE	ISCI
Proposed method	1.24×10^{-1}	$\mathbf{0.83 \times 10^{3}}$	1.49×10^{3}
DK-MPC	1.68×10^{-1}	3.60×10^{3}	$\mathbf{0.66 \times 10^{3}}$
DK-FOSMC	$\mathbf{1.22 \times 10^{-1}}$	0.86×10^{3}	1.83×10^{3}
Exp-CTFOSMC	1.27×10^{-1}	1.04×10^{3}	1.82×10^{3}

The best results are in bold

extends DK-MPC by incorporating TFO variables, sacrifices some ISCI in favor of achieving faster control speed and improved precision. As a result, the RMSE and ITSE metrics are better than those of DK-MPC. Although Exp-CTFOSMC is structurally similar to DK-FOSMC, it relies on an explicit model and is therefore more sensitive to parametric mismatches and unmodeled dynamics. Consequently, under mass and inertia uncertainties as well as external disturbances, DK-FOSMC achieves a lower RMSE and ITSE than Exp-CTFOSMC, which indirectly highlights the practical advantage of the proposed learning-based approach over traditional model-based methods. Since the proposed method is slower than DK-FOSMC in the initial stage, its RMSE is slightly higher than that of DK-FOSMC. However, as can be seen from Fig. 10, the proposed method converges faster compared to DK-FOSMC. In addition, the energy consumption of the proposed method is reduced by 18.13% and 18.58% compared to Exp-CTFOSMC and DK-FOSMC, respectively.

We quantify the attitude drift induced in the base satellite during manipulator motion using the integrated base satellite attitude disturbance $\sum_{k=1}^{T/\Delta T} \|\boldsymbol{\omega}_0(k)\|_2^2$ (Mao et al., 2024). Fig. 12 illustrates the effects of the four control methods on the attitude disturbance of the base satellite. Since the base satellite attitude disturbance is induced mainly by the dynamic coupling of the manipulator, smoother torque profiles generally excite lower base satellite angular velocities and thus lead to reduced attitude disturbance. Both DK-FOSMC and Exp-CTFOSMC produce relatively large control inputs in the initial phase; thus, their influence on the attitude is more pronounced, as evidenced by the larger peak variations in the base satellite attitude response. In contrast, the proposed method explicitly incorporates optimization of the control input, resulting in a smoother initial control profile and consequently smaller base satellite attitude drift. Although DK-MPC yields the smallest base satellite attitude drift, this comes at the expense of slower end-effector tracking dynamics and reduced tracking accuracy.

Fig. 13 further illustrates the role of the proposed TFO variable. Compared with DK-MPC, the proposed method replaces the conventional error variable $\boldsymbol{e}(k)$ in the cost function with the TFO variable $\boldsymbol{s}[\boldsymbol{z}(k)]$. During the initial transient, the memory property of $\boldsymbol{s}[\boldsymbol{z}(k)]$ accelerates the decay of the tracking error, thereby improving the response speed. In the steady-state regime, the fractional-order integral effect embedded in $\boldsymbol{s}[\boldsymbol{z}(k)]$ improves disturbance rejection, yielding lower residual tracking errors in the presence of lumped internal and external disturbances $\boldsymbol{D}(k)$.

Fig. 14 reports the online computation time for solving the proposed method's optimization problem, measured in MATLAB. Since DKN transforms the original system into a high-dimensional linear system, it substantially reduces the computational burden compared with solving a nonlinear

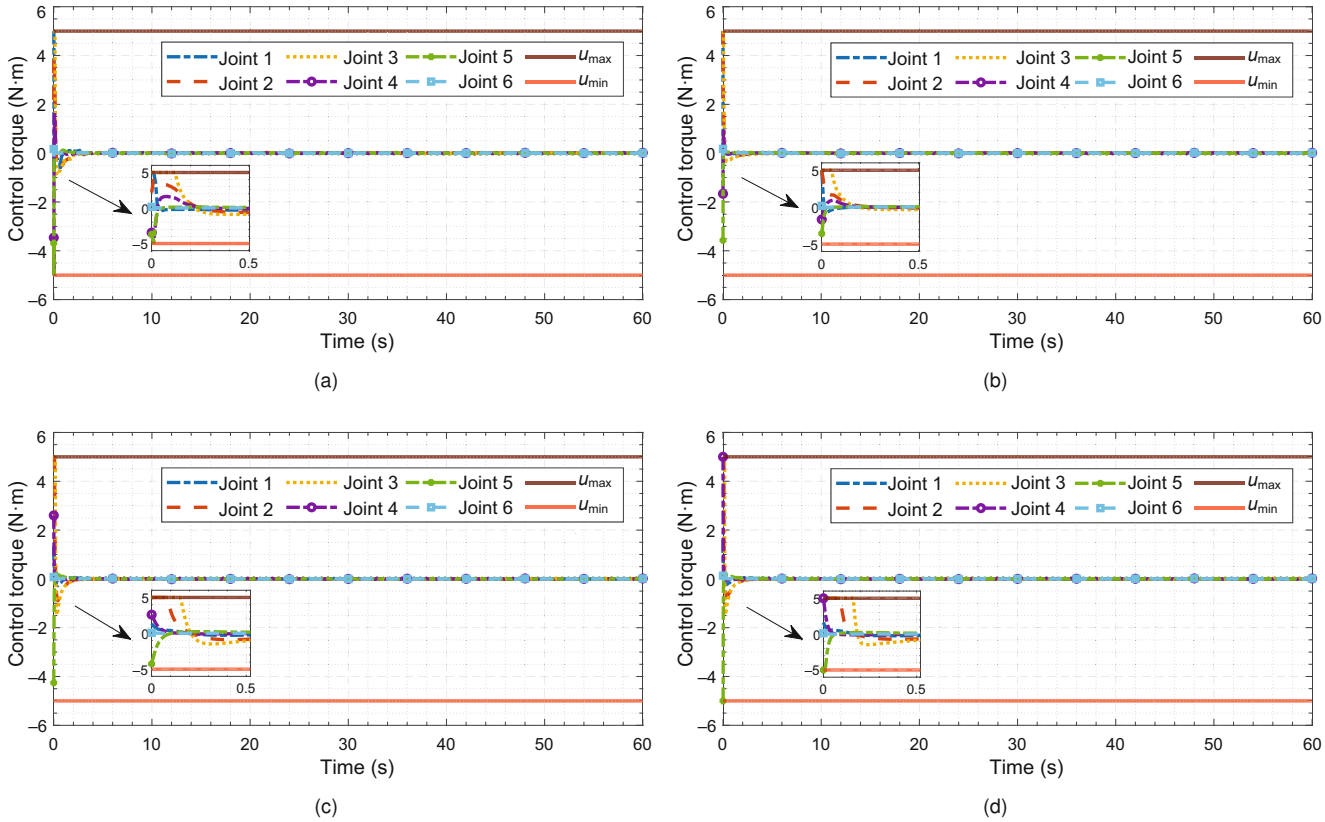

Fig. 11 Control torques of four methods (case 1): (a) proposed method; (b) DK-MPC; (c) DK-FOSMC; (d) Exp-CTFOSMC

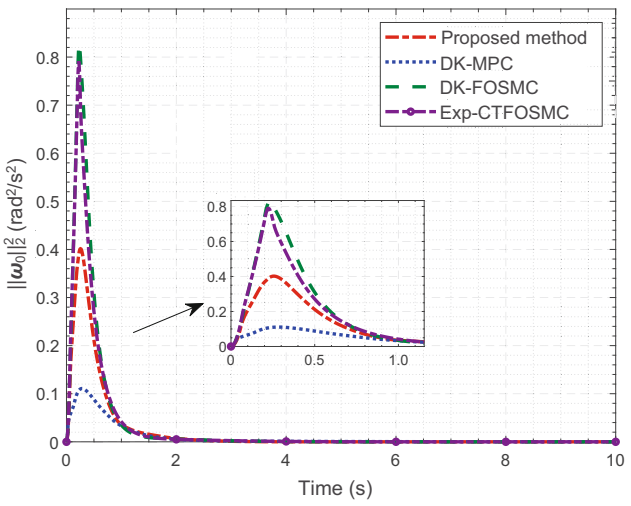

Fig. 12 Base satellite attitude disturbance

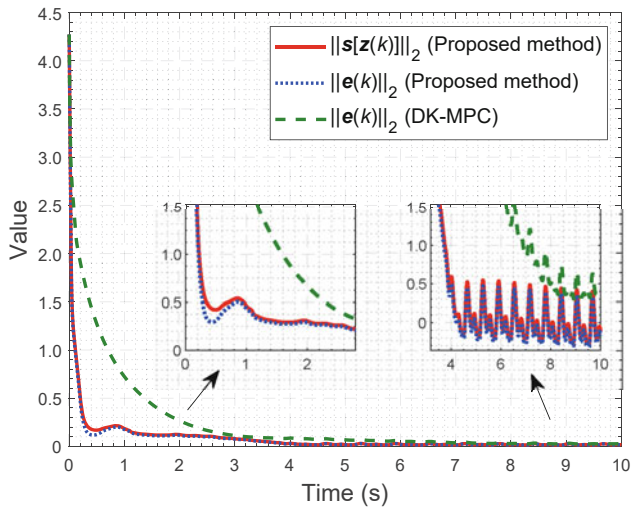

Fig. 13 Effect of the TFO variable compared with DK-MPC

optimization problem. The online solution time is on the millisecond scale, with a maximum computation time of approximately 9 ms and an average computation time of about 6.4 ms, indicating the proposed controller's potential for real-time applications.

6.4 Tracking performance evaluation (case 2)

In this subsection, we further evaluate the control performance of the four methods by assessing their effectiveness in tracking a time-varying trajectory (case 2).

To prevent the FFSR from entering a singular region, the end-effector's trajectory must satisfy Assumption 5. The target trajectory is set as

$$\boldsymbol{p}_{\mathrm{ref}}(t) = \begin{bmatrix} 0.7 - 0.05\cos\frac{\pi t}{20} \\ 0.05\sin\frac{\pi t}{20} \\ 0.7 - 0.05\cos\frac{\pi t}{20} \end{bmatrix} \mathrm{m}. \tag{35}$$

Similarly, the base satellite and each manipulator component are subject to 20% uncertainties in mass and inertia properties, and each element of unknown disturbances $\Delta_m^i(t) =$

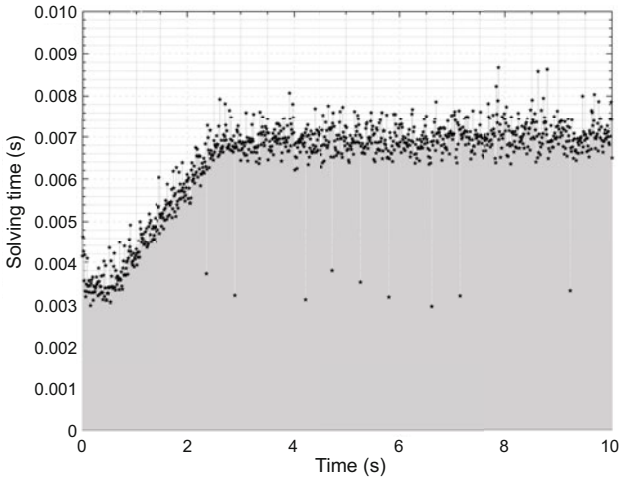

Fig. 14 Optimization time of the proposed method

$0.01 \times \left(\sin(20t) + \cos(10t) - \mathrm{e}^{-0.1t}\right)$ N·m $(i = 1, 2, \ldots, 6)$ is applied to each joint.

Fig. 15 illustrates the control results for the end-effector of the FFSR tracking a desired time-varying trajectory using the four control methods. The DK-MPC method notably fails to track the desired trajectory and exhibits the poorest control performance. The primary reason is that DK-MPC struggles to track the desired joint angles computed by DEN at each time step. Fig. 16 presents the error curves over time for the four control methods in case 2. During the initial phase of tracking control, DK-FOSMC exhibits the fastest response. However, the proposed method demonstrates superior tracking performance after the initial transient. Fig. 17 shows the control torques over time for the four control methods in case 2. Similar to case 1, DK-MPC requires the least energy consumption, but this advantage comes with a reduction in control speed and accuracy. This effect is even more pronounced in case 2 than in case 1 when tracking a time-varying trajectory. DK-FOSMC and Exp-CTFOSMC do not consider the optimization of control energy consumption and require a large control effort in the initial phase of tracking control. As a result, their energy consumption is higher compared to the proposed method. Table 6 presents a comparison of the four control methods using quantitative metrics. Relative to DK-FOSMC, the proposed method reduces RMSE by 4.05% and ITSE by 81.77%; relative

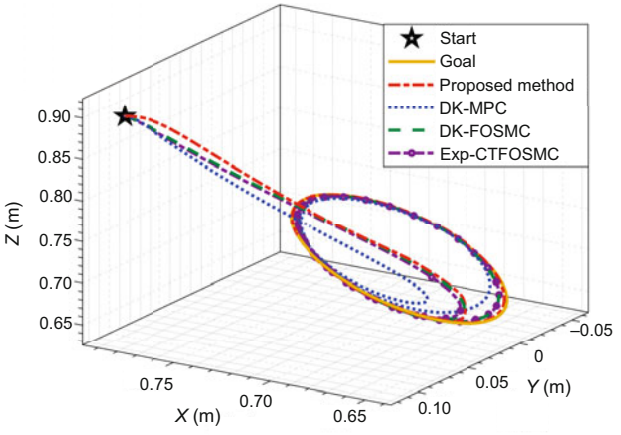

Fig. 15 3D trajectories of the four methods (case 2)

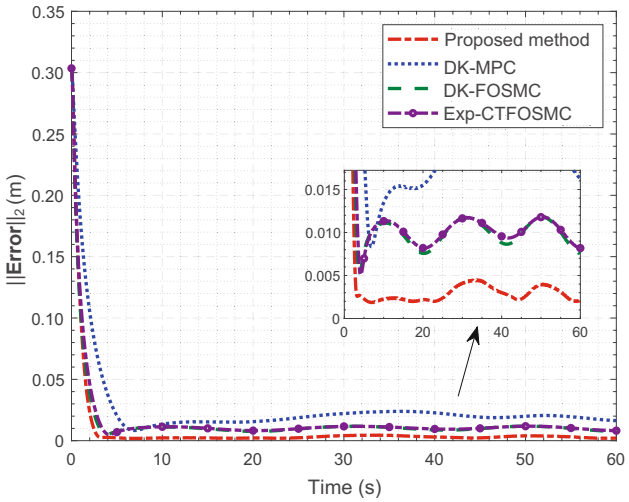

Fig. 16 Error curves of the four methods (case 2), where $\|\mathbf{Error}\|_2 = \sqrt{\|\mathbf{Error}\|_2^2}$

to Exp-CTFOSMC, it achieves reductions of 7.19% in RMSE and 82.95% in ITSE, indicating improved dynamic response and overall control performance. Moreover, by incorporating TFO variables into the MPC optimization, the proposed scheme improves energy efficiency, lowering energy consumption by 10.83% and 12.82% compared with DK-FOSMC and Exp-CTFOSMC, respectively.

Table 6 Tracking performance comparison (case 2)

Method	RMSE	ITSE	ISCI
Proposed method	$\mathbf{2.84 \times 10^{-2}}$	$\mathbf{0.37 \times 10^{3}}$	7.82×10^{2}
DK-MPC	4.24×10^{-2}	7.82×10^{3}	$\mathbf{3.11 \times 10^{2}}$
DK-FOSMC	2.96×10^{-2}	2.03×10^{3}	8.77×10^{2}
Exp-CTFOSMC	3.06×10^{-2}	2.17×10^{3}	8.97×10^{2}

The best results are in bold

To further evaluate the robustness and reliability of the proposed method under parametric uncertainties, external disturbances, and randomly sampled target points, additional Monte Carlo simulation results are provided in the supplementary materials.

7 Conclusions

This paper presents a two-layer data-driven framework for end-effector position-tracking control of micro-nano FFSRs, without relying on explicit analytical kinematic or dynamic models. By leveraging DNNs and the DK operator, the proposed method constructs a globally linearized model of the joint-space dynamics and introduces a terminal Grünwald–Letnikov fractional-order variable within the MPC framework. This integrated design enhances online control performance by improving tracking speed and accuracy compared with conventional DK-MPC. Additionally, the proposed approach achieves improved energy efficiency relative to DK-FOSMC. Simulation results demonstrate effectiveness of the proposed method, showing that it can achieve accurate, rapid, and energy-efficient end-effector trajectory tracking without reliance on explicit analytical models.

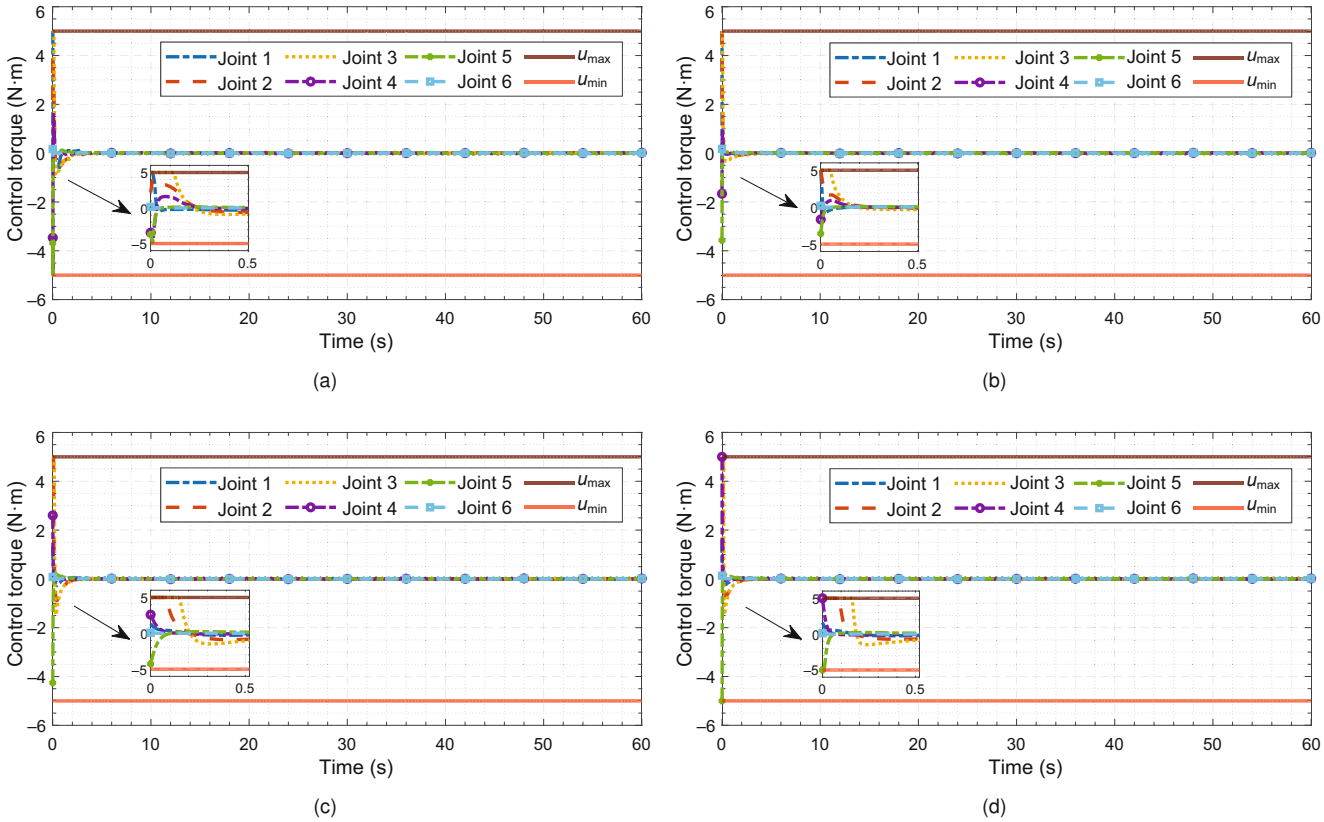

Fig. 17 Control torques of four methods (case 2): (a) proposed method; (b) DK-MPC; (c) DK-FOSMC; (d) Exp-CTFOSMC

Despite these contributions, this study has certain limitations. Notably, the DEN's fitting accuracy degrades under severe model uncertainties and over wide operating ranges. Future works could investigate the integration of online self-learning algorithms to enhance the system's robustness and validate the proposed framework through semi-physical testing as a step toward real-world deployment.

Supplementary information

The online version contains supplementary materials available at https://doi.org/10.1631/ENG.ITEE.2026.0054.

Acknowledgments

This work was supported by the Zhejiang Provincial Natural Science Foundation of China (No. LR24F030001).

Author contributions

Tao MENG supervised the project. Renhao MAO designed the research. Renhao MAO, Zhonglin ZUO, and Hang ZHOU processed the data. Renhao MAO and Kun WANG drafted the paper. Shujian SUN helped organize the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used ChatGPT to improve language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

Al Ali A, Shi JF, Zhu ZH, 2024. Path planning of 6-DOF free-floating space robotic manipulators using reinforcement learning. *Acta Astronaut*, 224:367-378. https://doi.org/10.1016/j.actaastro.2024.08.015

Bruder D, Fu X, Vasudevan R, 2021. Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics. *IEEE Rob Autom Lett*, 6(3):4369-4376. https://doi.org/10.1109/LRA.2021.3068117

Chen B, Wang MB, Hu L, et al., 2024. Data-driven Koopman model predictive control for hybrid energy storage system of electric vehicles under vehicle-following scenarios. *Appl Energy*, 365:123218. https://doi.org/10.1016/j.apenergy.2024.123218

Dou B, Yue XK, 2023. Disturbance observer-based fractional-order sliding mode control for free-floating space manipulator with disturbance. *Aerosp Sci Technol*, 132:108061. https://doi.org/10.1016/j.ast.2022.108061

Fallahiarezoodar N, Zhu ZH, 2025. Review of autonomous space robotic manipulators for on-orbit servicing and active debris removal. *Space Sci Technol*, 5:0291. https://doi.org/10.34133/space.0291

Gong K, 2025. Robust proximity rendezvous and coordinated control of space robots. *Adv Space Res*, 75(3):2856-2873. https://doi.org/10.1016/j.asr.2024.10.058

Huang ZX, Wang HQ, Niu B, et al., 2024. Practical fixed-time adaptive fuzzy control of uncertain nonlinear systems with time-varying asymmetric constraints: a unified barrier function based approach. *Front Inform Technol Electron Eng*, 25(9):1282-1294. https://doi.org/10.1631/FITEE.2300408

Jin A, Zhang F, Huang PF, 2024. Learning-based data-driven optimal deployment control of tethered space robot. *Adv Space Res*, 74(5):2214-2224. https://doi.org/10.1016/j.asr.2024.04.032

Jin RY, Rocco P, Geng YH, 2021. Observer-based fixed-time tracking control for space robots in task space. *Acta Astronaut*, 184:35-45. https://doi.org/10.1016/j.actaastro.2021.04.002

Koopman BO, 1931. Hamiltonian systems and transformation in Hilbert space. *Proc Natl Acad Sci USA*, 17(5):315-318. https://doi.org/10.1073/pnas.17.5.315

Korda M, Mezić I, 2018. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149-160. https://doi.org/10.1016/j.automatica.2018.03.046

Lavín-Delgado JE, Chávez-Vázquez S, Gómez-Aguilar JF, et al., 2023. Intelligent neural integral sliding-mode controller for a space robotic manipulator mounted on a free-floating satellite. *Adv Space Res*, 71(9):3734-3747. https://doi.org/10.1016/j.asr.2022.08.053

Lei WX, Zhao TY, Sun GH, 2023. Image based target capture of free floating space manipulator under unknown dynamics. *Adv Space Res*, 72(11):4923-4933. https://doi.org/10.1016/j.asr.2023.07.037

Li CP, Qian DL, Chen YQ, 2011. On Riemann-Liouville and Caputo derivatives. *Discrete Dyn Nat Soc*, 2011(1):562494. https://doi.org/10.1155/2011/562494

Liu RP, Guo J, Gill E, 2025. Motion planning of free-floating space robots through multi-layer optimization using the RRT* algorithm. *Acta Astronaut*, 228:940-956. https://doi.org/10.1016/j.actaastro.2024.12.036

Mao RH, Meng T, Wang K, et al., 2024. Deep Koopman-operator-based model predictive control for free-floating space robots with disturbance observer. *Aerosp Sci Technol*, 154:109515. https://doi.org/10.1016/j.ast.2024.109515

Nekoo SR, 2022. Output- and state-dependent Riccati equation: an output feedback controller design. *Aerosp Sci Technol*, 126:107649. https://doi.org/10.1016/j.ast.2022.107649

Papadopoulos E, Aghili F, Ma O, et al., 2021. Robotic manipulation and capture in space: a survey. *Front Rob AI*, 8:686723. https://doi.org/10.3389/frobt.2021.686723

Prakash A, Giri DK, Kumar SR, 2022. Dynamic velocity error based trajectory tracking for space robotic manipulator. *Aerosp Sci Technol*, 126:107650. https://doi.org/10.1016/j.ast.2022.107650

Rahmani M, Redkar S, 2024a. Deep neural data-driven Koopman fractional control of a worm robot. *Expert Syst Appl*, 256:124916. https://doi.org/10.1016/j.eswa.2024.124916

Rahmani M, Redkar S, 2024b. Enhanced Koopman operator-based robust data-driven control for 3 degree of freedom autonomous underwater vehicles: a novel approach. *Ocean Eng*, 307:118227. https://doi.org/10.1016/j.oceaneng.2024.118227

Rybus T, Wojtunik M, Basmadji FL, 2022. Optimal collision-free path planning of a free-floating space robot using spline-based trajectories. *Acta Astronaut*, 190:395-408. https://doi.org/10.1016/j.actaastro.2021.10.012

Shao XY, Sun GH, Yao WR, et al., 2021. Fractional-order resolved acceleration control for free-floating space manipulator with system uncertainty. *Aerosp Sci Technol*, 118:107041. https://doi.org/10.1016/j.ast.2021.107041

Shao YC, Jin YB, Huang ZL, et al., 2024. A learning-based control pipeline for generic motor skills for quadruped robots. *J Zhejiang Univ-Sci A*, 25(6):443-454. https://doi.org/10.1631/jzus.A2300128

Shenwai PG, Choudhary A, Pokuri T, et al., 2025. On the role of artificial intelligence in aerospace engineering: current state of the art and future trajectories. *Aeronaut J*, 129(1342):3506-3532. https://doi.org/10.1017/aer.2025.10050

Tang M, Xia WQ, Deng JQ, et al., 2025. An error-based observer improved by the repetitive control strategy for electro-optical tracking systems. *Front Inform Technol Electron Eng*, 26(3):441-455. https://doi.org/10.1631/FITEE.2300796

Tao R, Ding YB, Li HY, et al., 2025. Predefined-time controller design for a multiple space transportation robots system based on Lp-norm-normalized sign function. *Chin J Aeronaut*, 38(2):103285. https://doi.org/10.1016/j.cja.2024.10.017

Tu SL, Wang HQ, Huang Y, et al., 2024. A spaceborne advanced storage system for remote sensing microsatellites. *Front Inform Technol Electron Eng*, 25(4):600-615. https://doi.org/10.1631/FITEE.2200445