Digital Engineering and Digital Twin

Tech Science Press

DOI: 10.32604/dedt.2023.044210

ARTICLE

An Adaptive Parallel Feedback-Accelerated Picard Iteration Method for Simulating Orbit Propagation

Changtao Wang, Honghua Dai*[*] and Wenchuan Yang

School of Astronautics, Northwestern Polytechnical University, Xi’an, 710072, China

*Corresponding Author: Honghua Dai. Email: hhdai@nwpu.edu.cn

Received: 24 July 2023 Accepted: 10 November 2023 Published: 28 December 2023

ABSTRACT

A novel Adaptive Parallel Feedback-Accelerated Picard Iteration (AP-FAPI) method is proposed to meet the requirements of various aerospace missions for fast and accurate orbit propagation. The Parallel Feedback-Accelerated Picard Iteration (P-FAPI) method is an advanced iterative collocation method. With large-step computing and parallel acceleration, the P-FAPI method outperforms the traditional finite-difference-based methods, which require small-step and serial integration to ensure accuracy. Although efficient and accurate, the P-FAPI method suffers extensive trials in tuning method parameters, strongly influencing its performance. To overcome this problem, we propose the AP-FAPI method based on the relationship between the parameters and the convergence speed leveraging parallel technology. The proposed method ensures the best possible efficiency in prescribed accuracy. Three typical orbit propagation problems are illustrated to validate the high performance of the proposed method. Numerical simulations show that the AP-FAPI method is more accurate and efficient than the finite-difference-based and the P-FAPI methods.

KEYWORDS

Numerical integration method; adaptive method; orbit propagation; parallel computing

Nomenclature

$\boldsymbol{x}$, x_d	System state and its d-th component
$\tilde{\boldsymbol{x}}$, $\boldsymbol{\chi}_d$	System state vector at CGL nodes and its d-th component
$\boldsymbol{g}$, g_d	System dynamics and its d-th component
$\tilde{\boldsymbol{g}}$, $\bar{\boldsymbol{g}}_d$	System dynamics vector at CGL nodes and its d-th component
$\boldsymbol{J}$, $J_{i,j}$	Jacobian matrix and its (i,j) entry

1 Introduction

In recent years, the complexity of aerospace missions has grown, leading to an urgent need for precise and real-time orbit computing. For instance, it is crucial to obtain accurate long-term orbit propagation results in spacecraft on-orbit service operations [1]; meanwhile, the real-time computation of spacecraft position is vital in guidance navigation [2,3] and formation flying tasks [4,5]. These missions all require fast and accurate methods for solving orbit propagation problems.

Unfortunately, the traditional numerical methods based on finite difference method fail to meet the demand for high accuracy and high efficiency simultaneously because they rely on small integration steps to ensure accuracy [6]. To overcome this challenge, scholars have proposed a family of iterative collocation methods without the limitation of small integration steps, such as the Modified Chebyshev-Picard Iteration method and the Feedback-Accelerated Picard Iteration method [7–10]. These methods approach the real solution by correcting the approximated solution iteratively and simplify integration operations using the idea of discretization, which has been applied extensively in aerospace missions such as orbit propagation, Lambert's problem and so on [11–14]. Some excellent latest research for iterative collocation methods have been developed, such as the new balanced space-time Sinc-collocation method [15] and the error analysis of the Orthogonal Spline Collocation (OSC) method [16], which contributes effective method and improves the theoretical foundation. Moreover, these methods exhibit scalability to be parallel accelerated [17,18]. The Parallel Feedback-Accelerated Picard Iteration (P-FAPI) method is a representative parallel iterative collocation method that can quickly obtain highly accurate solutions to orbit propagation problems. However, improper method parameters will reduce the efficiency and accuracy of the P-FAPI method. While some adaptive methods exist for iterative collocation methods [19,20], there is a lack of adaptive methods utilizing parallel technology for parallel iterative collocation methods.

In this paper, we propose a new Adaptive Parallel Feedback-Accelerated Picard Iteration (AP-FAPI) method, which refines the method parameters of the P-FAPI method to maximize efficiency while ensuring accuracy. To achieve the highest possible efficiency, we define the method's computation speed and optimize it according to the relationship between the step size and the convergence speed represented by the iteration count. More importantly, we can efficiently obtain a series of iteration counts with parallel technology. To ensure the desired accuracy, we terminate prematurely in case of an excessive iteration count or significant computational error and consider the current iteration non-convergent.

This paper is organized as follows. In Section 2, we introduce the theory of the P-FAPI. In Section 3, we propose the Adaptive Parallel Feedback-Accelerated Picard Iteration method. In Section 4, we illustrate three orbit propagation cases to verify the high performance of the AP-FAPI method. Finally, we present our conclusions in Section 5.

2 Parallel Feedback-Accelerated Picard Iteration Method

The orbit propagation problems are often described as the nonlinear ordinary differential equation. The Parallel Feedback-Accelerated Picard Iteration (P-FAPI) method is an advanced iterative collocation method for solving these problems, which combines the variational iteration method [21] with the collocation method. The P-FAPI method can approach the real solution in large steps with parallel acceleration. This section introduces the P-FAPI method by solving the general first-order nonlinear differential equation as follows:

$$\begin{cases} \dfrac{d\boldsymbol{x}}{dt} = \boldsymbol{g}(\boldsymbol{x}(t), t), t \in \left[t_0, t_f\right] \\ \boldsymbol{x}\left(t_0\right) = \boldsymbol{x}_0 \end{cases} \tag{1}$$

where $\boldsymbol{x}\left(t\right) = \left[x_1(t), x_2(t), \ldots, x_d(t), \ldots, x_D\left(t\right)\right]^{\mathrm{T}}$ is a D-dimensional vector and $x_d\left(t\right)$ denotes the d-th component of $\boldsymbol{x}\left(t\right)$. Here, we set $t_0 = -1, t_f = 1$.

In the variational iterative method, the solution will converge iteratively to the real solution with an initial guess $\boldsymbol{x}_0\,(t)$ and the following corrective iterative formula:

$$\boldsymbol{x}_{n+1}\,(t) = \boldsymbol{x}_n\,(t) + \int_{-1}^{t} \boldsymbol{\lambda}\,(\tau)\,\{\dot{\boldsymbol{x}}_n\,(\tau) - \boldsymbol{g}\,[\boldsymbol{x}_n(\tau), \tau]\}\,d\tau \tag{2}$$

where $\boldsymbol{\lambda}\,(\tau)$ is a generalized Lagrange multiplier to be determined. Based on the variational principle, we obtain the first-order Taylor series expansion of $\boldsymbol{\lambda}\,(\tau)$ at the point of $\tau = t$:

$$\boldsymbol{\lambda}\,(\tau) \approx \boldsymbol{\lambda}\,(t) + \dot{\boldsymbol{\lambda}}\,(t)\,(\tau - t) = -\boldsymbol{I} + \boldsymbol{J}\,(t)\,(\tau - t) \tag{3}$$

where $\boldsymbol{I}$ is a unit matrix and $\boldsymbol{J}\,(t) = \partial \boldsymbol{g}\,(\boldsymbol{x}_n, t)/\partial \boldsymbol{x}_n$. Substituting Eq. (3) into Eq. (2) and abbreviating $\boldsymbol{G}\,(\tau) = \dot{\boldsymbol{x}}_n\,(\tau) - \boldsymbol{g}\,[\boldsymbol{x}_n(\tau), \tau] = [G_1(\tau), G_2(\tau), \ldots, G_d(\tau), \ldots, G_D\,(\tau)]^{\mathrm{T}}$, we have

$$\boldsymbol{x}_{n+1}\,(t) = \boldsymbol{x}_n\,(t) - \{\boldsymbol{I} + \boldsymbol{J}\,(t)\,t\} \int_{-1}^{t} \boldsymbol{G}\,(\tau)\,d\tau + \boldsymbol{J}\,(t) \int_{-1}^{t} \tau \boldsymbol{G}\,(\tau)\,d\tau \tag{4}$$

Then the linear combination of N+1 Chebyshev orthogonal basis functions is used to approximate each element of $\boldsymbol{x}\,(t)$, $\dot{\boldsymbol{x}}\,(t)$, $\boldsymbol{G}\,(t)$ and $\int_{-1}^{t} \boldsymbol{G}\,(\tau)\,d\tau$:

$$\begin{cases} x_d\,(t) \approx \sum\limits_{n=0}^{N} \alpha_{d,n}\phi_n\,(t) = \boldsymbol{\Phi}\,(t)\,\boldsymbol{A}_d \\ \dot{x}_d\,(t) \approx \sum\limits_{n=0}^{N} \alpha_{d,n}\dot{\phi}_n\,(t) = \dot{\boldsymbol{\Phi}}\,(t)\,\boldsymbol{A}_d \\ G_d\,(t) \approx \sum\limits_{n=0}^{N} \beta_{d,n}\phi_n\,(t) = \boldsymbol{\Phi}\,(t)\,\boldsymbol{B}_d \\ \int_{-1}^{t} G_d\,(\tau)\,d\tau \approx \sum\limits_{n=0}^{N} \beta_{d,n} \int_{-1}^{t} \phi_n\,(\tau)\,d\tau = \int_{-1}^{t} \boldsymbol{\Phi}\,(\tau)\,d\tau \boldsymbol{B}_d \end{cases} \tag{5}$$

where $\boldsymbol{A}_d = [\alpha_{d,0}, \alpha_{d,1}, \ldots, \alpha_{d,N}]^{\mathrm{T}}$ and $\boldsymbol{B}_d = [\beta_{d,0}, \beta_{d,1}, \ldots, \beta_{d,N}]^{\mathrm{T}}$ are the coefficient vectors of the basis function, $\boldsymbol{\Phi}\,(t) = [\phi_0(t), \phi_1(t), \ldots, \phi_N\,(t)]$ is the vector of the Chebyshev basis function.

Next, Eq. (5) is discretized by the collocation method. Here Chebyshev-Gauss-Lobatto (CGL) nodes are chosen as the collocation points, which are defined as

$$t_j = -\cos(\frac{(j-1)\,\pi}{M-1}), j = 1, 2, \ldots, M \tag{6}$$

where the number of collocation points M is usually selected as N+1, the same as the number of Chebyshev orthogonal basis functions. After discretizing, we have

$$\begin{bmatrix} \dot{x}_d\,(t_1) \\ \dot{x}_d\,(t_2) \\ \vdots \\ \dot{x}_d\,(t_M) \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\Phi}}\,(t_1) \\ \dot{\boldsymbol{\Phi}}\,(t_2) \\ \vdots \\ \dot{\boldsymbol{\Phi}}\,(t_M) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}\,(t_1) \\ \boldsymbol{\Phi}\,(t_2) \\ \vdots \\ \boldsymbol{\Phi}\,(t_M) \end{bmatrix}^{-1} \begin{bmatrix} x_d\,(t_1) \\ x_d\,(t_2) \\ \vdots \\ x_d\,(t_M) \end{bmatrix} = \boldsymbol{Q}\boldsymbol{\chi}_d \tag{7}$$

and

$$\begin{bmatrix} \int_{-1}^{t_1} G_d\,(\tau)\,d\tau \\ \int_{-1}^{t_2} G_d\,(\tau)\,d\tau \\ \vdots \\ \int_{-1}^{t_M} G_d\,(\tau)\,d\tau \end{bmatrix} = \begin{bmatrix} \int_{-1}^{t_1} \boldsymbol{\Phi}\,(\tau)\,d\tau \\ \int_{-1}^{t_2} \boldsymbol{\Phi}\,(\tau)\,d\tau \\ \vdots \\ \int_{-1}^{t_M} \boldsymbol{\Phi}\,(\tau)\,d\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}\,(t_1) \\ \boldsymbol{\Phi}\,(t_2) \\ \vdots \\ \boldsymbol{\Phi}\,(t_M) \end{bmatrix}^{-1} \begin{bmatrix} G_d\,(t_1) \\ G_d\,(t_2) \\ \vdots \\ G_d\,(t_M) \end{bmatrix} = \boldsymbol{P}\boldsymbol{G}_d \tag{8}$$

By substituting Eqs. (7) and (8) into Eq. (4), an iterative formula in matrix form is derived as follows:

$$\begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \\ \vdots \\ \boldsymbol{G}_d \\ \vdots \\ \boldsymbol{G}_D \end{bmatrix}_n = \left(\begin{bmatrix} \boldsymbol{Q} & & \\ & \ddots & \\ & & \boldsymbol{Q} \end{bmatrix} \begin{bmatrix} \boldsymbol{\chi}_1 \\ \boldsymbol{\chi}_2 \\ \vdots \\ \boldsymbol{\chi}_d \\ \vdots \\ \boldsymbol{\chi}_D \end{bmatrix}_n - \begin{bmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \\ \vdots \\ \tilde{\boldsymbol{g}}_d \\ \vdots \\ \tilde{\boldsymbol{g}}_D \end{bmatrix}_n \right) \tag{9}$$

$$\begin{aligned} \tilde{\boldsymbol{x}}_{n+1} = \begin{bmatrix} \boldsymbol{\chi}_1 \\ \boldsymbol{\chi}_2 \\ \vdots \\ \boldsymbol{\chi}_d \\ \vdots \\ \boldsymbol{\chi}_D \end{bmatrix}_{n+1} &= \begin{bmatrix} \boldsymbol{\chi}_1 \\ \boldsymbol{\chi}_2 \\ \vdots \\ \boldsymbol{\chi}_d \\ \vdots \\ \boldsymbol{\chi}_D \end{bmatrix}_n \\ &+ \begin{bmatrix} \boldsymbol{J}_{11}\boldsymbol{H} - \boldsymbol{P} & \boldsymbol{J}_{12}\boldsymbol{H} & \dots & \boldsymbol{J}_{1d}\boldsymbol{H} & \dots & \boldsymbol{J}_{1D}\boldsymbol{H} \\ \boldsymbol{J}_{21}\boldsymbol{H} & \mathrm{J}_{22}\boldsymbol{H} - \boldsymbol{P} & \dots & \boldsymbol{J}_{2d}\boldsymbol{H} & \dots & \boldsymbol{J}_{2D}\boldsymbol{H} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{J}_{d1}\boldsymbol{H} & \boldsymbol{J}_{d2}\boldsymbol{H} & \dots & \boldsymbol{J}_{dd}\boldsymbol{H} - \boldsymbol{P} & \dots & \boldsymbol{J}_{dD}\boldsymbol{H} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{J}_{D1}\boldsymbol{H} & \boldsymbol{J}_{D2}\boldsymbol{H} & \dots & \boldsymbol{J}_{Dd}\boldsymbol{H} & \dots & \boldsymbol{J}_{DD}\boldsymbol{H} - \boldsymbol{P} \end{bmatrix}_n \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \\ \vdots \\ \boldsymbol{G}_d \\ \vdots \\ \boldsymbol{G}_D \end{bmatrix}_n \\ &= \tilde{\boldsymbol{x}}_n + \boldsymbol{J}_n^* \boldsymbol{G}_n \end{aligned} \tag{10}$$

The details of $\boldsymbol{P}$, $\boldsymbol{Q}$, $\tilde{\boldsymbol{g}}_d$, $\boldsymbol{J}_{i,j}$ and $\boldsymbol{H}$ are provided in Appendix A. Then the flowchart of the P-FAPI method is obtained as Fig. 1.

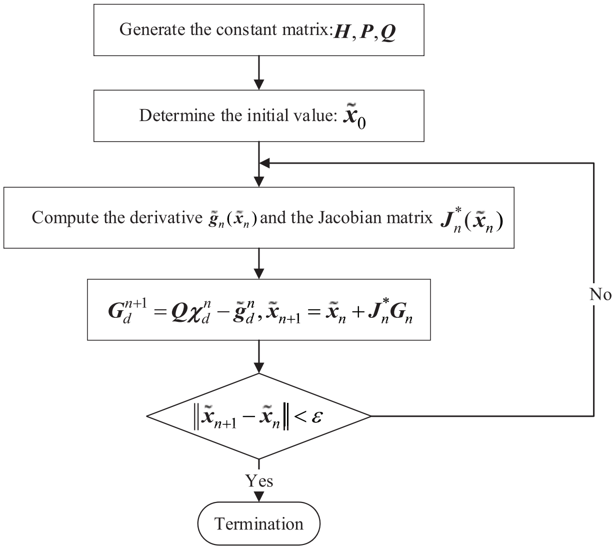
Generate the constant matrix: $\boldsymbol{H}, \boldsymbol{P}, \boldsymbol{Q}$
Determine the initial value: $\tilde{\boldsymbol{x}}_0$
Compute the derivative $\tilde{\boldsymbol{g}}_n(\tilde{\boldsymbol{x}}_n)$ and the Jacobian matrix $\boldsymbol{J}_n^*(\tilde{\boldsymbol{x}}_n)$
$\boldsymbol{G}_d^{n+1} = \boldsymbol{Q}\boldsymbol{\chi}_d^n - \tilde{\boldsymbol{g}}_d^n, \tilde{\boldsymbol{x}}_{n+1} = \tilde{\boldsymbol{x}}_n + \boldsymbol{J}_n^*\boldsymbol{G}_n$
$\left\| \tilde{\boldsymbol{x}}_{n+1} - \tilde{\boldsymbol{x}}_n \right\| < \varepsilon$
No
Yes
Termination

Figure 1: Flowchart of the P-FAPI method

3 Adaptive Parallel Feedback-Accelerated Picard Iteration Method

The performance of the P-FAPI method is heavily affected by the parameters, such as the number of collocations and step size. Excessive collocation points and small steps decrease the method's efficiency, whereas few collocation points and large steps compromise accuracy. In other words, the mismatch between the number of collocation points and step size will reduce performance. To achieve the highest possible efficiency while meeting accuracy requirements, we propose the Adaptive Parallel Feedback-Accelerated Picard Iteration (AP-FAPI) method by adjusting the step size to match the number of collocation points with inspiration from the finite volume scheme [22,23]. The adjustment principle of step size is presented as follows.

To characterize the efficiency of the P-FAPI method, we define the computation speed as follows:

$$v_m = \frac{dt}{I_c} \cdot \frac{1}{t_c} \tag{11}$$

where I_c, t_c, and dt denote the iteration count, the computational time of one iteration, and the step size, respectively. t_c is determined by the number of collocation points M, and I_c is associated with dt and M.

In the P-FAPI method, the number of collocation points is usually determined according to the compute architecture, meaning t_c is considered a constant. Under this premise, we maximize the equivalent step size for unit iteration dt/I_c in Eq. (11) by adjusting the step size to improve the computation speed as much as possible, where the iteration count I_c is recorded during the computation process. With CUDA stream technology [24], one computing process can obtain a group of iteration counts, dramatically increasing the method's efficiency. The process of the AP-FAPI method is illustrated as follows and shown in Fig. 2.

1. Generate a group containing S steps by the initial step size dt as follows:

   $$G_s = [n_1 \times dt, n_2 \times dt, \ldots, n_m \times dt, \ldots, n_S \times dt], \quad 0 < n_1 < n_2 < \ldots < n_m < \ldots < n_S \tag{12}$$

2. Launch S CUDA streams for calculation of the P-FAPI method with step in G_s as the parameter.
3. Without loss of generality, we can assume that there are K converged calculations in the above CUDA streams and their iteration counts are $I_{c1}, I_{c2}, \ldots, I_{cm}, \ldots, I_{cK}$, respectively, where $0 \le K \le S$.
4. When $K = 0$, set step size as $dt = dt/n_S$ and recalculate within this interval. Otherwise, record the K-th CUDA stream's result, compute the next interval, and determine its step size based on the following formula:

$$dt = \begin{cases} dt \times n_m & K = S \\ dt/n_m & K < S \end{cases}, \quad n_m = \underset{m \in [1,2,\ldots,K]}{argmax} \frac{n_m \times dt}{I_{cm}} \tag{13}$$

To further ensure the accuracy of the P-FAPI method, we defined the maximum iteration count I_m and the maximum iteration error E_m. During the iterative process, if the iteration count exceeds the maximum iteration count or the iteration error exceeds the maximum iteration error, the iteration is considered non-convergent.

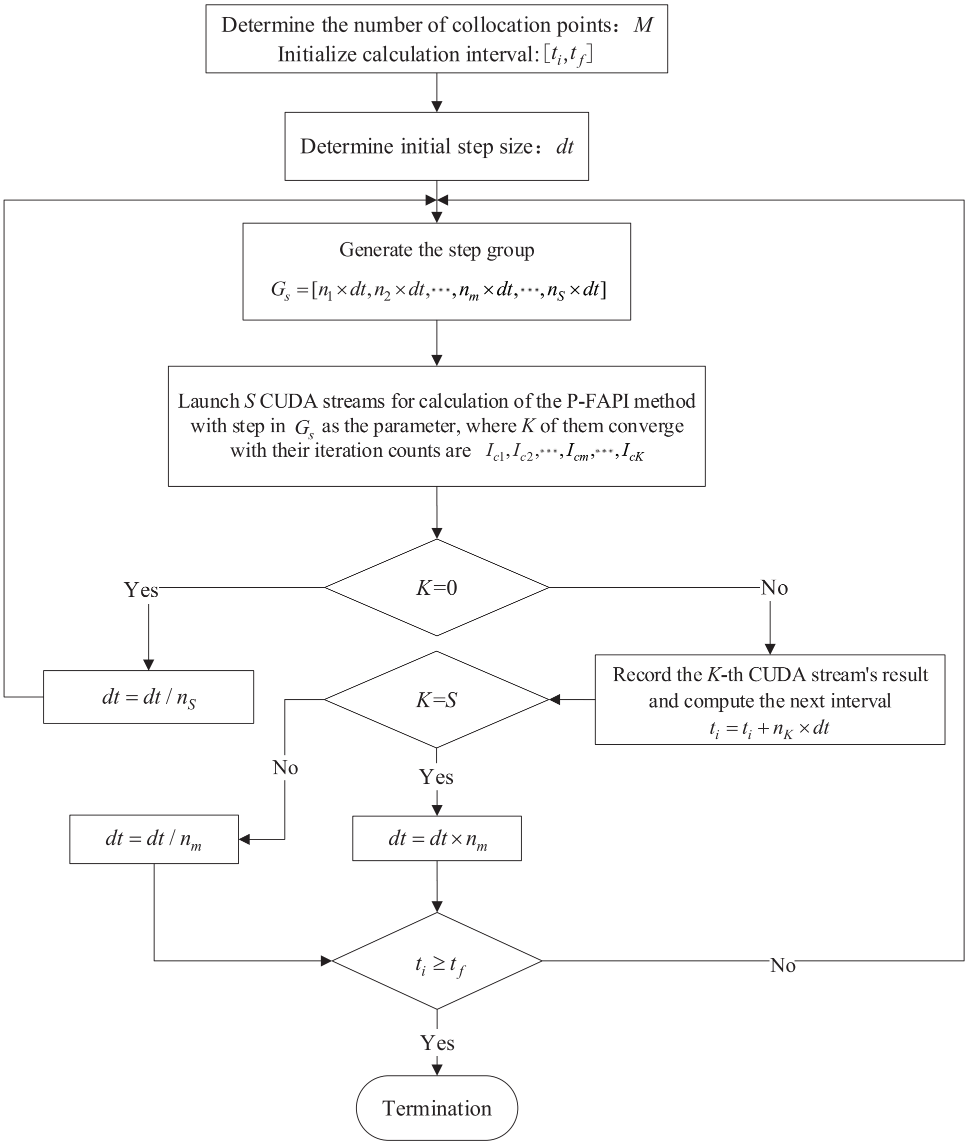
Determine the number of collocation points：M
Initialize calculation interval: $[t_i, t_f]$
Determine initial step size：dt
Generate the step group
$G_s = [n_1 \times dt, n_2 \times dt, \cdots, n_m \times dt, \cdots, n_S \times dt]$
Launch S CUDA streams for calculation of the P-FAPI method with step in G_s as the parameter, where K of them converge with their iteration counts are $I_{c1}, I_{c2}, \cdots, I_{cm}, \cdots, I_{cK}$
K=0
Yes
No
$dt = dt / n_S$
Record the K-th CUDA stream's result and compute the next interval
$t_i = t_i + n_K \times dt$
K=S
No
Yes
$dt = dt / n_m$
$dt = dt \times n_m$
$t_i \geq t_f$
No
Yes
Termination

Figure 2: Flowchart of the AP-FAPI method

4 Numerical Simulation

In this section, the high performance of the AP-FAPI method is validated in three typical orbit propagation problems based on the 40th-order EGM: a Low Earth Orbit (LEO) propagation problem, a Highly Elliptical Orbit (HEO) problem, and a Geostationary Earth Orbit (GEO) propagation problem. Their initial parameters and intervals are shown in Table 1, and the schematic diagram is shown in Fig. 3.

The numerical simulation is carried out in Microsoft Visual Studio 2019, using a computer with the Intel 9 12900K CPU and NVIDIA Quadro P1000 GPU. The performance of the AP-FAPI method is compared with that of the P-FAPI method and ODE45, the implementation of DP5(4) in MATLAB. Their parameters are shown in Tables 2 and 3.

Table 1: Initial parameters and intervals of different orbits

Type of orbit	$\boldsymbol{r}_0$/m	$\boldsymbol{v}_0$/(m·s^{-1})	$[t_0, t_f]/s$
LEO	$\begin{pmatrix} -0.3889 \\ 7.7388 \\ 0.6736 \end{pmatrix} \times 10^6$	$\begin{pmatrix} -3.5794 \\ 0 \\ 6.1997 \end{pmatrix} \times 10^3$	$[0, 8 \times 10^4]$
HEO	$\begin{pmatrix} 4.0500 \\ 0 \\ -7.0148 \end{pmatrix} \times 10^6$	$\begin{pmatrix} 0 \\ 9.1464 \\ 0 \end{pmatrix} \times 10^3$	$[0, 5 \times 10^5]$
GEO	$\begin{pmatrix} 4.2164 \\ 0 \\ 0 \end{pmatrix} \times 10^7$	$\begin{pmatrix} 0 \\ 3.0747 \\ 0 \end{pmatrix} \times 10^3$	$[0, 9 \times 10^5]$

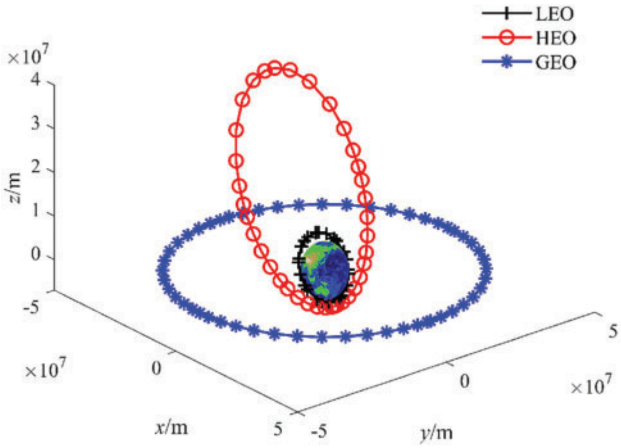

Figure 3: Illustration of LEO, HEO and GEO

Table 2: Parameters of the P-FAPI method and the AP-FAPI method in different orbits

Type of orbit	Method	M	dt (m)	tol (m)	I_m	E_m (m)	G_s
LEO	P-FAPI	64	200	1×10^{-6}	–		
	AP-FAPI				40	1×10^8	$[dt, 2 \times dt]$
HEO	P-FAPI	64	500	2×10^{-5}	–		
	AP-FAPI				40	1×10^8	$[dt, 2 \times dt]$
GEO	P-FAPI	64	2250	3×10^{-5}	–		
	AP-FAPI				40	1×10^{10}	$[dt, 2 \times dt]$

Table 3: Parameters of ODE45 in different orbits

Type of orbit	Relative error	Absolute error
LEO	1×10^{-10}	1×10^{-10}
HEO		
GEO		

The positional errors of different methods are shown in Fig. 4 (The reference solutions are obtained by ODE45 with the highest accuracy). The computational times and iteration counts of different methods are shown in Table 4. The iteration counts and generated step sizes in the AP-FAPI method are shown in Fig. 5.

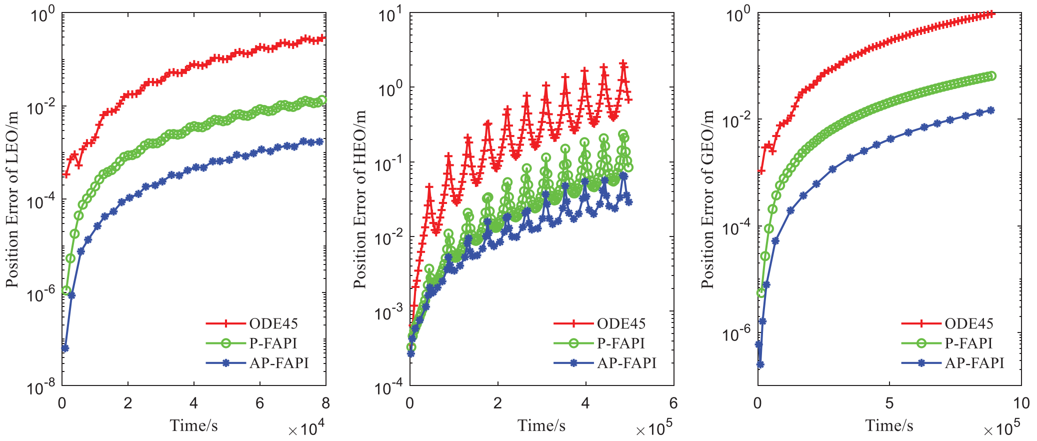

Figure 4: Positional errors of different methods

Table 4: Computational times and iteration counts of different methods

Type of orbits		ODE45	P-FAPI	AP-FAPI
LEO	Computational time	73.40 s	26.43 s	11.68 s
	Iteration count	–	2400	1012
HEO	Computational time	104.34 s	53.85 s	22.07 s
	Iteration count	–	4743	1627
GEO	Computational time	68.42 s	26.43 s	11.03 s
	Iteration count	–	2400	499

In Fig. 4, the results in three illustrative examples demonstrate that the P-FAPI method attains approximately one order of magnitude higher accuracy compared to ODE45, while the AP-FAPI method achieves approximately two orders of magnitude higher accuracy. The results in Table 4 indicate that the P-FAPI method is at least twice as fast as ODE45, and the AP-FAPI method is at least four times faster than ODE45. Fig. 5 demonstrates that in LEO and GEO propagation problems, the step size rapidly adjusts to an optimal value after a few iterations and remains consistent throughout the subsequent computations. In contrast, in HEO propagation problems, the step size varies frequently throughout the calculation process. This aligns with the fact that the LEO and GEO curves are flat, whereas the HEO curves exhibit a steep.

Overall, the results from the three simulations confirm that the AP-FAPI method with optimized parameters achieves greater accuracy and efficiency when compared to ODE45 and the P-FAPI method.

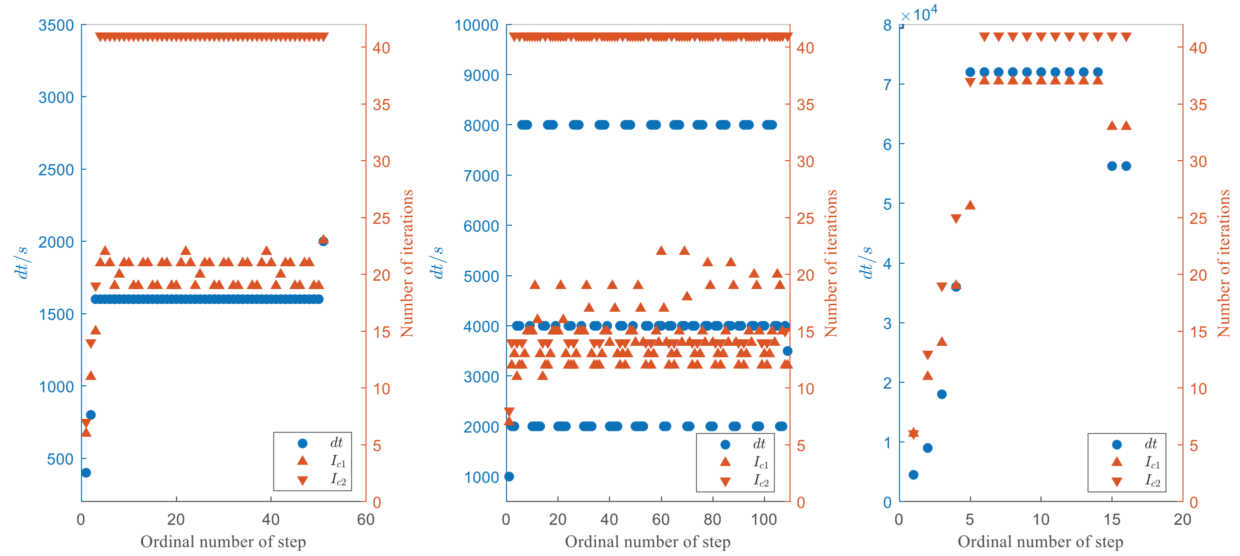

Figure 5: Iteration counts and generated step sizes in the AP-FAPI method

5 Conclusion

We proposed a novel adaptive parallel feedback accelerated Picard iteration method by refining the parameter during calculation. A mismatch between the parameters of the P-FAPI method, namely the number of collocation points and step size, will result in reduced performance. The proposed method could optimize efficiency while meeting the expected accuracy. It dynamically adjusts the step size to align with the number of collocation points based on the relationship between the step size and convergence represented by the iteration count. The high performance of the proposed method is verified in solving three typical orbit propagation problems. Numerical simulations show that the AP-FAPI method is at least four times faster than ODE45 and twice as fast as the P-FAPI method. It is also the most accurate method, yielding two orders of magnitude higher accuracy than ODE45 and one order higher than the P-FAPI method. The proposed method can be applied to solve various nonlinear dynamic problems efficiently and accurately. Future research can focus on the application of the AP-FAPI method in more complicated systems to explore its efficiency [25].

Acknowledgement: We would like to express our sincere appreciation for the exceptional contributions made by Professor Satya Atluri in the field of computational mechanics and engineering. On the occasion of Professor Satya Atluri's 80th birthday, we extend our warmest wishes for good health and continued success in his academic endeavors. Additionally, we gratefully acknowledge Professor Dai for his invaluable guidance and meticulous support throughout the preparation of this paper.

Funding Statement: This study was co-supported by the National Key Research and Development Program of China (No. 2021YFA0717100) and the National Natural Science Foundation of China (Nos. 12072270, U2013206).

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and adaptive method design: C. Wang, H. Dai; test case design: C. Wang, W. Yang; analysis and interpretation of results: C. Wang, W. Yang; draft manuscript preparation: C. Wang, W. Yang; All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used in this study are available upon request from the GitHub: https://github.com/CT-Wang/TSP-DEDT-data.git.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Zhang, Z., Deng, L., Feng, J., Chang, L., Li, D. et al. (2022). A survey of precision formation relative state measurement technology for distributed spacecraft. *Aerospace, 9(7),* 362.
2. Zhang, H., Li, J., Wang, Z., Guan, Y. (2021). *Guidance navigation and control for chang'E-5 powered descent*. Space: Science & Technology.
3. Huang, X., Li, M., Wang, X., Hu, J., Zhao, Y. et al. (2021). The Tianwen-1 guidance, navigation, and control for mars entry, descent, and landing. *Space: Science & Technology, 2021,* 9846185.
4. Gao, C., Zhao, Y. (2023). Model predictive control-based coupled orbit-attitude control for solar sail formation flying in the Earth-Moon system. *Astrophysics and Space Science, 368(6),* 47.
5. Bandyopadhyay, S., Foust, R., Subramanian, G. P., Chung, S. J., Hadaegh, F. Y. (2016). Review of formation flying and constellation missions using nanosatellites. *Journal of Spacecraft and Rockets, 53(3),* 567–578.
6. Chen, W., Cheng, J., Wu, X. (2014). *Numerical solutions of differential equations*. Shanghai, China: Fudan University Press.
7. Wang, X., Yue, X., Dai, H., Atluri, S. N. (2017). Feedback-accelerated picard iteration for orbit propagation and lambert's problem. *Journal of Guidance Control and Dynamics, 40(10),* 2442–2451.
8. Wang, Y., Ni, G., Liu, Y. (2020). Multistep newton-picard method for nonlinear differential equations. *Journal of Guidance, Control, and Dynamics, 43(11),* 2148–2155.
9. Bai, X., Junkins, J. L. (2011). Modified chebyshev-picard iteration methods for orbit propagation. *The Journal of the Astronautical Sciences, 58(4),* 583–613.
10. Woollands, R., Junkins, J. L. (2019). Nonlinear differential equation solvers via adaptive picard-chebyshev iteration: Applications in astrodynamics. *Journal of Guidance, Control, and Dynamics, 42(5),* 1007–1022.
11. Bai, X., Junkins, J. L. (2011). Modified chebyshev-picard iteration methods for solution of boundary value problems. *The Journal of the Astronautical Sciences, 58(4),* 615–642.
12. Feng, H., Yue, X., Wang, X. (2021). A quasi-linear local variational iteration method for orbit transfer problems. *Aerospace Science and Technology, 119,* 107222.
13. Wang, X., Xu, Q., Atluri, S. N. (2020). Combination of the variational iteration method and numerical algorithms for nonlinear problems. *Applied Mathematical Modelling, 79,* 243–259.
14. Feng, H., Yue, X., Wang, X., Zhang, Z. (2023). Decoupling and quasi-linearization methods for boundary value problems in relative orbital mechanics. *Nonlinear Dynamics, 111(1),* 199–215.
15. Yang, X., Wu, L., Zhang, H. (2023). A space-time spectral order sinc-collocation method for the fourth-order nonlocal heat model arising in viscoelasticity. *Applied Mathematics and Computation, 457,* 128192.

16. Zhang, H., Yang, X., Tang, Q., Xu, D. (2023). A robust error analysis of the OSC method for a multi-term fourth-order sub-diffusion equation. *Computers & Mathematics with Applications, 109(1),* 180–190.
17. Bai, X. (2010). *Modified Chebyshev-Picard iteration methods for solution of initial value and boundary value problems (Ph.D. Thesis)*. Texas A&M University, USA.
18. Wang, C., Dai, H., Zhang, Z., Wang, X., Yue, X. (2023). Parallel accelerated local variational iteration method and its application in orbit computation. *Chinese Journal of Theoretical and Applied Mechanics, 55(4),* 991–1003.
19. Dai, H., Zhang, Z., Wang, X., Feng, H., Wang, C. et al. (2023). Fast and accurate adaptive collocation iteration method for orbit dynamic problems. *Chinese Journal of Aeronautics, 36(9),* 231–242.
20. Wang, X., Elgohary, T. A., Zhang, Z., Tasif, T. H., Feng, H. et al. (2023). An adaptive local variational iteration method for orbit propagation in astrodynamics problems. *The Journal of the Astronautical Sciences, 70(1),* 2.
21. He, J. (1999). Variational iteration method—A kind of non-linear analytical technique: Some examples. *International Journal of Non-Linear Mechanics, 34(4),* 699–708.
22. Yang, X., Zhang, H., Zhang, Q., Yuan, G. (2022). Simple positivity-preserving nonlinear finite volume scheme for subdiffusion equations on general non-conforming distorted meshes. *Nonlinear Dynamics, 108(4),* 3859–3886.
23. Yang, X., Zhang, H., Zhang, Q., Yuan, G., Sheng, Z. (2019). The finite volume scheme preserving maximum principle for two-dimensional time-fractional Fokker-Planck equations on distorted meshe. *Applied Mathematics Letters, 97,* 99–106.
24. Cheng, J., Grossman, M., McKercher, T. (2014). *Professional CUDA C programming*. USA: John Wiley & Sons.
25. Zhang, H., Liu, Y., Yang, X. (2014). An efficient ADI difference scheme for the nonlocal evolution problem in three-dimensional space. *Journal of Applied Mathematics and Computing, 69(1),* 651–674.

Appendix A

In Eq. (10), the matrixes $\boldsymbol{Q}$, $\boldsymbol{P}$, $\boldsymbol{J}_{i,j}$, $\boldsymbol{H}$ and the vector $\tilde{\boldsymbol{g}}_d$ are as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \dot{\boldsymbol{\Phi}}(t_1) \\ \dot{\boldsymbol{\Phi}}(t_2) \\ \vdots \\ \dot{\boldsymbol{\Phi}}(t_M) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}(t_1) \\ \boldsymbol{\Phi}(t_2) \\ \vdots \\ \boldsymbol{\Phi}(t_M) \end{bmatrix}^{-1}, \boldsymbol{P} = \begin{bmatrix} \int_{-1}^{t_1} \boldsymbol{\Phi}(\tau)\, d\tau \\ \int_{-1}^{t_2} \boldsymbol{\Phi}(\tau)\, d\tau \\ \vdots \\ \int_{-1}^{t_M} \boldsymbol{\Phi}(\tau)\, d\tau \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}(t_1) \\ \boldsymbol{\Phi}(t_2) \\ \vdots \\ \boldsymbol{\Phi}(t_M) \end{bmatrix}^{-1} \tag{A.1}$$

$$\boldsymbol{J}_{i,j} = \text{dig}\left(\frac{\partial g_i(\boldsymbol{x}(t_1), t_1)}{x_j(t_1)}, \quad \ldots \quad , \frac{\partial g_i(\boldsymbol{x}(t_M), t_M)}{x_j(t_M)} \right) \tag{A.2}$$

$$\boldsymbol{T} = \text{dig}(t_1, t_2, \ldots, t_M), \boldsymbol{H} = \boldsymbol{PT} - \boldsymbol{TP} \tag{A.3}$$

$$\tilde{\boldsymbol{g}}_d = \begin{bmatrix} g_d(\boldsymbol{x}(t_1), t_1) \\ g_d(\boldsymbol{x}(t_2), t_2) \\ \vdots \\ g_d(\boldsymbol{x}(t_M), t_M) \end{bmatrix} \tag{A.4}$$