Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.043068

ARTICLE

Fast and Accurate Predictor-Corrector Methods Using Feedback-Accelerated Picard Iteration for Strongly Nonlinear Problems

Xuechuan Wang[1], Wei He[1,*], Haoyang Feng[1] and Satya N. Atluri[2]

[1]School of Astronautics, Northwestern Polytechnical University, Xi'an, 710072, China

[2]Department of Mechanical Engineering, Texas Tech University, Lubbock, 79409, USA

*Corresponding Author: Wei He. Email: heweicz@mail.nwpu.edu.cn

Received: 20 June 2023 Accepted: 27 October 2023 Published: 29 January 2024

ABSTRACT

Although predictor-corrector methods have been extensively applied, they might not meet the requirements of practical applications and engineering tasks, particularly when high accuracy and efficiency are necessary. A novel class of correctors based on feedback-accelerated Picard iteration (FAPI) is proposed to further enhance computational performance. With optimal feedback terms that do not require inversion of matrices, significantly faster convergence speed and higher numerical accuracy are achieved by these correctors compared with their counterparts; however, the computational complexities are comparably low. These advantages enable nonlinear engineering problems to be solved quickly and accurately, even with rough initial guesses from elementary predictors. The proposed method offers flexibility, enabling the use of the generated correctors for either bulk processing of collocation nodes in a domain or successive corrections of a single node in a finite difference approach. In our method, the functional formulas of FAPI are discretized into numerical forms using the collocation approach. These collocated iteration formulas can directly solve nonlinear problems, but they may require significant computational resources because of the manipulation of high-dimensional matrices. To address this, the collocated iteration formulas are further converted into finite difference forms, enabling the design of lightweight predictor-corrector algorithms for real-time computation. The generality of the proposed method is illustrated by deriving new correctors for three commonly employed finite-difference approaches: the modified Euler approach, the Adams-Bashforth-Moulton approach, and the implicit Runge-Kutta approach. Subsequently, the updated approaches are tested in solving strongly nonlinear problems, including the Matthieu equation, the Duffing equation, and the low-earth-orbit tracking problem. The numerical findings confirm the computational accuracy and efficiency of the derived predictor-corrector algorithms.

KEYWORDS

Predictor-corrector method; feedback-accelerated Picard iteration; nonlinear dynamical system; real-time computation

1 Introduction

In real-world engineering tasks and scientific research, predictor-corrector methods are fundamental to numerically solving nonlinear differential equations [1–3]. Currently, there are hundreds

of variants of these methods, which often involve the use of explicit formulas to obtain rough predictions and implicit formulas for further corrections. In general, the predictor-corrector method is widely used in asymptotic and weighted residual methods. This study focuses on finite difference approaches, in which correctors are typically employed once per step. Although multiple corrections can enhance computational accuracy, this approach significantly slows down the algorithm. Therefore, the preference is to reduce the step size rather than repeat corrections. Conventional correctors are often derived from Picard iteration (PI) to solve nonlinear algebraic equations (NAEs) associated with implicit formulas [4,5]. Despite their ease of implementation, these correctors may be insufficient in generating accurate results and ensuring the reliability of long-term simulations for strongly nonlinear dynamical systems. In practical applications, it can result in destructive time delays in tasks that aim for highly accurate results, or it can fail to achieve highly accurate results in real-time computation. Such instances are found in large-scale dynamics simulations [6], autonomous navigation [7], and path-tracking control [8].

This study presents three main contributions. First, an approach to derive fast and accurate correctors to enhance the performance of predictor-corrector methods for more advanced tasks in real life is proposed. This is achieved by replacing PI with feedback-accelerated Picard iteration (FAPI) [9]. These two iteration approaches are categorized as asymptotic approaches that iteratively solve analytical solutions of nonlinear systems starting from an initial guess [10]. The authors in [11] have demonstrated that PI can be considered a special case of FAPI. The latter approach is based on the concept of correcting an approximated solution through the integration of an optimally weighted residual function. FAPI degenerates into PI if the optimal weighting function is specified as a unit matrix instead of being derived from the optimal condition [12]. Therefore, FAPI converges faster and is more stable. Second, since FAPI cannot be employed directly to solve NAEs, the iterative formula is further discretized with appropriate basis functions and collocation nodes [13] to obtain numerical correctors corresponding to PI and FAPI. The PI correctors obtained in this method are equivalent to conventional correctors that directly employ PI to solve implicit finite difference formulas. Third, the proposed approach can be directly applied to other existing predictor-corrector approaches to enhance performance.

The applications of the enhanced predictor-corrector methods and FAPI span various domains, including simulations of n-body problems in celestial mechanics [6], molecular dynamics [14], high-dimensional nonlinear structural dynamics [15], and other problems in fluid mechanics, solid mechanics, heat transfer, and related areas. Particularly, these methods can offer straightforward applications in aerospace engineering for the accurate on-board prediction of spacecraft flight trajectories to achieve autonomous precision navigation [7].

This study is structured as follows. In Section 2, the relationship between PI and FAPI is revealed, and the methodology is introduced by constructing numerical correctors from the discretized collocation formulas of FAPI, which are derived for a general nonlinear dynamical system. Section 3 briefly reviews the underlying relationship between finite difference and collocation, followed by the derivation of FAPI correctors corresponding to three classical finite difference approaches: the modified Euler approach [16], the Adams-Bashforth-Moulton approach [17], and the implicit Runge-Kutta approach [18]. In Section 4, three nonlinear problems (the Matthieu equation [19], the Duffing equation [20], and the low-earth-orbit (LEO) tracking problem [21]) are used as numerical examples to verify the effectiveness of the proposed FAPI correctors.

2 Methodology

2.1 From PI to FAPI

Consider a nonlinear dynamical system in the form of

$$\frac{\mathrm{d}x}{\mathrm{d}t} = g(x,t), \quad t \in [t_0, t_f] \tag{1}$$

Without loss of generality, we suppose $x(t_0) = 0$. Picard iteration method solves this system via

$$x_{n+1}(t) = \int_{t_0}^{t} g(x_n, \tau)\mathrm{d}\tau \tag{2}$$

It can be rewritten as

$$x_{n+1}(t) = x_n(t) - \int_{t_0}^{t} \{\dot{x}_n(\tau) - g(x_n, \tau)\}\,\mathrm{d}\tau \tag{3}$$

where $\dot{x}_n(\tau) - g(x_n, \tau)$ is the residual error of Eq. (1). To make Picard iteration converge faster, we can add a weighting function in the integral of residual error. Then we obtain

$$x_{n+1}(t) = x_n(t) + \int_{t_0}^{t} \lambda(\tau)\,\{\dot{x}_n(\tau) - g(x_n, \tau)\}\,\mathrm{d}\tau \tag{4}$$

Suppose $\Pi[x(\tau), \lambda(\tau)]$ is a vector function of $x(\tau)$ and $\lambda(\tau)$,

$$\Pi[x(t), \lambda(t)] = x(\tau)|_{\tau=t} + \int_{t_0}^{t} \lambda(\tau)\,\{\dot{x}_n(\tau) - g\,(x_n, \tau)\}\,\mathrm{d}\tau \tag{5}$$

Let $\hat{x}(\tau)$ be the exact solution of Eq. (1). Naturally, it satisfies the express

$$\Pi[\hat{x}(\tau), \lambda(t)] = \hat{x}(\tau)\big|_{\tau=t} + \int_{t_0}^{t} \lambda(\tau)\left\{\dot{\hat{x}}(\tau) - g\left(\hat{x}, \tau\right)\right\}\mathrm{d}\tau = \hat{x}(\tau)\big|_{\tau=t} \tag{6}$$

Now we want to make all the components of the function $\Pi[x(t), \lambda(t)]$ stationary about x at $x(t) = \hat{x}(t)$. First, the variation of $\Pi[x(t), \lambda(t)]$ is derived as

$$\begin{aligned}
\delta\Pi[x(t), \lambda(t)] &= \delta x(\tau)|_{\tau=t} + \delta \int_{t_0}^{t} \lambda(\tau)\{\dot{x}(\tau) - g(x, \tau)\}\mathrm{d}\tau \\
&= \delta x(\tau)|_{\tau=t} + \int_{t_0}^{t} \delta\lambda(\tau)\{\dot{x}(\tau) - g(x, \tau)\}\mathrm{d}\tau + \int_{t_0}^{t} \lambda(\tau)\delta\{\dot{x}(\tau) - g(x, \tau)\}\mathrm{d}\tau \\
&= \int_{t_0}^{t} \delta\lambda(\tau)\{\dot{x}(\tau) - g(x, \tau)\}\mathrm{d}\tau + \delta x(\tau)|_{\tau=t} + \lambda(\tau)\delta x(\tau)|_{\tau=t_0}^{\tau=t} \\
&\quad - \int_{t_0}^{t} \left[\frac{\partial\lambda(\tau)}{\partial\tau} + \lambda(\tau)\frac{\partial g(x, \tau)}{\partial x}\right]\delta x(\tau)\mathrm{d}\tau - \int_{t_0}^{t} \lambda(\tau)\frac{\partial g(x, \tau)}{\partial\tau}\delta\tau\mathrm{d}\tau
\end{aligned} \tag{7}$$

Then we collect the terms including $\delta x(\tau)|_{\tau=t}$ and $\delta x(\tau)$,

$$\begin{aligned}&\delta x(\tau)|_{\tau=t}+\lambda(\tau)\delta x(\tau)|_{\tau=t},\\&\int_{t_0}^{t}\left[\frac{\partial\lambda(\tau)}{\partial\tau}+\lambda(\tau)\frac{\partial g(x,\tau)}{\partial x}\right]\delta x(\tau)\mathrm{d}\tau\end{aligned}\tag{8}$$

Note that the boundary value of $x(\tau)$ at $\tau=t_0$ is prescribed, that is to say $\delta x(\tau)|_{\tau=t_0}=0$. Thus, the stationary condition for $\Pi[x(\tau),\lambda(\tau)]$ is obtained as

$$\begin{cases}I+\lambda(\tau)|_{\tau=t}=0\\\dfrac{\partial\lambda}{\partial\tau}+\lambda(\tau)J(\tau)=0\end{cases}\tag{9}$$

from which we can derive the first order appoximation of λ.

$$\lambda\approx -I+J(t)(\tau-t)\tag{10}$$

where J is the Jacobian matrix of $g(x,\tau)$ w.r.t x, and I is an identity matrix. In Picard iteration, the stationarity conditions cannot be satisfied and thus $\delta\Pi[x(t),\lambda(t)]=O(\delta x)$. However, in FAPI, since the stationarity conditions can be satisfied, then

$$\delta x(\tau)|_{\tau=t}+\lambda(\tau)\delta x(\tau)|_{\tau=t_0}^{\tau=t}-\int_{t_0}^{t}\left[\frac{\partial\lambda(\tau)}{\partial\tau}+\lambda(\tau)\frac{\partial g(x,\tau)}{\partial x}\right]\delta x(\tau)\mathrm{d}\tau=0\tag{11}$$

which means the error caused will not exceed $O^2(\delta x)$. Therefore, we can conclude that the newly proposed method based on FAPI is more accurate than the existing methods based on PI, and to reach the given tolerance, the former needs fewer iterations.

By substituting Eq. (10) into Eq. (4), the correctional formula becomes

$$x_{n+1}(t)=x_n(t)-\{I+J(t)t\}\int_{t_0}^{t}G(\tau)\mathrm{d}\tau+J(t)\int_{t_0}^{t}\tau G(\tau)\mathrm{d}\tau\tag{12}$$

where $G(\tau)=\dot{x}_n(\tau)-g(x_n,\tau)$. Note that both J and G depend on x_n. Since Eq. (12) involves feedback terms of integration of residual errors that accelerate the convergence of Picard iteration, it is named as Feedback-Accelerated Picard Iteration (FAPI).

Previously, the formula of FAPI takes the form of Eq. (12). In this work, we propose another form of it. By differentiating Eq. (4), we have

$$\begin{aligned}\frac{\mathrm{d}x_{n+1}(t)}{\mathrm{d}t}=&\frac{\mathrm{d}x_n(t)}{\mathrm{d}t}+\lambda(\tau)|_{\tau=t}\{\dot{x}_n(\tau)|_{\tau=t}-g(x_n,\tau)|_{\tau=t}\}\\&+\int_{t_0}^{t}\frac{\partial\lambda(\tau)}{\partial t}\{\dot{x}_n(\tau)-g(x_n,\tau)\}\,\mathrm{d}\tau\end{aligned}\tag{13}$$

with Eqs. (9), (13) can be rewritten as

$$\frac{\mathrm{d}x_{n+1}(t)}{\mathrm{d}t}=g(x_n,t)+J(t)\int_{t_0}^{t}\lambda(\tau)\{\dot{x}_n(\tau)-g(x_n,\tau)\}\,\mathrm{d}\tau\tag{14}$$

Simply approximating $\lambda(\tau)$ as $-\boldsymbol{I}$, which is its zeroth order approximation, Eq. (14) becomes

$$\frac{\mathrm{d}x_{n+1}(t)}{\mathrm{d}t}=g(x_n,t)-J(t)\left[x_n(t)-x(t_0)-\int_{t_0}^{t}g(x_n,\tau)\mathrm{d}\tau\right]\tag{15}$$

By integrating both sides of Eq. (15), we have

$$x_{n+1}(t) = x(t_0) + \int_{t_0}^{t} \left\{ g(x_n, \tau) - J(\tau) \left[x_n(\tau) - x(t_0) - \int_{t_0}^{\tau} g(x_n, \xi) \mathrm{d}\xi \right] \right\} \mathrm{d}\tau \tag{16}$$

For clarity, the two versions of FAPI fomulae are listed in Table 1.

Table 1: The two versions of FAPI formula

Versions	Formula
1st	$x_{n+1}(t) = x_n(t) - \{I + J(t)t\} \int_{t_0}^{t} G(\tau)\mathrm{d}\tau + J(t) \int_{t_0}^{t} \tau G(\tau)\tau$
2nd	$x_{n+1}(t) = x(t_0) + \int_{t_0}^{t} \left\{ g(x_n, \tau) - J(\tau) \left[x_n(\tau) - x(t_0) - \int_{t_0}^{\tau} g(x_n, \xi) \mathrm{d}\xi \right] \right\} \mathrm{d}\tau$

2.2 Collocated FAPI

2.2.1 The 1st Version of Collocated FAPI

By collocating M nodes in time domain, Eq. (12) can be discretized. It is written as

$$\begin{bmatrix} x_{n+1}(t_1) \\ x_{n+1}(t_2) \\ \vdots \\ x_{n+1}(t_M) \end{bmatrix} = \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} - \left(\tilde{I} + \tilde{J}\tilde{T} \right) \begin{bmatrix} \int_{t_0}^{t_1} G(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} G(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} G(\tau)\mathrm{d}\tau \end{bmatrix} + \tilde{J} \begin{bmatrix} \int_{t_0}^{t_1} \tau G(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \tau G(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \tau G(\tau)\mathrm{d}\tau \end{bmatrix} \tag{17}$$

The block diagonal matrices $\tilde{I}$, $\tilde{J}$, $\tilde{T}$ are defined as

$$\tilde{I} = \begin{bmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix}, \tilde{J} = \begin{bmatrix} J(t_1) & & & \\ & J(t_2) & & \\ & & \ddots & \\ & & & J(t_M) \end{bmatrix}, \tilde{T} = \begin{bmatrix} t_1 I & & & \\ & t_2 I & & \\ & & \ddots & \\ & & & t_M I \end{bmatrix} \tag{18}$$

We suppose that $x(t) = [x_1(t), x_2(t), ..., x_d(t), \ldots, x_D(t)]^{\mathrm{T}}$ is a D-dimensional vector of time-varying variables, and that the variables $x_d(t)$ are approximated by the linear combinations of N orthogonal basis functions $\phi_{d,nb}(t)$,

$$x_d(t) = \sum_{nb=1}^{N} \alpha_{d,nb}\phi_{nb}(t) = \Phi(t)A_d \tag{19}$$

Using the approximation in Eq. (19), it is obtained that

$$\begin{bmatrix} G(t_1) \\ G(t_2) \\ \vdots \\ G(t_M) \end{bmatrix} = \begin{bmatrix} \dot{x}_n(t_1) \\ \dot{x}_n(t_2) \\ \vdots \\ \dot{x}_n(t_M) \end{bmatrix} - \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} = \tilde{R}\tilde{Q}\tilde{R}^{-1} \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} - \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \tag{20}$$

where $\tilde{R}$ is a row-rearranging-matrix. The differentiation matrix $\tilde{Q}$ is defined as

$$\tilde{Q} = \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & Q \end{bmatrix}, Q = \begin{bmatrix} \dot{\Phi}(t_1) \\ \dot{\Phi}(t_2) \\ \vdots \\ \dot{\Phi}(t_M) \end{bmatrix} \begin{bmatrix} \Phi(t_1) \\ \Phi(t_2) \\ \vdots \\ \Phi(t_M) \end{bmatrix}^{-1} \tag{21}$$

Further, by making the basis functions in Eq. (19) universal, it is found that

$$\begin{bmatrix} \int_{t_0}^{t_1} G(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} G(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} G(\tau)\mathrm{d}\tau \end{bmatrix} = \tilde{R}\tilde{P}\tilde{R}^{-1} \begin{bmatrix} G(t_1) \\ G(t_2) \\ \vdots \\ G(t_M) \end{bmatrix} \tag{22}$$

where the integration matrix $\tilde{P}$ is defined as

$$\tilde{P} = \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \end{bmatrix}, P = \begin{bmatrix} \int_{t_0}^{t_1} \Phi(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \Phi(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \Phi(\tau)\mathrm{d}\tau \end{bmatrix} \begin{bmatrix} \Phi(t_1) \\ \Phi(t_2) \\ \vdots \\ \Phi(t_M) \end{bmatrix}^{-1} \tag{23}$$

For simplicity, we denote $\tilde{Q} = \tilde{R}\tilde{Q}\tilde{R}^{-1}$ and $\tilde{P} = \tilde{R}\tilde{P}\tilde{R}^{-1}$. In our previous work, we simply let

$$\begin{bmatrix} \int_{t_0}^{t_1} \tau G(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \tau G(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \tau G(\tau)\mathrm{d}\tau \end{bmatrix} = \tilde{R}\tilde{P}\tilde{R}^{-1} \begin{bmatrix} t_1 G(t_1) \\ t_2 G(t_2) \\ \vdots \\ t_M G(t_M) \end{bmatrix} \tag{24}$$

However, it may cause deterioration of the numerical solution, because $\tau G(\tau)$ is approximated with the same basis functions of $G(\tau)$. It means that the approximation of $\tau G(\tau)$ is one order lower than it should be. A more rigorous approximation is made as following:

$$\begin{bmatrix} \int_{t_0}^{t_1} \tau G(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \tau G(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \tau G(\tau)\mathrm{d}\tau \end{bmatrix} = \tilde{R}\tilde{P}_\tau\tilde{R}^{-1} \begin{bmatrix} G(t_1) \\ G(t_2) \\ \vdots \\ G(t_M) \end{bmatrix} \tag{25}$$

where the modified integration matrix $\tilde{P}_\tau$ is derived as

$$\tilde{P}_\tau = \begin{bmatrix} P_\tau & & & \\ & P_\tau & & \\ & & \ddots & \\ & & & P_\tau \end{bmatrix}, P_\tau = \begin{bmatrix} \int_{t_0}^{t_1} \tau\Phi(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \tau\Phi(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \tau\Phi(\tau)\mathrm{d}\tau \end{bmatrix} \begin{bmatrix} \Phi(t_1) \\ \Phi(t_2) \\ \vdots \\ \Phi(t_M) \end{bmatrix}^{-1} \tag{26}$$

Denoting $\tilde{P}_\tau = \tilde{R}\tilde{P}_\tau\tilde{R}$, and substituting Eqs. (20), (22), and (25) into Eq. (17), it can be found that

$$\begin{bmatrix} x_{n+1}(t_1) \\ x_{n+1}(t_2) \\ \vdots \\ x_{n+1}(t_M) \end{bmatrix} = \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} + \left\{ \tilde{J}\left(\tilde{P}_\tau - \tilde{T}\tilde{P}\right) - \tilde{P} \right\} \left\{ \tilde{Q} \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} - \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \right\} \tag{27}$$

We further denote the constant matrix $\tilde{P}_\tau - \tilde{T}\tilde{P}$ as $\tilde{H}$, thus Eq. (27) becomes

$$\tilde{x}_{n+1} = \tilde{x}_n + (\tilde{J}\tilde{H} - \tilde{P})(\tilde{Q}\tilde{x}_n - \tilde{g}_n) \tag{28}$$

where $\tilde{x} = [x(t_1), x(t_2), ..., x(t_M)]^{\mathrm{T}}$, $\tilde{g} = [g(t_1), g(t_2), ..., g(t_M)]^{\mathrm{T}}$. Denote $\tilde{J}\tilde{H} - \tilde{P}$ as $\tilde{J}_{HP}$, the algorithm is further simplified as

$$\begin{bmatrix} x_{n+1}(t_1) \\ x_{n+1}(t_2) \\ \vdots \\ x_{n+1}(t_M) \end{bmatrix} = \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} + \tilde{J}_{HP} \left\{ \tilde{Q} \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} - \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \right\} \tag{29}$$

2.2.2 The 2nd Version of Collocated FAPI

Further numerical discretization is made to the collocation form of Eq. (15). It is expressed as

$$\begin{bmatrix} \dot{x}_{n+1}(t_1) \\ \dot{x}_{n+1}(t_2) \\ \vdots \\ \dot{x}_{n+1}(t_M) \end{bmatrix} = \begin{bmatrix} g(x_n, t_1) \\ g(x_n, t_2) \\ \vdots \\ g(x_n, t_M) \end{bmatrix} - \tilde{J} \left\{ \begin{bmatrix} x_n(t_1) \\ x_n(t_2) \\ \vdots \\ x_n(t_M) \end{bmatrix} - \begin{bmatrix} x(t_0) \\ x(t_0) \\ \vdots \\ x(t_0) \end{bmatrix} - \begin{bmatrix} \int_{t_0}^{t_1} g \mathrm{d}\tau \\ \int_{t_0}^{t_2} g \mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} g \mathrm{d}\tau \end{bmatrix} \right\} \tag{30}$$

In Eq. (30), $g(x_n, \tau)$ is denoted as g for simplicity. As aforementioned, the block diagonal matrices $\tilde{I}, \tilde{J}, \tilde{T}$ are defined as

$$\tilde{I} = \begin{bmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix}, \tilde{J} = \begin{bmatrix} J(t_1) & & & \\ & J(t_2) & & \\ & & \ddots & \\ & & & J(t_M) \end{bmatrix}, \tilde{T} = \begin{bmatrix} t_1 I & & & \\ & t_2 I & & \\ & & \ddots & \\ & & & t_M I \end{bmatrix} \tag{31}$$

We suppose that $\dot{x}(t) = [\dot{x}_1(t), \dot{x}_2(t), ..., \dot{x}_d(t), \ldots, \dot{x}_D(t)]^{\mathrm{T}}$ is a D-dimensional vector of time-varying variables. The components $\dot{x}_d(t)$ are approximated by the linear combinations of N orthogonal basis functions $\phi_{d,nb}(t)$,

$$\dot{x}_d(t) = \sum_{nb=1}^{N} \alpha_{d,nb}\phi_{nb}(t) = \Phi(t)A_d \tag{32}$$

Similar approximations can be made to the components $g_d(x, t)$ of $g(x, t)$, i.e.,

$$g_d(x, t) = \sum_{nb=1}^{N} \beta_{d,nb}\phi_{nb}(t) = \Phi(t)B_d \tag{33}$$

Using the approximations in Eqs. (32), (33), it is obtained that

$$\begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_M) \end{bmatrix} = \tilde{R}\tilde{P}\tilde{R}^{-1} \begin{bmatrix} \dot{x}(t_1) \\ \dot{x}(t_2) \\ \vdots \\ \dot{x}(t_M) \end{bmatrix} + \begin{bmatrix} x(t_0) \\ x(t_0) \\ \vdots \\ x(t_0) \end{bmatrix} \tag{34}$$

and

$$\begin{bmatrix} \int_{t_0}^{t_1} g(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} g(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} g(\tau)\mathrm{d}\tau \end{bmatrix} = \tilde{R}\tilde{P}\tilde{R}^{-1} \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \tag{35}$$

where the integration matrix $\tilde{P}$ is defined as

$$\tilde{P} = \begin{bmatrix} P & & & \\ & P & & \\ & & \ddots & \\ & & & P \end{bmatrix}, P = \begin{bmatrix} \int_{t_0}^{t_1} \Phi(\tau)\mathrm{d}\tau \\ \int_{t_0}^{t_2} \Phi(\tau)\mathrm{d}\tau \\ \vdots \\ \int_{t_0}^{t_M} \Phi(\tau)\mathrm{d}\tau \end{bmatrix} \begin{bmatrix} \Phi(t_1) \\ \Phi(t_2) \\ \vdots \\ \Phi(t_M) \end{bmatrix}^{-1} \tag{36}$$

For simplicity, we denote $\tilde{P} = \tilde{R}\tilde{P}\tilde{R}^{-1}$. By substituting Eqs. (30), (35) into Eq. (34), it can be found that

$$\overbrace{\tilde{x}_{n+1} = \tilde{x}(t_0) + \tilde{P}\tilde{g}_n}^{Picard\ iteration} \underbrace{-\tilde{P}\tilde{J}\left\{\tilde{x}_n - \tilde{x}(t_0) - \tilde{P}\tilde{g}_n\right\}}_{Acceleration\ terms} \tag{37}$$

where $\tilde{x} = [x(t_1), x(t_2), ..., x(t_M)]^{\mathrm{T}}$, $\tilde{x}(t_0) = [x(t_0), x(t_0), ..., x(t_0]^{\mathrm{T}}$, and $\tilde{g} = [g(t_1), g(t_2), ..., g(t_M)]^{\mathrm{T}}$. It will be shown later that the Jacobian matrix $\tilde{J}$ can be regarded as constant in many cases, without harming the convergence of algorithm.

For clarity, the two versions of collocated FAPI formulae are displayed in Table 2.

Table 2: The two versions of collocated FAPI formula

Versions	Formula
1st	$\tilde{x}_{n+1} = \tilde{x}_n + (\tilde{J}\tilde{H} - \tilde{P})(\tilde{Q}\tilde{x}_n - \tilde{g}_n)$
2nd	$\tilde{x}_{n+1} = \tilde{x}(t_0) + \tilde{P}\tilde{g}_n - \tilde{P}\tilde{J}\left\{\tilde{x}_n - \tilde{x}(t_0) - \tilde{P}\tilde{g}_n\right\}$

2.3 FAPI as Numerical Corrector

Each row of Eqs. (29) or (37) can be used independently as a numerical corrector. For instance, the last row of Eq. (37) can be written as

$$x_{n+1}(t_M) = x(t_0) + \tilde{P}^M \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} - \tilde{P}^M \tilde{J} \left\{ \begin{bmatrix} x_n(t_1) - x(t_0) \\ x_n(t_2) - x(t_0) \\ \vdots \\ x_n(t_M) - x(t_0) \end{bmatrix} - \tilde{P} \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \right\} \tag{38}$$

where the superscript M of matrices $\tilde{P}^M$ denotes the $(D(M-1)+1)$-th to (DM)-th row matrix of $\tilde{P}$ that are related with the collocation point t_M.

Given the values of collocation nodes $x(t_1)$, $x(t_2)$, …, $x(t_{M-1})$ being settled, Eq. (38) can be used to update the value of collocation node $x(t_M)$. Similarly, an unknown collocation node $x(t_m)$ can be updated using the $(D(m-1)+1)$-th to (Dm)-th row of Eq. (37) once the values of the other collocation nodes are settled. The methodology of FAPI corrector is illustrated in Fig. 1.

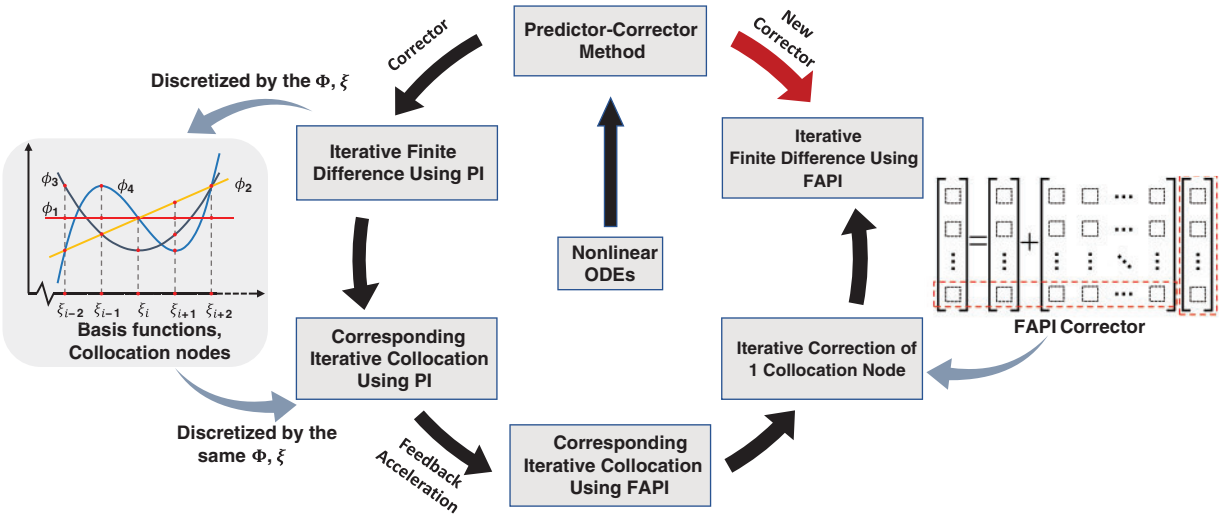


Figure 1: The process to derive FAPI corrector for predictor-corrector method when dealing general nonlinear ODEs

3 Finite Difference Integrators Featured by FAPI

To apply FAPI as numerical corrector in finite difference method, we need to use the same collocation nodes and basis functions underlying the finite difference discretization. The relationship between finite difference and collocation is briefly reviewed herein.

3.1 From Collocation to Finite Difference

The collocation form of Eq. (1) can be written as

$$\tilde{Q} \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_M) \end{bmatrix} = \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} \tag{39}$$

where $\tilde{Q}$ is the rearranged differentiation matrix. The $(D(m-1)+1)$-th to (Dm)-th row of Eq. (39) is

$$\tilde{Q}^m \begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_M) \end{bmatrix} = g(t_m) \tag{40}$$

Eq. (40) is a finite difference form of Eq. (1). If $m \neq M$, it is an explicit finite difference formula, since $x(t_M)$ can be explicitly obtained using Eq. (40) once the other values of collocation points are known. If $m = M$, it becomes an implicit formula.

Besides, Eq. (1) can be equivalently expressed as

$$x(t) = \int_{t_0}^{t} g(x,\tau)\mathrm{d}\tau + x(t_0) \tag{41}$$

The collocation form of Eq. (41) is

$$\begin{bmatrix} x(t_1) \\ x(t_2) \\ \vdots \\ x(t_M) \end{bmatrix} = \tilde{P} \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} + \begin{bmatrix} x(t_0) \\ x(t_0) \\ \vdots \\ x(t_0) \end{bmatrix} \tag{42}$$

where $\tilde{P}$ is the rearranged integration matrix. The $(D(m-1)+1)$-th to (Dm)-th row of Eq. (42) is

$$x(t_m) = \tilde{P}^m \begin{bmatrix} g(t_1) \\ g(t_2) \\ \vdots \\ g(t_M) \end{bmatrix} + x(t_0) \tag{43}$$

It is an implicit finite difference formula. Usually, we let $m = M$ in Eq. (43) to solve for $x(t_M)$.

Since Eqs. (43) and (40) are obtained by numerical differentiation and integration, respectively, using collocation method, we can find the counterpart of a finite difference method as a collocation method by choosing proper basis functions and collocation nodes, or derive a finite difference formula from collocation method.

3.2 Modified Euler Method Using FAPI

Herein, we consider the Modified Euler method (MEM) that only uses 2 nodes, of which one is known and the other is unknown. The collocation counterpart of this method has 2 collocation nodes, and takes the Lagrange polynomials $\phi_0(t) = (t-t_2)/(t_1-t_2)$ and $\phi_1(t) = (t-t_1)/(t_2-t_1)$ (or simply $\phi_0(t) = 1$ and $\phi_1(t) = t$)) as basis functions. From Eq. (21), we have

$$Q = \begin{bmatrix} \dot{\Phi}(t_1) \\ \dot{\Phi}(t_2) \end{bmatrix} \begin{bmatrix} \Phi(t_1) \\ \Phi(t_2) \end{bmatrix}^{-1} = \frac{1}{t_2 - t_1} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \tag{44}$$

and

$$\tilde{Q} = \tilde{R}\tilde{Q}\tilde{R}^{-1} = \frac{1}{t_2 - t_1} \begin{bmatrix} -I & I \\ -I & I \end{bmatrix} \tag{45}$$

By substituting Eq. (45) into Eq. (39), we have

$$\frac{1}{t_2-t_1}\begin{bmatrix}-I & I\\ -I & I\end{bmatrix}\begin{bmatrix}x(t_1)\\ x(t_2)\end{bmatrix}=\begin{bmatrix}g(t_1)\\ g(t_2)\end{bmatrix} \tag{46}$$

The first row of Eq. (46) is an explicit formula of Euler method, while the second row is implicit. They can be used independently or conjointly in a predictor-corrector manner. In the later case, the explicit formula provides prediction for the implicit formula to make further correction. Usually, the correction is made by the Picard iteration method due to its low computational complexity and inversion-free property. Based on Eq. (46), a predictor-corrector formula using Picard iteration can thus be expressed as

$$\frac{1}{t_2-t_1}\begin{bmatrix}-I & I\end{bmatrix}\begin{bmatrix}x(t_1) & x_0(t_2)\end{bmatrix}^{\mathrm{T}}=g(t_1) \tag{47}$$

$$\frac{1}{t_2-t_1}\begin{bmatrix}-I & I\end{bmatrix}\begin{bmatrix}x(t_1) & x_{n+1}(t_2)\end{bmatrix}^{\mathrm{T}}=g_n(t_2) \tag{48}$$

where x_{n+1} and g_n, $n=0,1,2,\ldots$, stand for the $(n+1)$-th corrected x and n-th corrected g, respectively. Usually n is simply set as 0, implying the correctional formula is only executed for once.

However, the modified Euler method utilizes the trapezoidal formula instead of the implicit Euler formula to make correction. The trapezoidal formula can be derived from Eq. (42). By using the same collocation nodes and basis functions that lead to Eq. (46), we have

$$P=\begin{bmatrix}\int_{t_1}^{t_1}\Phi(\tau)\mathrm{d}\tau\\ \int_{t_1}^{t_2}\Phi(\tau)\mathrm{d}\tau\end{bmatrix}\begin{bmatrix}\Phi(t_1)\\ \Phi(t_2)\end{bmatrix}^{-1}=\frac{t_2-t_1}{2}\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix} \tag{49}$$

and

$$\tilde{P}=\tilde{R}\tilde{P}\tilde{R}^{-1}=\frac{t_2-t_1}{2}\begin{bmatrix}0 & 0\\ I & I\end{bmatrix} \tag{50}$$

Note that we start the integration at t_1 for simplicity. By substituting Eq. (50) into Eq. (42), we have

$$\begin{bmatrix}x(t_1)\\ x(t_2)\end{bmatrix}=\frac{t_2-t_1}{2}\begin{bmatrix}0 & 0\\ I & I\end{bmatrix}\begin{bmatrix}g(t_1)\\ g(t_2)\end{bmatrix}+\begin{bmatrix}x(t_1)\\ x(t_1)\end{bmatrix} \tag{51}$$

The second row of Eq. (51) is the formula of trapezoidal method. Given $x(t_1)$, $x(t_2)$ can be solved with Picard iteration, which is

$$x_{n+1}(t_2)=\frac{t_2-t_1}{2}\begin{bmatrix}I & I\end{bmatrix}\begin{bmatrix}g(t_1) & g_n(t_2)\end{bmatrix}^{\mathrm{T}}+x(t_1) \tag{52}$$

Eqs. (47) and (52) constitute the modified Euler method.

3.2.1 The 1st Version of ME-FAPI Method

Now we replace Eq. (52) with the feedback-accelerated Picard iteration. The necessary matrices are derived first as following:

$$\tilde{J}=\begin{bmatrix}J(t_1) & \\ & J(t_2)\end{bmatrix},\ \tilde{T}=\begin{bmatrix}t_1I & \\ & t_2I\end{bmatrix},\ \tilde{P}_\tau=\frac{t_2-t_1}{6}\begin{bmatrix}0 & 0\\ (2t_1+t_2)I & (2t_2+t_1)I\end{bmatrix} \tag{53}$$

With Eqs. (50) and (53), $\tilde{H}$ is derived as

$$\tilde{H} = \tilde{P}_{\tau} - \tilde{T}\tilde{P} = -\frac{(t_2 - t_1)^2}{6}\begin{bmatrix} 0 & 0 \\ 2I & I \end{bmatrix} \tag{54}$$

Thus

$$\tilde{J}_{HP} = \tilde{J}\tilde{H} - \tilde{P} = -\frac{(t_2 - t_1)^2}{6}\begin{bmatrix} 0 & 0 \\ 2J(t_2) & J(t_2) \end{bmatrix} - \frac{t_2 - t_1}{2}\begin{bmatrix} 0 & 0 \\ I & I \end{bmatrix} \tag{55}$$

According to Eq. (29), we have

$$x_{n+1}(t_2) = x_n(t_2) + \frac{1}{t_2 - t_1}\tilde{J}_{HP}^{(2)}\begin{bmatrix} x_n(t_2) - x(t_1) - (t_2 - t_1)g(t_1) \\ x_n(t_2) - x(t_1) - (t_2 - t_1)g_n(t_2) \end{bmatrix} \tag{56}$$

It is rewritten as

$$\overbrace{x_{n+1}(t_2) = x(t_1) + \frac{h}{2}[g(t_1) + g_n(t_2)]}^{Picard\ \ iteration} \underbrace{-\frac{h^2}{6}J(t_2)\left\{\frac{3}{h}[x_{n-1}(t_2) - x(t_1)] - 2g(t_1) - g_n(t_2)\right\}}_{Acceleration\ \ terms} \tag{57}$$

where h is the interval between t_1 and t_2.

3.2.2 The 2nd Version of ME-FAPI Method

According to Eq. (37), we have

$$x_{n+1}(t_2) = x(t_1) + \tilde{P}^{(2)}\begin{bmatrix} g(t_1) \\ g(t_2) \end{bmatrix} - \tilde{P}^{(2)}\tilde{J}\left\{\begin{bmatrix} x_n(t_1) - x(t_1) \\ x_n(t_2) - x(t_1) \end{bmatrix} - \tilde{P}\begin{bmatrix} g(t_1) \\ g(t_2) \end{bmatrix}\right\} \tag{58}$$

It is rewritten as

$$\overbrace{x_{n+1}(t_2) = x(t_1) + \frac{h}{2}\{g(t_1) + g_n(t_2)\}}^{Picard\ \ iteration} \underbrace{-\frac{h}{2}J(t_2)\left\{x_n(t_2) - x(t_1) - \frac{h}{2}[g(t_1) + g_n(t_2)]\right\}}_{Acceleration\ \ terms} \tag{59}$$

where h is the interval between t_1 and t_2. By comparing with Eq. (52), we can see that an acceleration term related with the Jacobian matrix is added at the right hand side.

3.3 Adams-Bashforth-Moulton Method Using FAPI

Taking the 4-th order Adams-Bashforth-Moulton (ABM) method for instance, it solves Eq. (1) with the following formula:

$$x_0(t_5) = x(t_4) + \frac{h}{24}[55g(t_4) - 59g(t_3) + 37g(t_2) - 9g(t_1)] \tag{60}$$

$$x_n(t_5) = x(t_4) + \frac{h}{24}[9g_{n-1}(t_5) + 19g(t_4) - 5g(t_3) + g(t_2)] \tag{61}$$

of which Eq. (60) is a predictor using an Adams-Bashforth formula and Eq. (61) is a corrector that uses Picard iteration to solve an Adams-Moulton formula.

3.3.1 The 1st Version of ABM-FAPI Method

To replace the Picard iteration with Feedback-Accelerated Picard iteration in the Adams-Moulton corrector (Eq. (61)), we only need to determine the differentiation matrix $\tilde{Q}$, the integration matrix $\tilde{P}$, the Jacobian matrix $\tilde{J}$, and matrix $\tilde{T}$. Those matrices simply depend on the collocation points $t_2, ..., t_5$ and the basis functions of the collocation counterpart of Adams-Moulton formula. Let the basis functions used for interpolation in the Adams-Moulton formula be the Lagrange polynomials.

$$\phi_{nb}(t) = \prod_{2\le j\le 5, nb\ne j} \frac{t - t_j}{t_{nb} - t_j} \tag{62}$$

With Eq. (62), the differentiation matrix $\tilde{Q}$ is derived as

$$\tilde{Q} = \frac{1}{6h}\begin{bmatrix} -11 & 18 & -9 & 2 \\ -2 & -3 & 6 & -1 \\ 1 & -6 & 3 & 2 \\ -2 & 9 & -18 & 11 \end{bmatrix} \otimes I \tag{63}$$

To keep consistent with the Adams-Moulton formula underlying Eq. (61), we let the integration in Eq. (12) start from t_4. Thus, the matrices P is written as

$$P = \begin{bmatrix} \int_{t_4}^{t_2} \Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_3} \Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_4} \Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_5} \Phi(\tau)\mathrm{d}\tau \end{bmatrix} \begin{bmatrix} \Phi(t_2) \\ \Phi(t_3) \\ \Phi(t_4) \\ \Phi(t_5) \end{bmatrix}^{-1} \tag{64}$$

and

$$P_\tau = \begin{bmatrix} \int_{t_4}^{t_2} \tau\Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_3} \tau\Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_4} \tau\Phi(\tau)\mathrm{d}\tau \\ \int_{t_4}^{t_5} \tau\Phi(\tau)\mathrm{d}\tau \end{bmatrix} \begin{bmatrix} \Phi(t_2) \\ \Phi(t_3) \\ \Phi(t_4) \\ \Phi(t_5) \end{bmatrix}^{-1} \tag{65}$$

They are further expressed as

$$\tilde{P} = \frac{h}{24}\begin{bmatrix} -8 & -32 & -8 & 0 \\ 1 & -13 & -13 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & -5 & 19 & 9 \end{bmatrix} \otimes I \tag{66}$$

and

$$\tilde{P}_\tau = \frac{h^2}{360}\begin{bmatrix} -16 & -432 & -288 & 16 \\ 22 & -261 & -324 & 23 \\ 0 & 0 & 0 & 0 \\ 38 & -189 & 684 & 367 \end{bmatrix} \otimes I + t_2\tilde{P} \tag{67}$$

Then the matrix $\tilde{H}$ is derived as

$$\tilde{H} = \tilde{P}_{\tau} - \tilde{T}\tilde{P} = \frac{h^2}{360}\begin{bmatrix} -16 & -432 & -288 & 16 \\ 7 & -66 & -129 & 8 \\ 0 & 0 & 0 & 0 \\ -7 & 36 & -171 & -38 \end{bmatrix} \otimes I \tag{68}$$

The matrix $\tilde{J}_{HP}^{(4)} = \tilde{J}^{(4)}\tilde{H} - \tilde{P}^{(4)}$ is

$$\tilde{J}_{HP}^{(4)} = \frac{h^2}{360}\begin{bmatrix} -7 & 36 & -171 & -38 \end{bmatrix} \otimes J(t_5) - \frac{h}{24}\begin{bmatrix} 1 & -5 & 19 & 9 \end{bmatrix} \otimes I \tag{69}$$

By substituting Eqs. (63) and (69) into Eq. (29), we have

$$x_{n+1}(t_5) = x_n(t_5) + \tilde{J}_{HP}^{(4)}\left\{\tilde{Q}\begin{bmatrix} x(t_2) \\ x(t_3) \\ x(t_4) \\ x_n(t_5) \end{bmatrix} - \begin{bmatrix} g(t_2) \\ g(t_3) \\ g(t_4) \\ g_n(t_5) \end{bmatrix}\right\} \tag{70}$$

It is explicitly expressed as

$$\begin{aligned} x_{n+1}(t_5) = {} & x(t_4) + \frac{h}{24}[9g_{n-1}(t_5) + 19g(t_4) - 5g(t_3) + g(t_2)] \\ & + \frac{h^2}{360}J(t_5)\{\frac{-90x(t_2) + 450x(t_3) + 450x(t_4) - 810x_n(t_5)}{6h} \\ & + 7g(t_2) - 36g(t_3) + 171g(t_4) + 38g_n(t_5)\} \end{aligned} \tag{71}$$

where the terms related with $J(t_5)$ aid to accelerate the convergence of Picard iteration of the Adams-Moulton formula in Eq. (61).

3.3.2 *The 2nd Version of ABM-FAPI Method*

According to Eq. (37), we have

$$x_{n+1}(t_5) = x(t_4) + \tilde{P}^{(4)}\begin{bmatrix} g(t_2) \\ g(t_3) \\ g(t_4) \\ g(t_5) \end{bmatrix} - \tilde{P}^{(4)}\tilde{J}\left\{\begin{bmatrix} x_n(t_2) - x(t_4) \\ x_n(t_3) - x(t_4) \\ x_n(t_4) - x(t_4) \\ x_n(t_5) - x(t_4) \end{bmatrix} - \tilde{P}\begin{bmatrix} g(t_2) \\ g(t_3) \\ g(t_4) \\ g(t_5) \end{bmatrix}\right\} \tag{72}$$

It is rewritten as

$$\begin{aligned} x_{n+1}(t_5) = {} & x(t_4) + \frac{h}{24}[9g_{n-1}(t_5) + 19g(t_4) - 5g(t_3) + g(t_2)] - \frac{h}{24} \\ & \left\{9J(t_5)\left[x_n(t_5) - x(t_4) - \frac{h(9g_{n-1}(t_5) + 19g(t_4) - 5g(t_3) + g(t_2))}{24}\right]\right. \\ & - 5J(t_3)\left[x(t_3) - x(t_4) - \frac{h(g_{n-1}(t_5) - 13g(t_4) - 13g(t_3) + g(t_2))}{24}\right] \\ & \left. + J(t_2)\left[x(t_2) - x(t_4) - \frac{h(-8g(t_4) - 32g(t_3) - 8g(t_2))}{24}\right]\right\} \end{aligned} \tag{73}$$

3.4 Implicit Runge-Kutta Method Using FAPI

The equivalence between implicit Runge-Kutta (IRK) method and collocation method has been well discussed and proved [22]. For that, the collocated FAPI provides iterative formulae for implicit Runge-Kutta methods. A 4-stage implicit Runge-Kutta method is taken as an example, where its corresponding collocation method takes the first kind of Chebyshev polynomials and Chebyshev-Gauss-Lobatto (CGL) nodes as basis functions and collocation nodes, respectively.

By re-scaling the time segment as $[-1, 1]$, the first kind of Chebyshev polynomials are obtained from the recurrence relation

$$\phi_0(\xi) = 1,\ \phi_1(\xi) = t,\ \phi_{n+1}(\xi) = 2t\phi_n(\xi) - \phi_{n-1}(\xi) \tag{74}$$

where ξ is the rescaled time variable. The CGL nodes are $\xi_1 = -1, \xi_2 = -1/2, \xi_3 = 1/2, \xi_4 = 1$. The differentiation matrix $\tilde{Q}$ is obtained as

$$\tilde{Q} = \begin{bmatrix} 0 & 1 & 4\xi_1 & 12\xi_1^2 - 3 \\ 0 & 1 & 4\xi_2 & 12\xi_2^2 - 3 \\ 0 & 1 & 4\xi_3 & 12\xi_3^2 - 3 \\ 0 & 1 & 4\xi_4 & 12\xi_4^2 - 3 \end{bmatrix} \begin{bmatrix} 1 & \xi_1 & 2\xi_1^2 - 1 & 4\xi_1^3 - 3\xi_1 \\ 1 & \xi_2 & 2\xi_2^2 - 1 & 4\xi_2^3 - 3\xi_2 \\ 1 & \xi_3 & 2\xi_3^2 - 1 & 4\xi_3^3 - 3\xi_3 \\ 1 & \xi_4 & 2\xi_4^2 - 1 & 4\xi_4^3 - 3\xi_4 \end{bmatrix}^{-1} \otimes I \tag{75}$$

From Eqs. (23) and (26), the integration matrices $\tilde{P}$ and $\tilde{P}_\tau$ are

$$\tilde{P} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \xi_2 - \xi_1 & \dfrac{\xi_2^2 - \xi_1^2}{2} & \dfrac{2(\xi_2^3 - \xi_1^3)}{3} - (\xi_2 - \xi_1) & \xi_2^4 - \xi_1^4 - \dfrac{3(\xi_2^2 - \xi_1^2)}{2} \\ \xi_3 - \xi_1 & \dfrac{\xi_3^2 - \xi_1^2}{2} & \dfrac{2(\xi_3^3 - \xi_1^3)}{3} - (\xi_3 - \xi_1) & \xi_3^4 - \xi_1^4 - \dfrac{3(\xi_3^2 - \xi_1^2)}{2} \\ \xi_4 - \xi_1 & \dfrac{\xi_4^2 - \xi_1^2}{2} & \dfrac{2(\xi_4^3 - \xi_1^3)}{3} - (\xi_4 - \xi_1) & \xi_4^4 - \xi_1^4 - \dfrac{3(\xi_4^2 - \xi_1^2)}{2} \end{bmatrix} \times \begin{bmatrix} 1 & \xi_1 & 2\xi_1^2 - 1 & 4\xi_1^3 - 3\xi_1 \\ 1 & \xi_2 & 2\xi_2^2 - 1 & 4\xi_2^3 - 3\xi_2 \\ 1 & \xi_3 & 2\xi_3^2 - 1 & 4\xi_3^3 - 3\xi_3 \\ 1 & \xi_4 & 2\xi_4^2 - 1 & 4\xi_4^3 - 3\xi_4 \end{bmatrix}^{-1} \otimes I \tag{76}$$

$$\tilde{P}_\tau = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \dfrac{\xi_2^2 - \xi_1^2}{2} & \dfrac{\xi_2^3 - \xi_1^3}{3} & \dfrac{\xi_2^4 - \xi_1^4}{2} - \dfrac{\xi_2^2 - \xi_1^2}{2} & \dfrac{4(\xi_2^5 - \xi_1^5)}{5} - (\xi_2^3 - \xi_1^3) \\ \dfrac{\xi_3^2 - \xi_1^2}{2} & \dfrac{\xi_3^3 - \xi_1^3}{3} & \dfrac{\xi_3^4 - \xi_1^4}{2} - \dfrac{\xi_3^2 - \xi_1^2}{2} & \dfrac{4(\xi_3^5 - \xi_1^5)}{5} - (\xi_3^3 - \xi_1^3) \\ \dfrac{\xi_4^2 - \xi_1^2}{2} & \dfrac{\xi_4^3 - \xi_1^3}{3} & \dfrac{\xi_4^4 - \xi_1^4}{2} - \dfrac{\xi_4^2 - \xi_1^2}{2} & \dfrac{4(\xi_4^5 - \xi_1^5)}{5} - (\xi_4^3 - \xi_1^3) \end{bmatrix} \times \begin{bmatrix} 1 & \xi_1 & 2\xi_1^2 - 1 & 4\xi_1^3 - 3\xi_1 \\ 1 & \xi_2 & 2\xi_2^2 - 1 & 4\xi_2^3 - 3\xi_2 \\ 1 & \xi_3 & 2\xi_3^2 - 1 & 4\xi_3^3 - 3\xi_3 \\ 1 & \xi_4 & 2\xi_4^2 - 1 & 4\xi_4^3 - 3\xi_4 \end{bmatrix}^{-1} \otimes I \tag{77}$$

The 4-stage implicit Runge-Kutta method using Picard iteration is expressed as

$$\begin{bmatrix} x_{n+1}(\xi_1) \\ x_{n+1}(\xi_2) \\ x_{n+1}(\xi_3) \\ x_{n+1}(\xi_4) \end{bmatrix} = \begin{bmatrix} x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \end{bmatrix} + \tilde{P} \begin{bmatrix} g(\xi_1) \\ g(\xi_2) \\ g(\xi_3) \\ g(\xi_4) \end{bmatrix} \tag{78}$$

3.4.1 The 1st Version of IRK-FAPI Method

By substituting Eqs. (75)–(77) into the 1st collocated FAPI formula

$$\begin{bmatrix} x_{n+1}(\xi_1) \\ x_{n+1}(\xi_2) \\ x_{n+1}(\xi_3) \\ x_{n+1}(\xi_4) \end{bmatrix} = \begin{bmatrix} x_n(\xi_1) \\ x_n(\xi_2) \\ x_n(\xi_3) \\ x_n(\xi_4) \end{bmatrix} + \tilde{J}_{HP} \left\{ \tilde{Q} \begin{bmatrix} x_n(\xi_1) \\ x_n(\xi_2) \\ x_n(\xi_3) \\ x_n(\xi_4) \end{bmatrix} - \begin{bmatrix} g(\xi_1) \\ g(\xi_2) \\ g(\xi_3) \\ g(\xi_4) \end{bmatrix} \right\} \tag{79}$$

where $\tilde{J}_{HP} = \tilde{J}\tilde{H} - \tilde{P}$ and $\tilde{H} = \tilde{P}_{\tau} - \tilde{T}\tilde{P}$, we can obtain the 1st version of the 4-stage IRK-FAPI formula. Note that it is expressed in a rescaled time domain $\xi \in [-1, 1]$.

3.4.2 The 2nd Version of IRK-FAPI Method

The 2nd version of the 4-stage IRK-FAPI formula can be obtained by substituting Eq. (76) into the 2nd collocated FAPI formula as following:

$$\begin{bmatrix} x_{n+1}(\xi_1) \\ x_{n+1}(\xi_2) \\ x_{n+1}(\xi_3) \\ x_{n+1}(\xi_4) \end{bmatrix} = \begin{bmatrix} x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \end{bmatrix} + \tilde{P} \begin{bmatrix} g(\xi_1) \\ g(\xi_2) \\ g(\xi_3) \\ g(\xi_4) \end{bmatrix} - \tilde{P}\tilde{J} \left\{ \begin{bmatrix} x_{n+1}(\xi_1) \\ x_{n+1}(\xi_2) \\ x_{n+1}(\xi_3) \\ x_{n+1}(\xi_4) \end{bmatrix} - \begin{bmatrix} x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \\ x(\xi_1) \end{bmatrix} - \tilde{P} \begin{bmatrix} g(\xi_1) \\ g(\xi_2) \\ g(\xi_3) \\ g(\xi_4) \end{bmatrix} \right\} \tag{80}$$

4 Numerical Results and Discussion

In this section, conventional predictor-corrector methods, the corresponding FAPI enhanced methods, and MATLAB-built-in ode45/ode113 were used to solve the dynamical responses of some typical nonlinear systems, such as the Mathieu equation, the forced Duffing equation, and the perturbed two-body problem. The numerical simulation was conducted in MATLAB R2017a using an ASUS laptop with an Intel Core i5-7300HQ CPU. GPU acceleration and parallel processing were not used in the following examples.

In this study, to conduct a comprehensive and rational evaluation of each approach, two approaches were employed to compare the accuracy, convergence speed, and computational efficiency of conventional approaches and the corresponding FAPI-enhanced methods. The first approach involves performing the iterative correction only once, which is the general usage of the prediction-correction algorithm. The second approach involves repeating the iterative correction until the corresponding terminal conditions are met.

The maximum computational error in the simulation time interval was selected to represent the computational accuracy when evaluating the computational accuracy of each approach. The solution of ode45 or ode113 was employed as the benchmark for calculating this computational error. The relative and absolute errors of ode45 and ode113 were set as 10^{-15}, while the convergence criteria for the predictor-corrector methods are related to specific problems. In each problem, the computational efficiency of different approaches was compared under the same requirement of computational

accuracy, with computational time used as the metric for measuring the computational efficiency of the various methods.

Mathieu Equation

The Mathieu equation is

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + (\delta - \varepsilon \cos t)x = 0 \tag{81}$$

where the parameters are set as $\delta = 0.5$ and $\varepsilon = 0.1$. The initial conditions are $t = 0$, $x = 1$, and $\mathrm{d}x/\mathrm{d}t = 0$. The Mathieu equation is employed to investigate nonlinear vibration problems with periodic forcing and the periodic motion of nonlinear autonomous systems. The stability of the Mathieu equation's solution depends on the parameters δ and ε. The set of parameters employed in this study yields stable quasi-periodic motion, which has been confirmed through simulation results obtained from ode45 and other methods.

Forced Duffing Equation

The forced Duffing equation is

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + c\frac{\mathrm{d}x}{\mathrm{d}t} + k_1x + k_2x^3 = f\cos(\omega t) \tag{82}$$

where the parameters are set as $c = 0.01$, $k_1 = 1$, $k_2 = 1$, $f = 7.5$, and $\omega = 1$. The initial conditions are $t = 0$, $x = 1.5$, and $\mathrm{d}x/\mathrm{d}t = 0$. The system could exhibit chaotic motion using ode45 and other reliable approaches. Any small state perturbation may result in substantial deviations in the simulation results because of the sensitivity of chaotic systems to the current state. Therefore, when solving chaotic problems, computational errors of general methods with low accuracy accumulate rapidly.

Low-Earth-Orbit Tracking Problem

The orbital problem is considered in the following form:

$$m\frac{\mathrm{d}^2r}{\mathrm{d}t^2} = -\frac{\partial U}{\partial r} \tag{83}$$

where $r = [x, y, z]^{\mathrm{T}}$ represents the position vector and U represents the potential function of the Earth gravity field. The gravity potential is modeled using the 10 deg Earth Gravity Model 2008. The initial condition is set as

$$r_0 = [-0.3889, 7.7388, 0.6736] \times 10^6\,\mathrm{m}$$

$$v_0 = [-3.5794, 0, 6.1997] \times 10^3\mathrm{m/s}$$

The computational accuracy of orbital motion plays a crucial role in determining the success of space missions. In the case of LEO, the long-term trajectory of a spacecraft can be significantly affected by a very small perturbation term of gravity force. A high-order gravitational model should be used to achieve relatively high accuracy. In this case, most computational time is dedicated to assessing the gravitational force terms. The relative error of position is defined as $\varepsilon = \|\Delta r\|_2/\|r\|_2$, where r represents the reference result obtained using ode113 and Δr represents the position error at the sampling point. The relative error of the velocity can be modeled similarly. The simulation time for the LEO tracking problem is set to 10 times the orbital period to reliably assess each method.

4.1 Modified Euler Method Using FAPI

Using ode45/ode113 as the benchmark, Figs. 2 and 3 show a straightforward comparison of the phase plane portraits, time responses, and computational errors of MEM, 1st ME-FAPI, and 2nd ME-FAPI when solving the Matthieu and forced Duffing equations. Fig. 4 shows a comparison of the trajectory of LEO and the computational errors. Table 3 presents the time step size of each method.

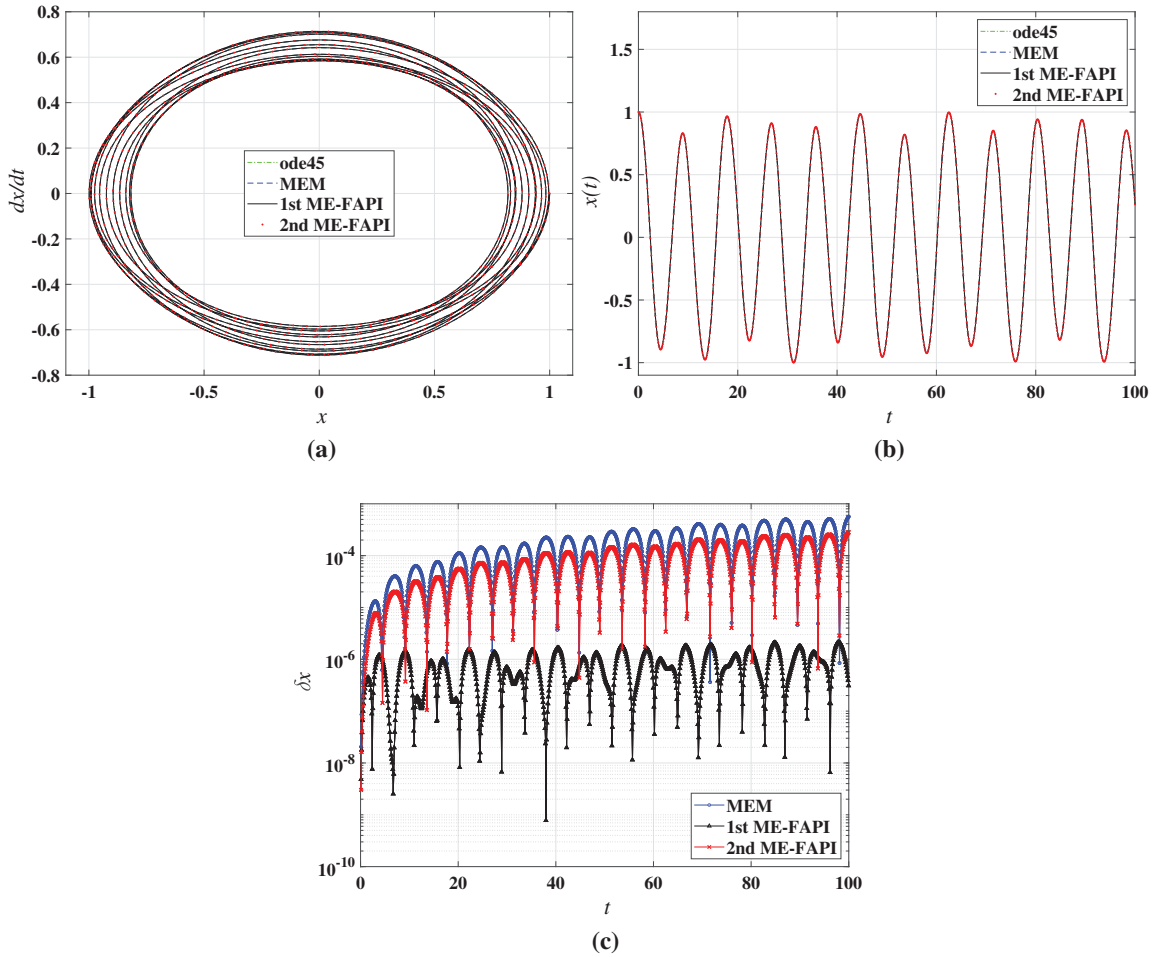


Figure 2: Results of Mathieu equation using MEM, ME-FAPI, and ode45: (a) phase plane portrait; (b) time responses; (c) computational error

The MEM and ME-FAPI methods exhibited a good match with ode45/ode113. In Fig. 2c, the error curve of the 1st ME-FAPI method is relatively steady, whereas the error curves of the 2nd ME-FAPI method and MEM gradually increase with time. This indicates that the 1st ME-FAPI method has a significantly lower error accumulation rate than the 2nd ME-FAPI method and MEM when solving the Mathieu equation. In the numerical results of the Duffing equation, an improvement in computational accuracy with the 1st ME-FAPI method is also observed. The error curve of the LEO tracking problem in Fig. 4b shows a significant correlation with the current orbital position (or altitude) of the satellite, which is due to the strong association between the macroscopic gravity field and the orbital position (altitude). This observation reveals that the 2nd ME-FAPI method is more accurate in solving the orbit tracking problem.

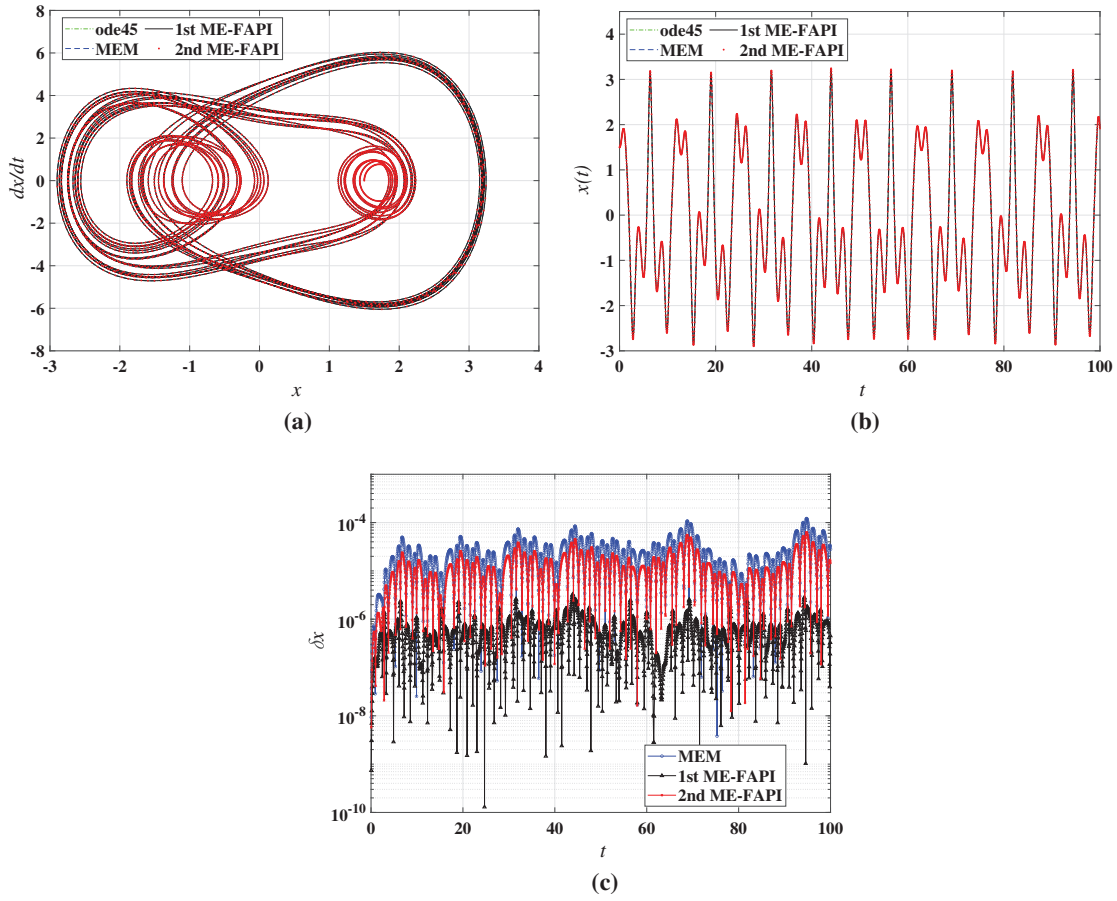


Figure 3: Results of the forced duffing equation using MEM, ME-FAPI, and ode45: (a) phase plane portrait; (b) time responses; (c) computational error

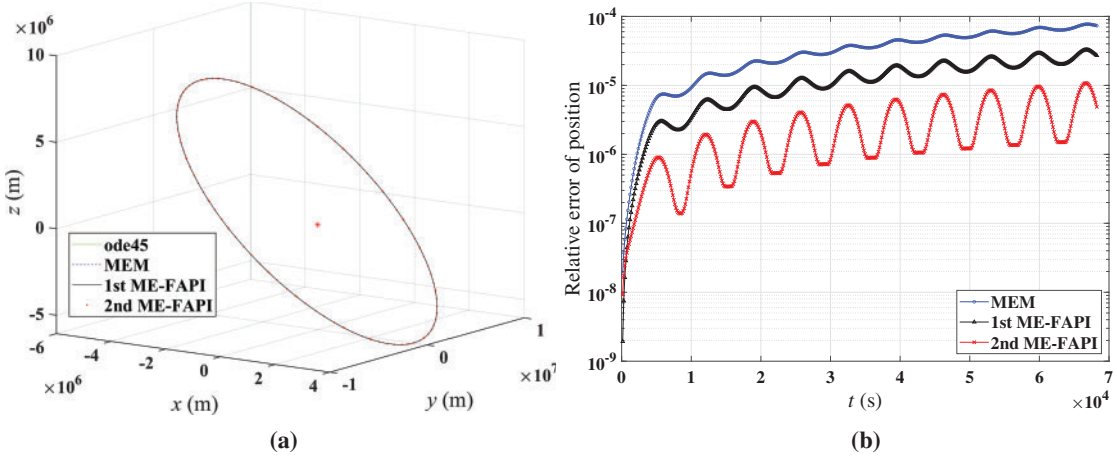


Figure 4: Results of MEM, ME-FAPI, and ode113 for LEO: (a) trajectory of LEO; (b) computational error

Table 3: The configurations of MEM and ME-FAPI method

Problems	Corrected once	Corrected until converged
	MEM/ME-FAPI	MEM/ME-FAPI
Mathieu equation	$\Delta t = 0.01$, $t = [0, 100]$	$tol = 10^{-12}$, $t = [0, 100]$
The forced duffing equation	$\Delta t = 0.001$, $t = [0, 100]$	$tol = 10^{-12}$, $t = [0, 100]$
Low-earth-orbit tracking problem	$\Delta t = 1s$	$tol = 10^{-8}$

A more comprehensive presentation of the computational accuracy and efficiency of each method is shown in Figs. 5–7 when the proposed ME-FAPI method and MEM are only corrected once. The maximum computational error of each method was recorded by sweeping the time step size h. This enables us to depict the curve of the computational error to the time step size. In addition, the error tolerance for each method is specified, and the corresponding h values are selected to enable a comparison of the computational time for each method under the same computational accuracy. Table 3 shows the duration of the simulated dynamical response.

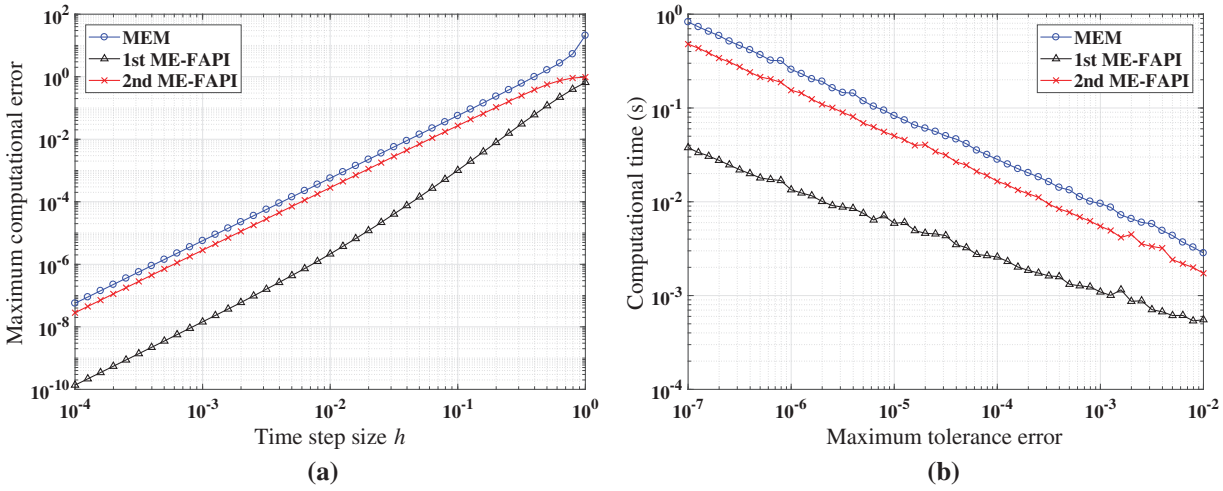


Figure 5: Performance of MEM and ME-FAPI in solving Mathieu equation when corrected only once: (a) Maximal error; (b) computational time

Figs. 5–7 demonstrate that the ME-FAPI method proposed in this study outperforms MEM when corrected once in terms of accuracy and efficiency. The two versions of the ME-FAPI method have varying applicability for different nonlinear systems. For instance, the 1st ME-FAPI method is more suitable for solving the Mathieu equation, where its computational error is 1–2 magnitudes lower than that of the MEM, and its computational time is only 10% of the latter. However, the 2nd ME-FAPI method is more appropriate for the LEO tracking problem, offering 1 order of magnitude higher computational accuracy and a computational speed 1 time faster than the MEM.

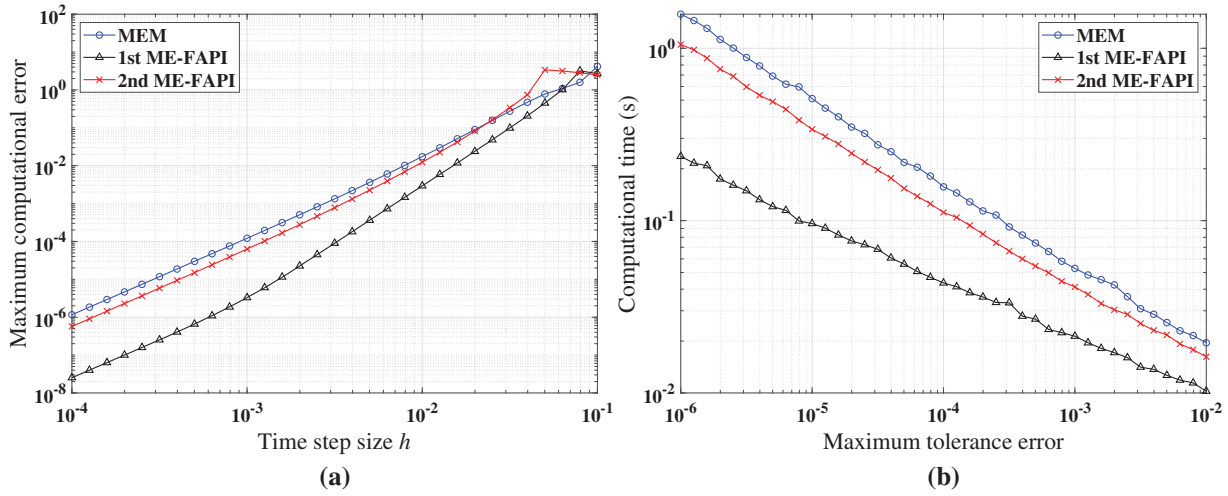


Figure 6: Performance of MEM, and ME-FAPI in solving the forced Duffing equation when corrected only once: (a) Maximal error; (b) computational time

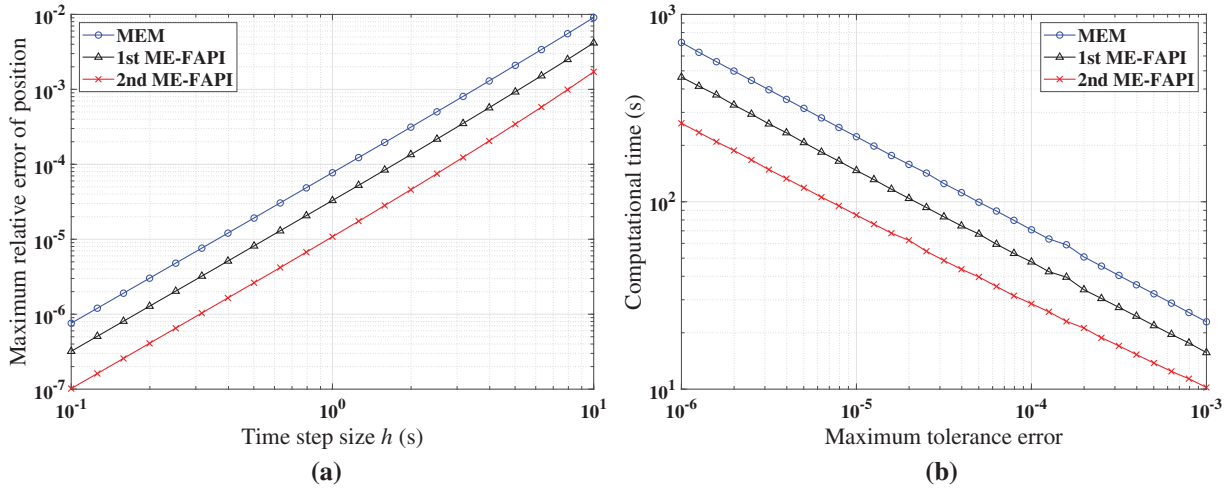


Figure 7: Performance of MEM, and ME-FAPI for LEO when corrected only once: (a) Maximal error; (b) computational time

The following section examines the performance indices of each approach after the convergence of the iterative process, including the maximum computational error, average iteration steps, and computational efficiency. Table 3 presents the configurations of these methods, and Figs. 8–10 illustrate the simulation results. Since the convergence speed of each method shows similar rules in different cases, only the corresponding simulation results of the Mathieu equation are retained.

Figs. 8–10 demonstrate that at least one of the two versions of the ME-FAPI method proposed in this study will have higher computational efficiency than the classical MEM. Only the 1st ME-FAPI method is more efficient than MEM when solving the Mathieu equation, as illustrated in Fig. 8.

However, the 1st and 2nd ME-FAPI methods are more efficient than MEM when solving the Duffing equation. Thus, in practical applications, it is crucial to select the appropriate approach based on the specific problem to be solved.

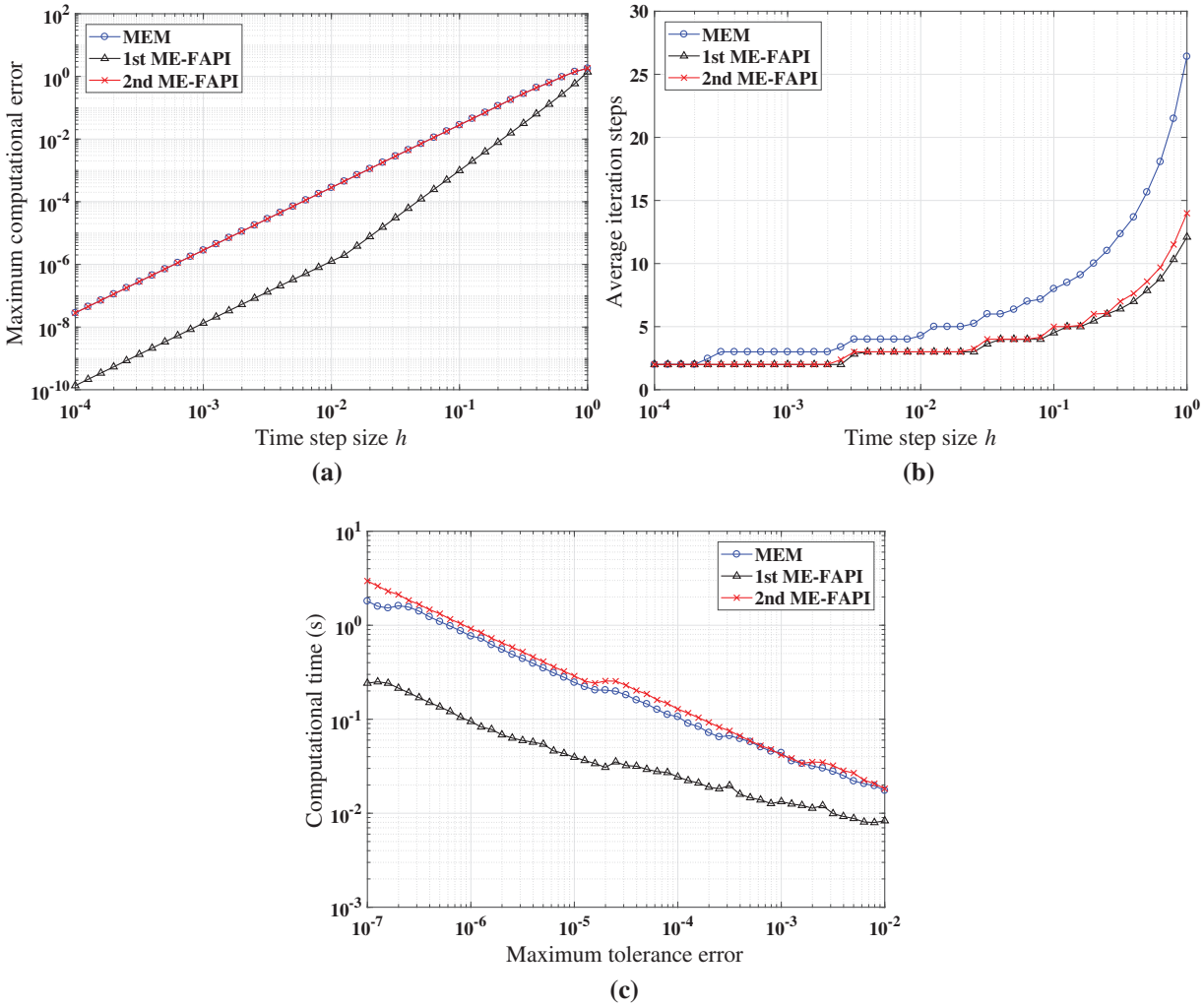


Figure 8: Performance of MEM, and ME-FAPI in solving Mathieu equation when the algorithms converge: (a) Maximal error; (b) average iteration steps; (c) computational time

Furthermore, Fig. 8b illustrates that the 1st ME-FAPI and 2nd ME-FAPI methods converge almost as fast, while the MEM method exhibits the slowest convergence. The advantage of convergence speed substantially enhances the efficiency of the method. For instance, the evaluations of the force model consume most of the computational time when solving the perturbed two-body problem. Therefore, a reduction in the number of iterations plays a significant role in reducing computational cost.

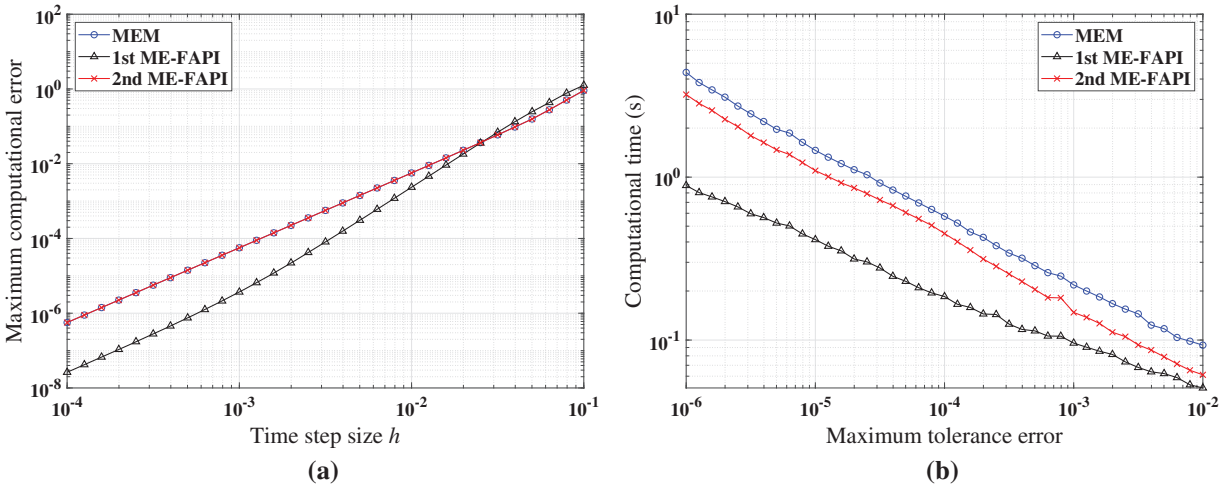


Figure 9: Performance of MEM, and ME-FAPI in solving the forced duffing equation when the algorithms converge: (a) Maximal error; (b) computational time

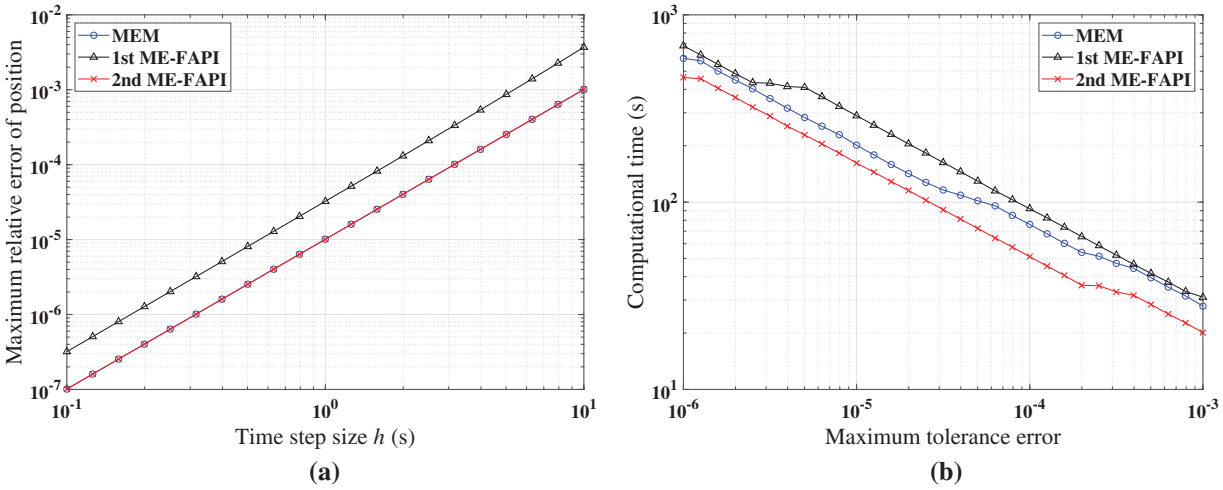


Figure 10: Performance of MEM, and ME-FAPI for LEO when the algorithms converge: (a) Maximal error; (b) computational time

4.2 Adams-Bashforth-Moulton Method Using FAPI

When corrected only once, the simulation results in Figs. 11–13 provide an overview of the computational accuracy and efficiency of each method. Table 4 shows the duration of the simulated dynamical response.

Figs. 11–13 demonstrate that when corrected only once, the 1st ABM-FAPI method is more accurate and efficient than the ABM method in all cases. For instance, the computational error of the 1st ABM-FAPI method is 1–2 orders of magnitude lower than that of the ABM method at the same step size when solving the Mathieu equation (Fig. 11). Furthermore, at the same computational accuracy, the former can save approximately 10% – 60% of the computational time of the latter.

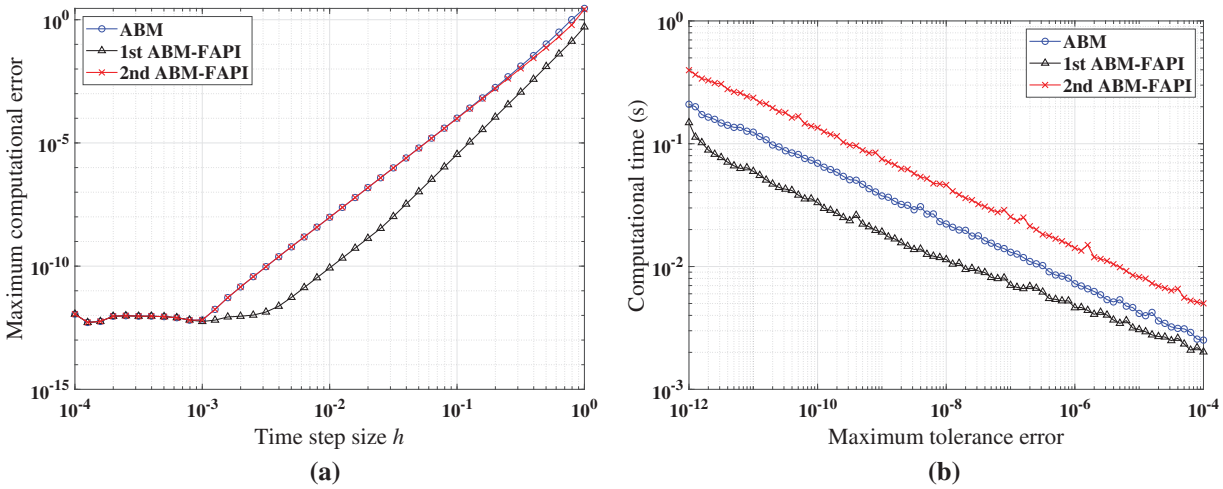


Figure 11: Performance of ABM, and ABM-FAPI in solving Mathieu equation when corrected only once: (a) Maximal error; (b) computational time

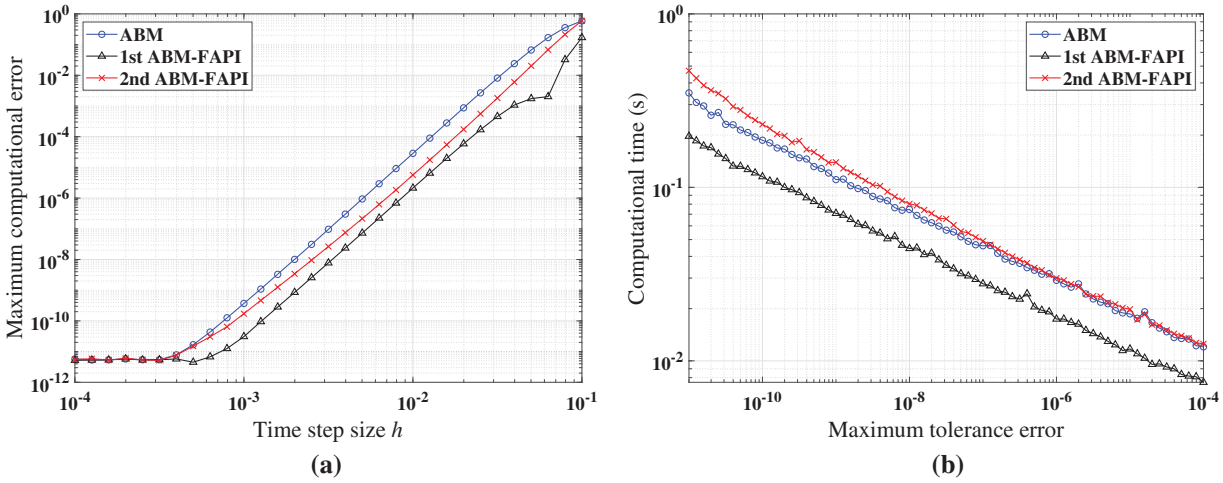
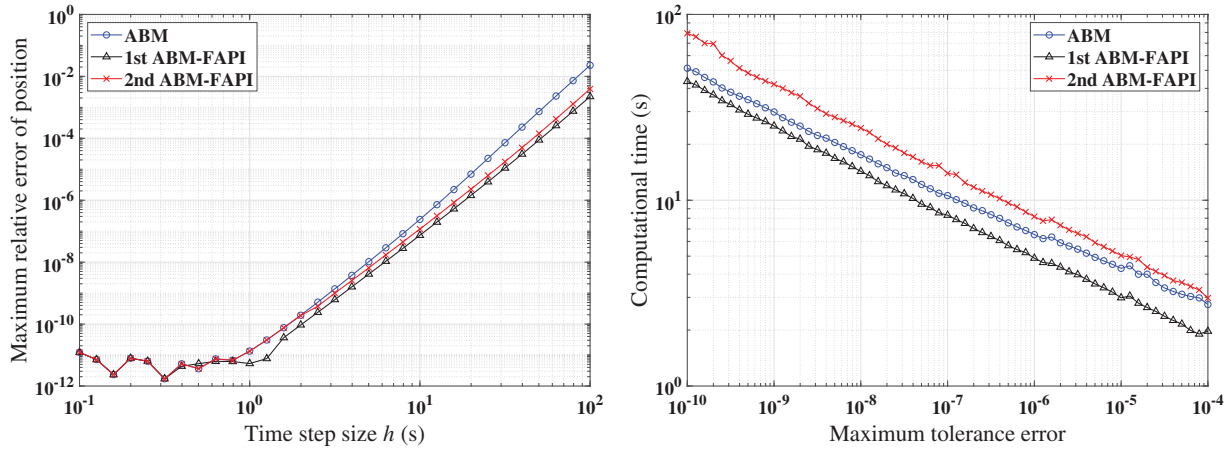


Figure 12: Performance of ABM, and ABM-FAPI in solving the forced Duffing equation when corrected only once: (a) Maximal error; (b) computational time

New results can be obtained when the algorithms converge, as shown in Figs. 14–16. Table 4 shows the configurations of these methods. For the same reason as in the previous section, only the simulation results of the Mathieu equation concerning the convergence speed are retained.

The simulation findings indicate that, in all cases, the 1st ABM-FAPI method significantly reduces computational time compared with the classical ABM method. For instance, as shown in Fig. 14c, the 1st ABM-FAPI method can save approximately 28% – 65% of the computational time of the ABM method.

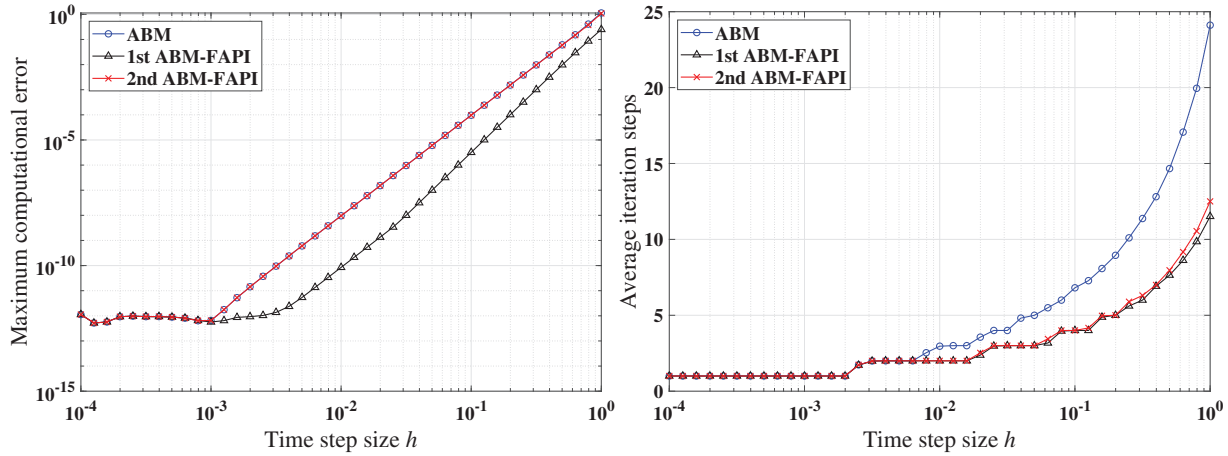


(a) (b)

Figure 13: Performance of ABM, and ABM-FAPI for LEO when corrected only once: (a) Maximal error; (b) computational time

Table 4: The configurations of ABM, and ABM-FAPI

Problems	Corrected once	Corrected until converged
	ABM/ABM-FAPI	ABM/ABM-FAPI
Mathieu equation	$t = [0, 200]$	$tol = 10^{-14}$, $t = [0, 200]$
The forced duffing equation	$t = [0, 100]$	$tol = 10^{-14}$, $t = [0, 100]$
Low-earth-orbit tracking problem	/	$tol = 10^{-9}$



(a) (b)

Figure 14: (Continued)

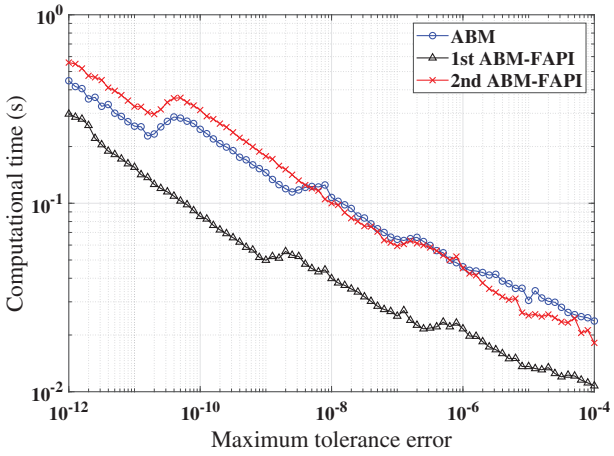


Figure 14: Performance of ABM, and ABM-FAPI in solving Mathieu equation when the algorithms converge: (a) Maximal error; (b) average iteration steps; (c) computational time

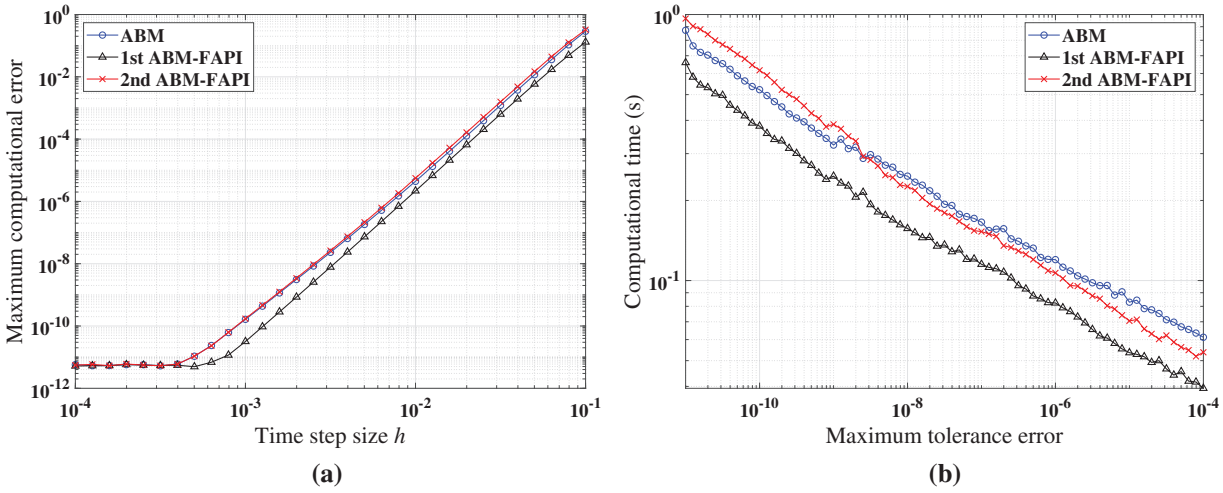


Figure 15: Performance of ABM, and ABM-FAPI in solving the forced Duffing equation when the algorithms converge: (a) Maximal error; (b) computational time

Fig. 14b demonstrates that the convergence speeds of the two versions of the ABM-FAPI method are almost the same. The ABM and ABM-FAPI methods converge at the same speed when the step size is small. However, the convergence speed of the ABM method lags behind that of the ABM-FAPI method as the step size increases.

Furthermore, the algorithm complexity of the 2nd ABM-FAPI method, which incorporates more information about the Jacobi matrix at time nodes, is significantly higher than that of the 1st ABM-FAPI method. However, as shown in Figs. 11–16, the performance of the 2nd ABM-FAPI method does not significantly outperform that of other methods, whereas the relatively "simpler" 1st ABM-FAPI method exhibits a significant advantage in computational accuracy and efficiency.

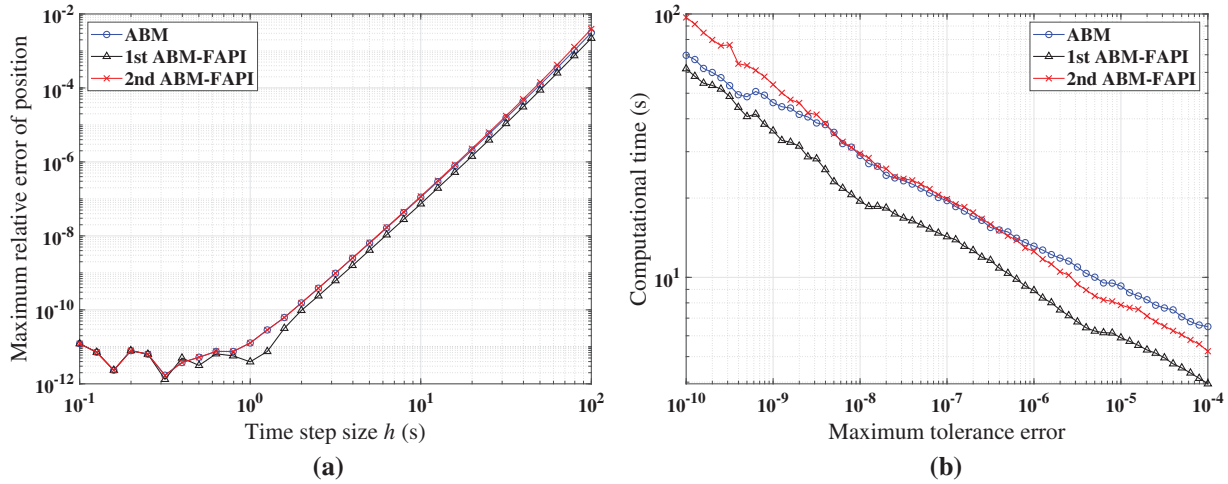


Figure 16: Performance of ABM, and ABM-FAPI for LEO when the algorithms converge: (a) Maximal error; (b) computational time

4.3 Implicit Runge-Kutta Method Using FAPI

The initial approximation in each step is selected as a straight line without additional computation of the derivatives when using the IRK method and the corresponding IRK-FAPI method to solve the nonlinear system. This initial approximation serves as a "cold start" for the methods. Generally, the "cold start" is a very rough prediction method.

Figs. 17–19 show the simulation results when correcting once. Table 5 presents the duration of the simulated dynamical response. Due to the notably poor accuracy of the IRK method, to the extent that it can barely satisfy the corresponding accuracy requirements in some cases, its computational time is not recorded.

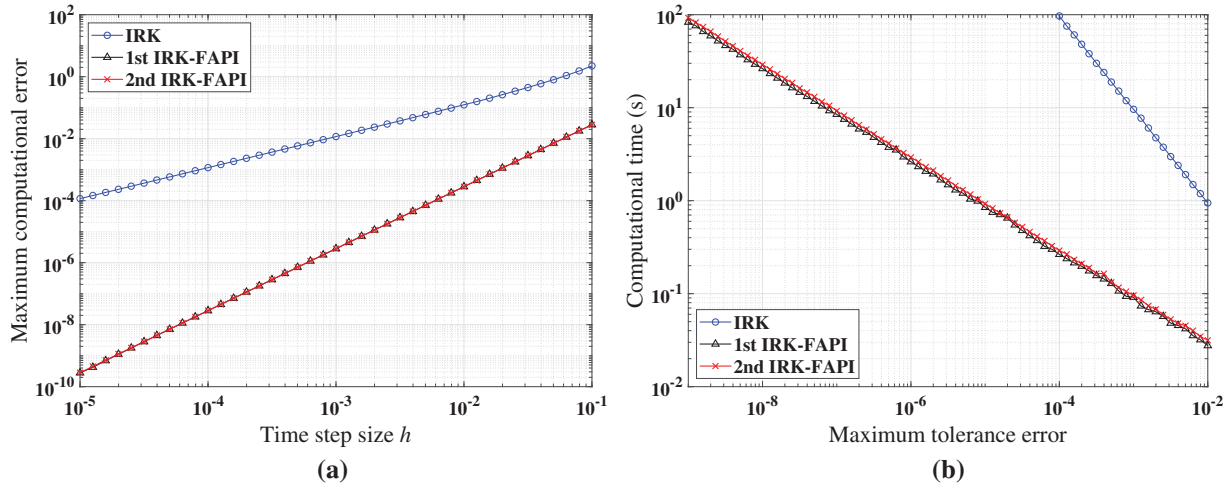


Figure 17: Performance of IRK, and IRK-FAPI in solving Mathieu equation when corrected only once: (a) Maximal error; (b) computational time

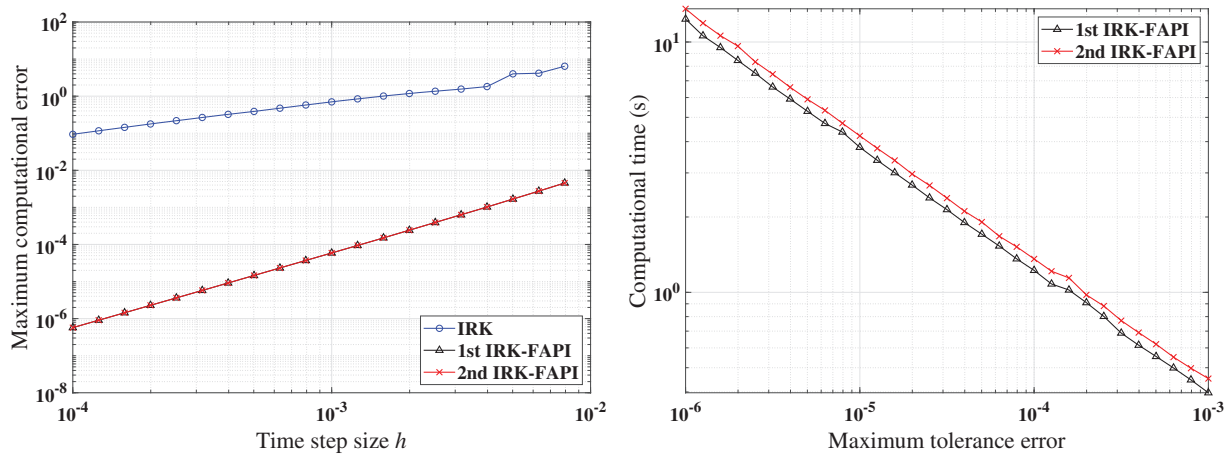


Figure 18: Performance of IRK, and IRK-FAPI in solving the forced Duffing equation when corrected only once: (a) Maximal error; (b) computational time

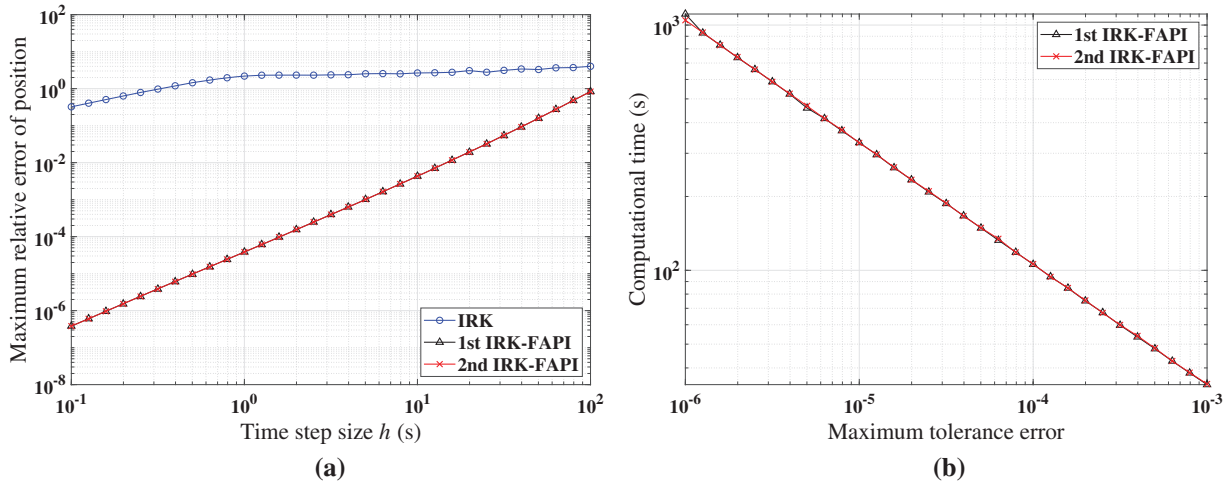


Figure 19: Performance of IRK, and IRK-FAPI for LEO when corrected only once: (a) Maximal error; (b) computational time

Table 5: The configurations of IRK, and IRK-FAPI

Problems	Corrected once	Corrected until converged
	IRK/IRK-FAPI	IRK/IRK-FAPI
Mathieu equation	$t = [0, 100]$	$tol = 10^{-14}$, $t = [0, 100]$
The forced duffing equation	$t = [0, 100]$	$tol = 10^{-14}$, $t = [0, 1000]$
Low-earth-orbit tracking problem	/	$tol = 10^{-10}/tol = 10^{-8}$

Figs. 17–19 demonstrate that the IRK-FAPI method outperforms the IRK method with extremely rough initial guesses in terms of accuracy and efficiency. As shown in Fig. 18, even with a very small step size, the IRK method cannot predict the long-term dynamical response of the chaotic system, whereas the IRK-FAPI method is very accurate by correcting once.

Similarly, Figs. 20–22 show the simulation results when the algorithm converges. Table 5 presents the configurations of these methods.

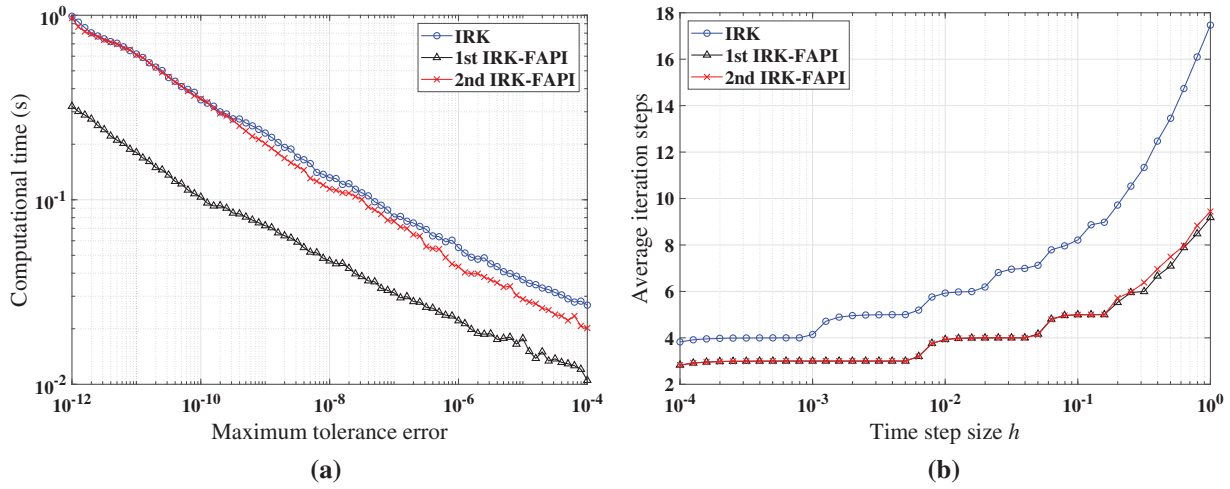


Figure 20: Performance of IRK, and IRK-FAPI in solving Mathieu equation when the algorithms converge: (a) computational time; (b) average iteration steps

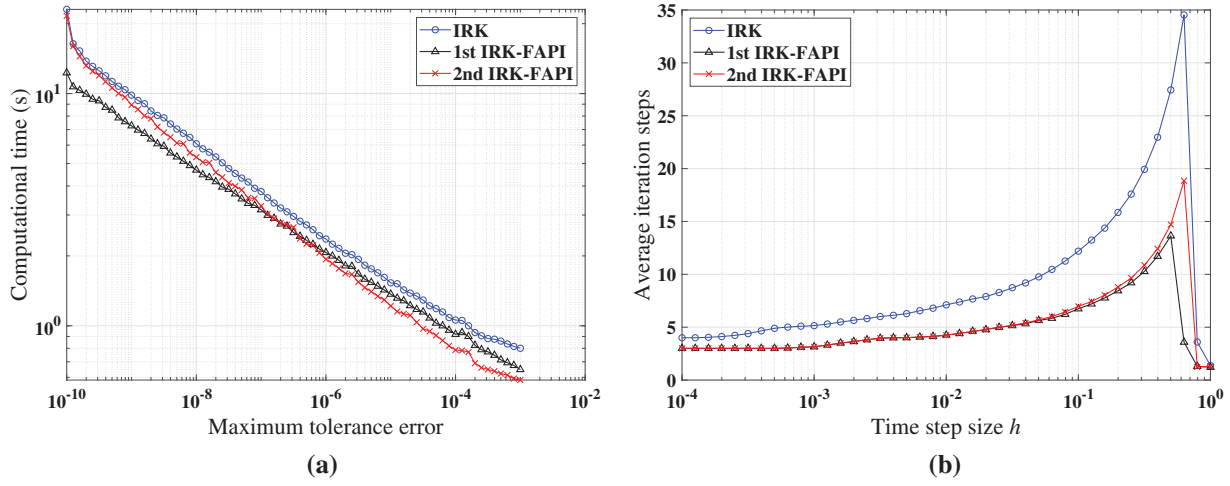


Figure 21: Performance of IRK, and IRK-FAPI in solving the forced Duffing equation when the algorithms converge: (a) computational time; (b) average iteration steps

Figs. 20–22 demonstrate that at least one of the two versions of the IRK-FAPI method will have higher computational efficiency than the IRK method, although the difference is not obvious in the case of the Duffing-Holmes's oscillator. For instance, when solving the Mathieu equation, the 1st ABM-FAPI method can save more than half the computational time of the ABM method.

Furthermore, the two versions of the IRK-FAPI method converge at almost the same speed, and both converge faster than the IRK method.

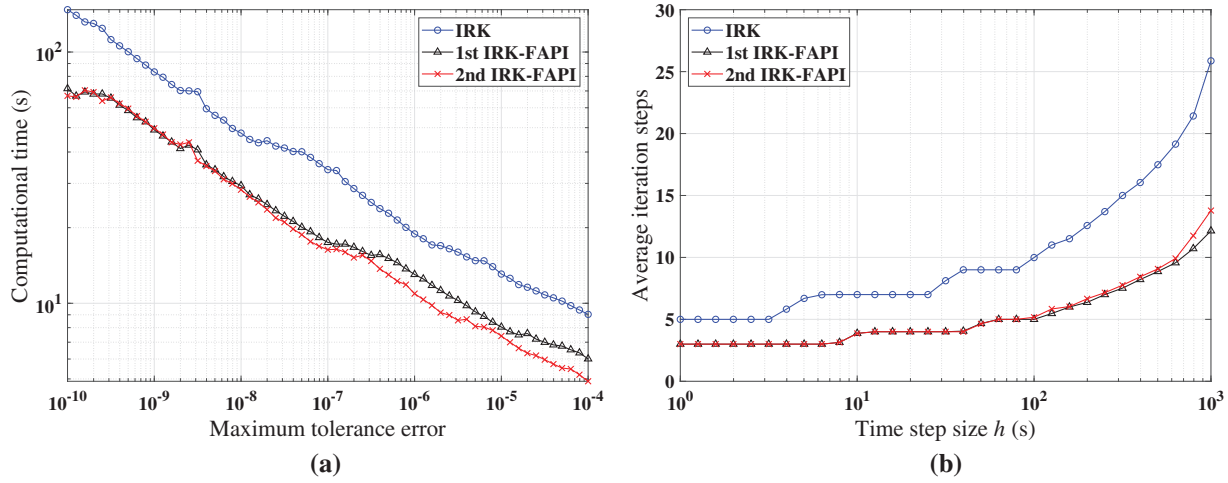


Figure 22: Performance of IRK, and IRK-FAPI for LEO when the algorithms converge: (a) computational time; (b) average iteration steps

5 Conclusion

This study proposes a novel approach for deriving efficient predictor-corrector approaches based on FAPI. Three classical approaches, including the modified Euler approach, the Adams-Bashforth-Moulton approach, and the implicit Runge-Kutta approach, are used to demonstrate how FAPI can be employed to modify and improve commonly used predictor-corrector approaches. For each classical method, two versions of the FAPI correctors are derived. The resulting six FAPI-featured approaches are further used to numerically solve three different types of nonlinear dynamical systems originating from various research fields, ranging from mechanical vibration to astrodynamics. The numerical findings indicate that these enhanced correctors can effectively solve these problems with high accuracy and efficiency, with the 1st version of FAPI featured methods displaying superior performance in most cases. For instance, the 1st FAPI correctors achieve 1–2 orders of magnitude higher accuracy and efficiency than conventional correctors in solving the Mathieu equation. Furthermore, at least one version of the FAPI correctors shows superiority over the original correctors in all these problems, particularly when the correctors are used only once per step. In future studies, the application of the proposed approach to bifurcation problems and high-dimensional nonlinear structural problems will be presented.

The proposed approach offers a general framework for enhancing existing predictor-corrector methods by deriving new correctors using FAPI. Although three existing methods were retrofitted and their improvements demonstrated with three problems, it is anticipated that similar enhancements can be achieved by applying the proposed approach to various other predictor-corrector methods and nonlinear problems. For clarity, methods with low approximation orders (<5) were selected to illustrate our approach. However, methods of higher order can also be derived using the same procedure. In addition, our method can be directly combined with existing adaptive computational approaches for predictor-corrector methods to adaptively solve real-world problems.

Acknowledgement: The authors are grateful for the support by Northwestern Polytechnical University and Texas Tech University. Additionally, we would like to express our appreciation to anonymous reviewers and journal editors for assistance.

Funding Statement: This work is supported by the Fundamental Research Funds for the Central Universities (No. 3102019HTQD014) of Northwestern Polytechnical University, Funding of National Key Laboratory of Astronautical Flight Dynamics, and Young Talent Support Project of Shaanxi State.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Xuechuan Wang, Wei He; data collection: Wei He; analysis and interpretation of results: Xuechuan Wang, Wei He, Haoyang Feng; draft manuscript preparation: Xuechuan Wang, Wei He, Haoyang Feng, Satya N. Atluri. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data supporting the conclusions of this article are included within the article. Any queries regarding these data may be directed to the corresponding author.

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

References

1. Butcher, J. C. (2016). *Numerical methods for ordinary differential equations*. USA: John Wiley & Sons.
2. Smith, G. D., Smith, G. D., Smith, G. D. S. (1985). *Numerical solution of partial differential equations: Finite difference methods*. UK: Oxford university press.
3. Hilber, H. M., Hughes, T. J., Taylor, R. L. (1977). Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics, 5(3),* 283–292.
4. Fox, L. (1962). *Numerical solution of ordinary and partial differential equations*. UK: Pergamon Press.
5. Süli, E. (2010). *Numerical solution of ordinary differential equations*. UK: University of Oxford.
6. Meyer, K., Hall, G., Offin, D. (2008). *Introduction to hamiltonian dynamical systems and the N-body problem*. German: Springer Science and Business Media.
7. Turan, E., Speretta, S., Gill, E. (2022). Autonomous navigation for deep space small satellites: Scientific and technological advances. *Acta Astronautica, 83,* 56–74.
8. Xu, S., Peng, H. (2019). Design, analysis, and experiments of preview path tracking control for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems, 21(1),* 48–58.
9. 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.
10. Inokuti, M., Sekine, H., Mura, T. (1978). General use of the lagrange multiplier in nonlinear mathematical physics. *Variational Method in the Mechanics of Solids, 33(5),* 156–162.
11. Wang, X., Atluri, S. N. (2017). A novel class of highly efficient and accurate time-integrators in nonlinear computational mechanics. *Computational Mechanics, 59,* 861–876.
12. 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.
13. Atluri, S. N. (2005). *Methods of computer modeling in engineering & the sciences*, vol. 1. USA: Tech Science Press.

14. Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics, 117(1),* 1–19.
15. Noël, J. P., Kerschen, G. (2017). Nonlinear system identification in structural dynamics: 10 more years of progress. *Mechanical Systems and Signal Processing, 83,* 2–35.
16. Nazir, G., Zeb, A., Shah, K., Saeed, T., Khan, R. A. et al. (2021). Study of COVID-19 mathematical model of fractional order via modified euler method. *Alexandria Engineering Journal, 60(6),* 5287–5296.
17. Kumar, S., Kumar, R., Agarwal, R. P., Samet, B. (2020). A study of fractional lotka-volterra population model using haar wavelet and adams-bashforth-moulton methods. *Mathematical Methods in the Applied Sciences, 43(8),* 5564–5578.
18. Kennedy, C. A., Carpenter, M. H. (2019). Diagonally implicit runge–kutta methods for stiff odes. *Applied Numerical Mathematics, 146,* 221–244.
19. Kovacic, I., Rand, R., Mohamed Sah, S. (2018). Mathieu's equation and its generalizations: Overview of stability charts and their features. *Applied Mechanics Reviews, 70(2),* 020802.
20. Singh, H., Srivastava, H. (2020). Numerical investigation of the fractional-order liénard and duffing equations arising in oscillating circuit theory. *Frontiers in Physics, 8,* 120.
21. Amato, D., Bombardelli, C., Baù, G., Morand, V., Rosengren, A. J. (2019). Non-averaged regularized formulations as an alternative to semi-analytical orbit propagation methods. *Celestial Mechanics and Dynamical Astronomy, 131(5),* 1–38.
22. Zennaro, M. (1986). Natural continuous extensions of runge-kutta methods. *Mathematics of Computation, 46(173),* 119–133.