Journal of Scientific Computing (2023) 95:27
https://doi.org/10.1007/s10915-023-02156-y

Parallel Numerical Picard Iteration Methods

Yinkun Wang[1]

Received: 14 December 2021 / Revised: 7 June 2022 / Accepted: 12 February 2023 /
Published online: 1 March 2023

Abstract
In this work, we propose a new class of parallel time integrators for initial-value problems based on the well-known Picard iteration. To this end, we first investigate a class of sequential integrators, known as numerical Picard iteration methods, which falls into the general framework of deferred correction methods. We show that the numerical Picard iteration methods admit a $\min(J, M+1)$-order rate of convergence, where J denotes the number of Picard iterations and $M+1$ is the number of collocation points. We then propose a class of parallel solvers so that J Picard iterations can be proceeded simultaneously and nearly constantly. We show that the parallel solvers yield the same convergence rate as that of the numerical Picard iteration methods. The main features of the proposed parallelized approach are as follows. (1) Instead of computing the solution point by point [as in revisionist integral deferred correction (RIDC) methods], the proposed methods proceed segment by segment. (2) The proposed approach leads to a higher speedup; the speedup is shown to be $J(M+1)$ (while the speedup of the Jth order RIDC is, at most, J). (3) The approach is applicable for non-uniform points, such as Chebyshev points. The stability region of the proposed methods is analyzed in detail, and we present numerical examples to verify the theoretical findings.

Keywords Parallel computation · Picard iteration · Initial-value problems · Stability region

1 Introduction

Numerical methods are of significant importance in science and engineering for initial-value problems (IVPs). In the past several decades, great efforts have been focused on the construction of efficient, stable, high-order, and easily parallelized time integrators for solving IVPs [3–5, 15–17]. While Runge–Kutta and linear multi-step methods are popular approaches used to obtain a high-order rate of convergence, alternative approaches, such as enhancing the convergence of low-order schemes through deferred correction (DC) methods [12, 21], have also been proposed. Two important variants of DC methods, the spectral deferred correction (SDC) method [12] and the integral deferred correction (IDC) method [6], have also been

✉ Yinkun Wang
yinkun5522@163.com

[1] Department of Mathematics, National University of Defense Technology, Changsha, People's Republic of China

proposed in recent years. To further enhance efficiency, parallel-in-time computations have begun to receive great attention. In fact, early studies along this line date back to more than 50 years ago [20]. Since then, there have been several interesting approaches for parallel-in-time computations; the reader is referred to [13, 14, 19, 22] and references therein for more detailed information.

In this paper, we propose a novel parallel-in-time integrator based on Picard iterations: parallel numerical Picard iteration. The main reason for choosing the Picard iteration as our starting point is that it is one of the simplest approaches and has been successfully applied to the propagations of perturbed orbits in astrodynamics [26, 27]. The original efforts were made by Clenshaw and collaborators [9–11], who developed the Picard iteration numerically with Chebyshev polynomials. The Picard iteration has also been adopted in [24, 25] to improve the convergence of SDCs and low-order methods. Nevertheless, the convergence of the numerical Picard method has not been well studied. To this end, our first contribution in this work is to present detailed convergence analysis of a class of numerical Picard iteration methods. In particular, we show a super-convergence property of the numerical Picard iteration methods by using the Legendre–Gauss points. More precisely, we demonstrate that the numerical Picard iteration methods admit a $\min(J, M+1)$-order rate of convergence, where J denotes the number of Picard iterations and $M+1$ is the number of collocation points.

In addition, we propose a class of parallel solvers by rearranging the order of numerical Picard iterations in different sub-intervals so that J Picard iterations can be proceeded simultaneously and nearly constantly. Furthermore, we demonstrate that the parallel solvers yield the same convergence rate as that of the numerical Picard iteration methods. We remark that our approach is analogous to that of the revisionist integral deferred correction (RIDC) methods in [7, 8]. However, compared to RIDC, the approaches proposed herein yield the following main features.

- Instead of computing the solution point by point (as in RIDC methods), the proposed methods proceed segment by segment.
- The proposed approach leads to a higher speedup: the speedup is shown to be $J(M+1)$ (while the speedup of the Jth order RIDC is, at most, J).
- The approach is applicable for non-uniform points, such as Chebyshev points.

The stability region of the proposed methods is analyzed in detail; we also present numerical examples to verify the theoretical findings.

The rest of this paper is organized as follows. In Sect. 2, we review the numerical Picard iteration methods and present our convergence analysis results. The parallel numerical Picard iteration methods are presented in Sect. 3, followed by stability analysis in Sect. 4. Numerical examples are presented in Sect. 5 to verify the theoretical results. We finally give concluding remarks in Sect. 6.

2 Numerical Picard Iteration Methods

In this section, we present the basic ideas of the numerical Picard iteration methods. First, we consider the following problem:

$$\begin{cases} y'(t) = f(t, y(t)), t \in [a, b], \\ y(0) = y_a, \end{cases} \tag{1}$$

where $y_a, y(t) \in \mathbb{C}^m$ and $f : \mathbb{R} \times \mathbb{C}^m \to \mathbb{C}^m$. We assume that the function f satisfies the Lipschitz continuous condition

$$\|f(\cdot, y_1) - f(\cdot, y_2)\| \le L\|y_1 - y_2\|, \tag{2}$$

where L is the Lipschitz constant and $\|\cdot\|$ denotes the Euclidean norm. Integrating Eq. (1) with respect to t, we obtain

$$y(t) = y(a) + \int_a^t f(\tau, y(\tau))d\tau. \tag{3}$$

The well-known form of the Picard iteration is given by [16]

$$y_{i+1}(t) = y(a) + \int_a^t f(\tau, y_i(\tau))d\tau, \quad i = 0, 1, \ldots, \tag{4}$$

where y_0 is an initial guess. It has been shown that the above Picard iteration yields a super-linear convergence rate when t is close enough to a [18]. Consequently, for a large time domain, one can split the interval $[a, b]$ into small sub-intervals and conduct the Picard iteration on each sub-interval.

Note that, in the above iterations, repeated evaluations of integrals are needed. To alleviate this computational issue, a numerical version of the Picard iterations is proposed. To illustrate its main idea, we consider a standard interval $I = [-1, 1]$ and let $\{c_i : i = 0, 1, \ldots, M\}$ be a set of collocation points on I such that $-1 \le c_0 < c_1 < \cdots < c_M \le 1$. For a given vector $\boldsymbol{\varphi} = [\varphi_0, \varphi_1, \ldots, \varphi_M]^\top$, we introduce the standard Lagrange interpolation operator $\mathcal{I}_M : \mathbb{C}^{M+1} \times I \to \mathbb{C}$:

$$\mathcal{I}_M(\boldsymbol{\varphi}, t) = \sum_{j=0}^{M} \varphi_j \ell_j(t), t \in I,$$

where the functions $\ell_j(t)$ are given by

$$\ell_j(t) := \prod_{\substack{i \ne j \\ i=0,1,\ldots,M}} \frac{t - c_i}{c_j - c_i}, \quad j = 0, \ldots, M.$$

We set $Y^{[0]}(t) \equiv y(-1), t \in I$, and once the ith approximation $Y^{[i]}$ is obtained, the numerical Picard iteration solves the $(i+1)$th approximation $Y^{[i+1]}$ by

$$Y^{[i+1]}(t) = y(-1) + \int_{-1}^t \mathcal{I}_M(f(\mathbf{Y}^{[i]}), s)ds, \quad i = 0, 1, \ldots, \tag{5}$$

where

$$\mathbf{Y}^{[i]} := [Y^{[i]}(c_0), Y^{[i]}(c_1), \ldots, Y^{[i]}(c_M)]^\top,$$
$$f(\mathbf{Y}^{[i]}) := [f(c_0, Y^{[i]}(c_0)), f(c_1, Y^{[i]}(c_1)), \ldots, f(c_M, Y^{[i]}(c_M))]^\top.$$

For easily presenting our analysis results, we next define a numerical-Picard-iteration-based integrator for Eq. (1). To this end, we first discretize $[a, b]$ into sub-intervals

$$\mathcal{M}_h := \{t_n : a = t_0 < t_1 < \cdots < t_N = b\}.$$

Then, we set

$$I_n := (t_n, t_{n+1}], \ \bar{I}_n := \{t_n\} \cup I_n, \ h_n := t_{n+1} - t_n, \ n = 0, 1, \ldots, N-1.$$

We also define the size of the mesh $\mathcal{M}_h$ as

$$h := \max\{h_n : 0 \le n \le N-1\}.$$

We now denote $S_{M+1}(\mathcal{M}_h)$ as the space of the piecewise polynomial space

$$S_{M+1}(\mathcal{M}_h) := \left\{v \in C([a,b]) : v|_{I_n} \in \pi_{M+1}, 0 \le n \le N-1\right\},$$

where π_{M+1} denotes the space of all polynomials of degree $M+1$. We assume that Y_n is the initial condition for the interval I_n, i.e., $y(t_n) = Y_n$, which is obtained by the previous step or initial condition, and set an initial guess as $Y_n^{[0]}(t) \equiv Y_n$. We then make a transform from I_n to I and adopt the numerical Picard iteration (5) to obtain

$$Y_n^{[i+1]}(t) = Y_n + \frac{h_n}{2}\int_{-1}^{\frac{2}{h_n}(t-t_n)-1} \mathcal{I}_M(f(\mathbf{Y}_n^{[i]}), s)ds, \quad i = 0, 1, \ldots, \tag{6}$$

where

$$\begin{aligned}
&t_{n,i} := t_n + \frac{h_n}{2}(c_i + 1), \quad i = 0, 1, \ldots, M.\\
&\mathbf{Y}_n^{[i]} := [Y_n^{[i]}(t_{n,0}), Y_n^{[i]}(t_{n,1}), \ldots, Y_n^{[i]}(t_{n,M})]^\top,\\
&f(\mathbf{Y}_n^{[i]}) := [f(t_{n,0}, Y_n^{[i]}(t_{n,0})), f(t_{n,1}, Y_n^{[i]}(t_{n,1})), \ldots, f(t_{n,M}, Y_n^{[i]}(t_{n,M}))]^\top.
\end{aligned}$$

We are now ready to define the numerical Picard iteration solution on the entire domain: for a given positive number J, the element $\eta_h^{[J]} \in S_{M+1}(\mathcal{M}_h)$ is called the J-Picard solution of Eq. (1) if it satisfies

$$\eta_h^{[J]}|_{I_n} = Y_n^{[J]}, n = 0, 1, \ldots, N-1,$$

where $Y_n^{[J]}$ is generated by Eq. (6).

Remark 1 Different kinds of collocation points have been proposed in the numerical Picard iteration methods. For the special case in which Chebyshev points are used, i.e.,

$$c_i = \cos\frac{i\pi}{M}, \quad i = 0, 1, \ldots, M,$$

the method is called the modified Chebyshev-Picard iteration method [2].

To easily present the numerical Picard iteration method, we next introduce the integration matrix associated with the points $\{c_i : i = 0, 1, \ldots, M\}$: given a vector $\boldsymbol{\varphi} \in \mathbb{C}^{M+1}$ and $\boldsymbol{\phi} = [\phi_0, \phi_1, \ldots, \phi_M]^\top$ that is given by

$$\phi_i = \int_{-1}^{c_i} \mathcal{I}_M(\boldsymbol{\varphi}, s)ds, \quad i = 0, 1, \ldots, M,$$

we define the linear mapping $\mathcal{S}_M : \mathbb{C}^{M+1} \to \mathbb{C}^{M+1}$ by

$$\boldsymbol{\phi} = \mathcal{S}_M(\boldsymbol{\varphi}).$$

With these notations, we summarize the numerical Picard iteration method as follows.

Algorithm 1: Numerical Picard iteration method

Input: Domain $[a, b]$, initial condition y_a, number of intervals N, number of points on each interval M, number of Picard iterations J

Output: Evaluations on the endpoints of N intervals $\boldsymbol{\eta} := [\eta_0^{[J]}, \eta_1^{[J]}, \ldots, \eta_N^{[J]}]^\top$

(Initialize and pre-compute integration matrix);

Set $\eta_0^{[J]} = y_a$ and derive the matrix $\mathcal{S}_M$;

for $n = 1$ *to* N **do**

  $\mathbf{Y}_n = [\eta_{n-1}^{[J]}, \eta_{n-1}^{[J]}, \ldots, \eta_{n-1}^{[J]}]^\top$;

  Set the initial approximate solution $\mathbf{Y}_n^{[0]} = \mathbf{Y}_n$;

  (Picard iteration)

  for $j = 1$ *to* J **do**

$$\mathbf{Y}_n^{[j]} = \mathbf{Y}_n + \frac{h_n}{2}\mathcal{S}_M f(\mathbf{Y}_n^{[j-1]}), \tag{7}$$

  end

  If $c_M = 1$, the approximation $\eta_n^{[J]}$ at $t = t_n$ is included in $\mathbf{Y}_n^{[J]}$.

  If $c_M \neq 1$, derive the approximation at $t = t_n$

$$\eta_n^{[J]} = \eta_{n-1}^{[J]} + \frac{h_n}{2}\boldsymbol{\omega}' f(\mathbf{Y}^{[J-1]}),$$

  where $\boldsymbol{\omega} = (\omega_0, \omega_1, \ldots, \omega_M)^\top$ and $\omega_i := \int_{-1}^{1} \ell_i(s)ds$, $i = 0, 1, \ldots, M$.

end

According to (7) in the numerical Picard iteration method, it is clear that the values $\boldsymbol{Y}_n^{[j]}$ at different nodes in I_n depend only on the values $\boldsymbol{Y}_n^{[j-1]}$ in the previous iteration; hence, the numerical Picard iteration method refreshes the state values segment by segment, which means that the values $\boldsymbol{Y}_n^{[j]}$ can be obtained simultaneously. This is quite different from the SDC methods, in which the values at different nodes must be computed point by point. This property has the merit that the matrix–vector product and the evaluations of the forcing function f are easily computed in parallel in one numerical Picard iteration.

We next present the convergence analysis of the above numerical Picard method. For this purpose, we recall the following interpolation error in terms of the Peano kernel theorem [23].

Lemma 1 *If* $\varphi \in C^d(\bar{I}_n)$, $1 \leq d \leq M+1$, *then the Lagrange interpolant defined on the set* $\{t_{n,j}\}$ *satisfies*

$$\varphi\left(t_n + \frac{h_n}{2}(s+1)\right) - \mathcal{I}_M(\boldsymbol{\varphi}, s) = \left(\frac{h_n}{2}\right)^d \int_{-1}^{1} K_d(s,z)\varphi^{(d)}\left(t_n + \frac{h_n}{2}(z+1)\right)dz,$$

where $\boldsymbol{\varphi} := [\varphi(t_{n,0}), \varphi(t_{n,1}), \ldots, \varphi(t_{n,M})]^\top$ *and*

$$K_d(s,z) := \frac{1}{(d-1)!}\left\{(s-z)_+^{d-1} - \sum_{j=0}^{M} \ell_j(s)(c_j - z)_+^{d-1}\right\},$$

with $(s-z)_+^p := 0$ *for* $s < z$ *and* $(s-z)_+^p := (s-z)^p$ *for* $s \geq z$.

We also have the following lemma.

Lemma 2 *Suppose that N and m are non-negative integers and h is a positive real number such that Nh is a constant independent of N and h. If the sequence $\{\epsilon_n : 0 \le n \le N\}$ satisfies $\epsilon_0 = 0$ and*

$$\epsilon_{n+1} \le c_0 h^m + (1 + c_1 h)\epsilon_n, \, 1 \le n \le N-1,$$

where c_0 and c_1 are positive numbers independent of h, then there exists a positive number c independent of h such that

$$\epsilon_n \le c h^{m-1}, \, n \le N.$$

Proof It is obtained through iteration that

$$\begin{aligned}\epsilon_n &\le c_0 h^m \sum_{i=0}^{n-1} (1 + c_1 h)^i = \frac{c_0 h^{m-1}}{c_1} \left((1 + c_1 h)^n - 1\right) \\ &\le \frac{c_0}{c_1} \left(e^{c_1 n h} - 1\right) h^{m-1}, \, 1 \le n \le N,\end{aligned}$$

where we use the inequality $1 + x \le e^x$. Since Nh is a constant independent of h, the conclusion follows directly. □

We are now ready to present the following convergence analysis.

Theorem 1 *Assume that the solution $y(t)$ of Eq. (1) is $(M+2)$-times continuously differentiable and $\eta_h^{[J]}$ is the J-Picard solution. For sufficiently small h, the following error estimate holds:*

$$\left\| y - \eta_h^{[J]} \right\|_\infty = \max_{t \in [a,b]} \left| y(t) - \eta_h^{[J]}(t) \right| \le C h^{\min(J, M+1)}, \tag{8}$$

where the constant C is independent of h.

Proof We denote $\mathbf{y}_n = [y(t_{n,0}), y(t_{n,1}), \ldots, y(t_{n,M})]^\top$. By Lemma 1, there exists

$$y'\left(t_n + \frac{h_n}{2}(s+1)\right) = f\left(t_n + \frac{h_n}{2}(s+1), y\left(t_n + \frac{h_n}{2}(s+1)\right)\right) = \mathcal{I}_M(f(\mathbf{y}_n), s) + \left(\frac{h_n}{2}\right)^{M+1} H_n(s), \tag{9}$$

where

$$H_n(s) = \int_{-1}^{1} K_J(s, z) y^{(M+2)}\left(t_n + \frac{h_n}{2}(z+1)\right) dz.$$

Integration of Eq. (9) leads to

$$y\left(t_n + \frac{h_n}{2}(s+1)\right) = y(t_n) + \frac{h_n}{2} \int_{-1}^{s} \mathcal{I}_M(f(\mathbf{y}_n), z) dz + \left(\frac{h_n}{2}\right)^{M+2} \int_{-1}^{s} H_n(z) dz. \tag{10}$$

Recalling the formula for $Y_n^{[J]}$ in the Picard iteration, we rewrite $e_h(t) := y(t) - \eta_h^{[J]}(t)$ as

$$e_h\left(t_n + \frac{h_n}{2}(s+1)\right) = e_h(t_n) + \frac{h_n}{2} \int_{-1}^{s} \mathcal{I}_M(f(\mathbf{y}_n) - f(\mathbf{Y}_n^{[J-1]}), z) dz + \left(\frac{h_n}{2}\right)^{M+2} \int_{-1}^{s} H_n(z) dz.$$

Using the Lipschitz condition, we have

$$|e_h\left(t_n + \frac{h_n}{2}(s+1)\right)| \le |e_h(t_n)| + c h_n \left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\| \sum_{i=0}^{M} |\beta_i(s)| + c h_n^{M+2}, \tag{11}$$

where $\beta_i(s) = \int_{-1}^{s} \ell_i(z)dz$ and c is a generic constant independent of h. In particular, setting $s = 1$, the following is obtained:

$$|e_h(t_{n+1})| \leq |e_h(t_n)| + ch_n \left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\| + ch_n^{M+2}. \tag{12}$$

We next analyze the bounds of $e_h(t_n)$ and $\left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\|$. Combining Eqs. (6) and (10), we have

$$\left\| \mathbf{y}_n - \mathbf{Y}_n^{[j+1]} \right\| \leq |e_h(t_n)| + ch_n \left\| \mathbf{y}_n - \mathbf{Y}_n^{[j]} \right\| + ch_n^{M+2}, \; j = 0, 1, \ldots, J-2.$$

Note that we also have

$$\left\| \mathbf{y}_n - \mathbf{Y}_n^{[0]} \right\| \leq |e_h(t_n)| + \max_{i=0,1,\ldots,M} |y(t_{n,i}) - y(t_n)| \leq |e_h(t_n)| + ch_n.$$

It then follows that

$$\begin{aligned} \left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\| &\leq \left(|e_h(t_n)| + ch_n^{M+2} \right) \sum_{j=0}^{J-2} (ch_n)^j + (ch_n)^{J-1} \left\| \mathbf{y}_n - \mathbf{Y}_n^{[0]} \right\| \\ &\leq |e_h(t_n)| \sum_{j=0}^{J-1} (ch_n)^j + ch_n^{\min(J,M+2)}. \end{aligned} \tag{13}$$

By substituting (13) into Eq. (12), we obtain

$$|e_h(t_{n+1})| \leq |e_h(t_n)| \sum_{j=0}^{J} (ch_n)^j + ch_n^{\min(J+1,M+2)} \leq |e_h(t_n)| \sum_{j=0}^{J} (ch)^j + ch^{\min(J+1,M+2)}. \tag{14}$$

For a sufficiently small h, there exists a constant c_1 independent of h such that

$$\sum_{j=0}^{J} (ch)^j \leq 1 + c_1 h.$$

Moreover, we have that $e_h(t_0) = 0$. By Lemma 2, we have that

$$e_h(t_n) \leq ch^{\min(J,M+1)}, \quad n = 1, 2, \ldots, N.$$

It then follows directly from (13) that

$$\left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\| \leq ch^{\min(J,M+1)}.$$

Then, the desired error bound (8) follows by using the bounds of $e_h(t_n)$, $\left\| \mathbf{y}_n - \mathbf{Y}_n^{[J-1]} \right\|$, and the equality (11). □

Next, we show that the J-Picard solution yields a super-convergence property when the Gaussian quadrature points are adopted. For this purpose, let

$$\delta_h^{[J]}(t) := -\eta_h^{[J]\prime}(t) + f(t, \eta_h^{[J]}(t)), \quad t \in [a, b]$$

be the defect associated with the J-Picard solution. We first present the following lemma.

Lemma 3 *If f is bounded and satisfies the Lipschitz condition, then there exists a constant C independent of h such that*

$$\max_{\substack{i=0,1,\ldots,M \\ n=0,1,\ldots,N-1}} \left|\delta_h^{[J]}(t_{n,i})\right| \leq Ch^J.$$

Proof Let $\eta_h^{[J]}(t) = Y_n^{[J]}(t)$, $t \in I_n$ and $\{Y_n^{[j]} : j = 0, 1, \ldots, J\}$ be the sequences generated by the Picard iteration on I_n. For $t \in \{t_{n,i} : i = 0, 1, \ldots, M\}$, $\delta_h^{[J]}$ can be represented by

$$\delta_h^{[J]}(t) = f(t, Y_n^{[J]}(t)) - f(t, Y_n^{[J-1]}(t)).$$

Using the Lipschitz condition of f, there exists a constant independent of h such that

$$\left|\delta_h^{[J]}(t_{n,i})\right| \leq C \left|Y_n^{[J]}(t_{n,i}) - Y_n^{[J-1]}(t_{n,i})\right|, \quad i = 0, 1, 2 \cdots, M. \tag{15}$$

Let $\mathbf{Y}_n^{[j]}$ be the vector of the function $Y_n^{[j]}$ evaluated at the points $\{t_{n,i} : i = 0, 1, \ldots, M\}$. It is noted from the Picard iteration that $Y_n^{[0]}(t_{n,i}) = \eta_h^{[J]}(t_n)$ and

$$Y_n^{[j+1]}(t_{n,i}) = \eta_h^{[J]}(t_n) + \frac{h_n}{2} \int_{-1}^{c_i} \mathcal{I}_M(f(\mathbf{Y}_n^{[j]}), s)ds, \quad j = 0, 1, \ldots, J-1.$$

Hence, there exists a constant C independent of h such that

$$\left\|\mathbf{Y}_n^{[j+1]} - \mathbf{Y}_n^{[j]}\right\| \leq Ch_n \left\|\mathbf{Y}_n^{[j]} - \mathbf{Y}_n^{[j-1]}\right\|, \quad j = 1, 2, \ldots, J-1,$$

and $\left\|\mathbf{Y}_n^{[1]} - \mathbf{Y}_n^{[0]}\right\| \leq Ch_n$. It follows directly that

$$\left\|\mathbf{Y}_n^{[J]} - \mathbf{Y}_n^{[J-1]}\right\| \leq Ch_n^J. \tag{16}$$

The desired inequality follows directly from (15) and (16). The proof is completed. □

We also recall the following standard lemma for the Gauss quadrature error [1].

Lemma 4 *If $\varphi \in C^{2M+2}(\bar{I}_n)$, and the points $\{c_j : j = 0, 1, \ldots, M\}$ are chosen to be Legendre–Gauss points, then the error*

$$E_M(\varphi) = \int_{t_n}^{t_{n+1}} \varphi(t)dt - \frac{h_n}{2} \sum_{j=0}^{M} \omega_j \varphi(t_{n,j})$$

satisfies

$$|E_M(\varphi)| \leq Ch_n^{2M+3},$$

where C is a constant independent of h_n.

We are now ready to present the super-convergence result.

Theorem 2 *Assume that the solution $y(t)$ of (1) is $(2M+3)$-times continuously differentiable and $\eta_h^{[J]}$ is a J-Picard solution. If the step size h is sufficiently small and the points $\{c_j\}_{j=0}^M$ are chosen to be Legendre–Gauss points, then the following error estimate holds:*

$$\max_{t \in \mathcal{M}_h} |y(t) - \eta_h^{[J]}(t)| \leq Ch^{\min(J, 2M+2)}, \tag{17}$$

where the constant C is independent of h.

Proof By Theorem 1, it holds that $\|e_h\| := \|y(t) - \eta_h^{[J]}(t)\| \leq Ch^{\min(J,M+1)}$. Moreover, we have

$$e_h'(t) = f(t, y(t)) - f(t, \eta_h^{[J]}(t)) + \delta_h^{[J]}(t) = f_y(t, y(t))e_h(t) + \delta_h^{[J]}(t) + R_h(t), \ t \in [a, b] \tag{18}$$

with $e_h(a) = 0$ and $\|R_h\| \leq Ch^{\min(2J,2M+2)}$. We then have

$$e_h(t) = \int_a^t r(t, s)(\delta_h^{[J]}(s) + R_h(s))ds, t \in [a, b],$$

where $r(t, s)$ denotes the resolvent of the ODE (18):

$$r(t, s) := exp\left(\int_s^t f_y(z, y(z))dz\right), \text{ with } r \in C^{2M+2}(D),$$

where $D := \{(t, s) : a \leq s \leq t \leq b\}$. The error $e_h(t_n)$ can be written as

$$e_h(t_n) = \sum_{j=0}^{n} \int_{t_j}^{t_{j+1}} r(t_n, s)\delta_h^{[J]}(s)ds.$$

Supposing that each of the integrals is approximated by the Gaussian quadrature based on the Legendre Gauss points, we have

$$\int_{t_j}^{t_{j+1}} r(t_n, s)\delta_h^{[J]}(s)ds = \frac{h_j}{2} \sum_{k=0}^{M} \omega_k r(t_n, t_{n,k})\delta_h^{[J]}(t_{n,k}) + E_{M,n},$$

where the term $E_{M,n}$ denotes the quadrature error. Then, Lemma 4 indicates that $E_{M,n}$ is bounded by Ch^{2M+3},. Furthermore, Lemma 3 shows that

$$\frac{h_j}{2} \sum_{k=0}^{M} \omega_k r(t_n, t_{n,k})\delta_h^{[J]}(t_{n,k}) \leq Ch^{J+1},$$

where C is a generic constant independent of h. Hence, the integral $\int_{t_j}^{t_{j+1}} r(t_n, s)\delta_h^{[J]}(s)ds$ is bounded by $Ch^{\min(J+1,2M+3)}$, which completes the proof. □

3 Parallel Numerical Picard Iteration Methods

In addition to the parallelization in one Picard iteration due to the property of computation segment by segment, we further investigate the parallelization between the numerical Picard iterations on different sub-intervals. We propose a new parallel method called the parallel numerical Picard iteration (PNPI) method. Note that in the traditional numerical Picard iteration method one must complete the Picard iterations on the current interval before moving on to the next one. Instead, the proposed PNPI method allows one to perform Picard iterations simultaneously on different sub-intervals, once rough initial conditions and approximations are obtained.

We first illustrate one parallelization idea by considering a case with $N = 4$ and $J = 3$. We denote by $\eta_n^{[j]}$ the jth approximation value at t_n for all $n = 0, 1, \ldots, N$, $j = 0, 1, \ldots, J-1$,, and set

$$\eta_0^{[j]} = y_a, \quad j = 0, 1, \ldots, J - 1.$$

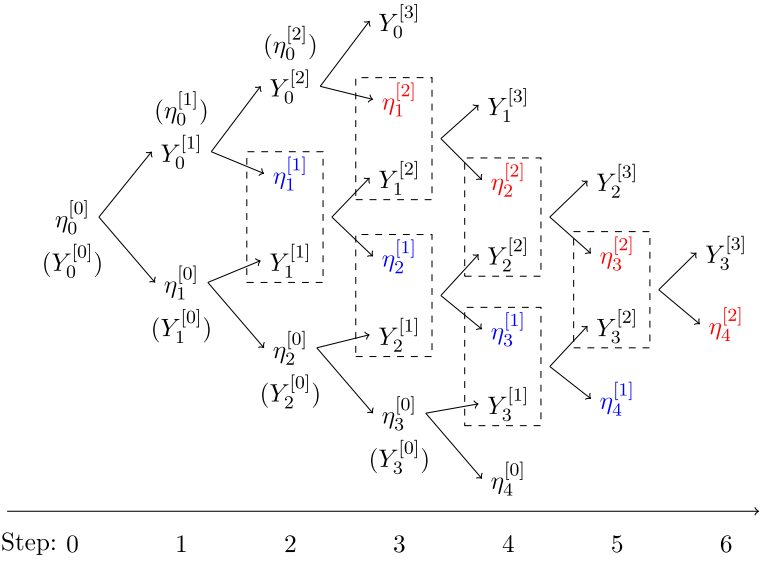


Fig. 1 Diagrams of parallel algorithm

The notation $Y_n^{[j]}$ denotes the jth approximation of solution on the sub-interval I_n. The computation consists of the following steps (See Fig. 1):

- On the first sub-interval I_0, the initial-state value $\eta_0^{[0]}$ is known and a rough approximation $Y_0^{[0]}$ is obtained by setting $Y_0^{[0]}(t) \equiv \eta_0^{[0]}, t \in I_0$. One numerical Picard iteration can be performed to obtain a more accurate solution $Y_0^{[1]}$ on I_0 and an initial-state value $\eta_1^{[0]}$ for the second sub-interval I_1.
- The second numerical Picard iteration can proceed on I_0 with $\eta_0^{[1]}$ and $Y_0^{[1]}$ to obtain a more accurate solution $Y_0^{[2]}$ for I_0 and a more accurate initial-state value $\eta_1^{[1]}$ for I_1. Meanwhile, another numerical Picard iteration can be conducted on I_1 with $\eta_1^{[0]}$ and $Y_1^{[0]} \equiv \eta_1^{[0]}$ to obtain a more accurate approximation $Y_1^{[1]}$ for I_1 and an initial-state value $\eta_2^{[0]}$ for I_2.
- We can next conduct three numerical Picard iterations on I_0, I_1, and I_2 simultaneously to obtain a more accurate approximation for the current sub-interval and provide a more accurate initial-state value for the next sub-interval.
- The procedure can continue until the computations are finished.

We note that there are J numerical Picard iterations that can be computed in parallel all the time, except at the beginning and last $J-1$ steps.

However, the parallelization of the numerical Picard iterations shown in Fig. 1 leads to bad convergence of the iteration. To explain this, we focus on the second Picard iteration on I_1 with the initial-state condition $\eta_1^{[1]}$ and a rough approximation $Y_1^{[1]}$. Although $\eta_1^{[1]}$ is more accurate than $\eta_1^{[0]}$, it is easily found that the accuracy of $Y_1^{[1]}$ is of the same level as that of $\eta_1^{[0]}$ according to (6); that is,

$$Y_1^{[1]}(t) = \eta_1^{[0]} + \frac{h_1}{2} \int_{-1}^{\frac{2}{h_1}(t-t_1)-1} \mathcal{I}_M(f(\mathbf{Y}_1^{[0]}), s)ds.$$

Then, the accuracy of $\eta_2^{[1]}$ and $Y_1^{[2]}$ obtained from the numerical Picard iteration will be limited by the accuracy of $Y_1^{[1]}$, i.e., that of $\eta_1^{[0]}$. In fact, it can also be proved that the convergence of the numerical Picard iteration method with the above parallelization is only of one order. To overcome bad convergence, we make a slight modification for $Y_1^{[1]}$ by adding the difference between $\eta_1^{[0]}$ and $\eta_1^{[1]}$,

$$\tilde{Y}_1^{[1]} = Y_1^{[1]} + \eta_1^{[1]} - \eta_1^{[0]},$$

and then conduct the numerical Picard iteration with $\eta_1^{[1]}$ and $\tilde{Y}_1^{[1]}$ to derive $\eta_2^{[1]}$ and $Y_1^{[2]}$. It will be shown in Theorem 3 that the convergence order stays the same as that of the serial numerical Picard iteration methods.

We are now ready to present parallel numerical Picard iteration methods that make use of both the merits of the parallelization and the high-order convergence.

Definition 1 For a given positive number J, the element $\eta_h^{[J]} \in S_{M+1}(\mathcal{M}_h)$ is called the parallel J-Picard solution of Eq. (1) if it satisfies

$$\eta_h^{[J]}\Big|_{I_n} = Y_n^{[J]}, \quad n = 0, 1, \ldots, N-1,$$

where $Y_n^{[J]}$ is generated by the iteration

$$\tilde{Y}_n^{[i]}(t) = \begin{cases} Y_n^{[i]}(t), i = 0, \\ Y_n^{[i]}(t) + \eta_n^{[i]} - \eta_n^{[i-1]}, \quad i = 1, 2, \ldots, J-1, \end{cases} \tag{19}$$

$$Y_n^{[i+1]}(t) = \eta_n^{[i]} + \frac{h_n}{2} \int_{-1}^{\frac{2}{h_n}(t-t_n)-1} \mathcal{I}_M(f(\tilde{\mathbf{Y}}_n^{[i]}), s) ds, i = 0, 1, \ldots, J-1, \tag{20}$$

with $\eta_0^{[j]} = y_a$, $j = 0, 1, \ldots, J-1$, $\eta_k^{[j]} = Y_{k-1}^{[j+1]}(t_k)$, $k = 1, \ldots, N-1$, $j = 0, 1, \ldots, J-1$, and $Y_k^{[0]}(t) \equiv \eta_k^{[0]}$, $t \in I_k$, $k = 0, 1, \ldots, N-1$.

With the above definition, the PNPI method is presented in Algorithm 2. It is clear from Algorithm 2 that there is no restriction on the selection of points in each subinterval for the PNPI method. It is more flexible for the PNPI method than the RICD method [8] which requires the uniform nodes for easily conducting the parallelization.

The convergence analysis of the PNPI method is given in Theorem 3.

Theorem 3 *Assuming that the solution $y(t)$ of (1) is $(M+2)$-times continuously differentiable and that $\eta_h^{[J]}$ is a parallel J-Picard solution, if the step size h is sufficiently small, then the following error estimate holds:*

$$\left\| y - \eta_h^{[J]} \right\|_\infty = \max_{t \in [a,b]} \left| y(t) - \eta_h^{[J]}(t) \right| \le c h^{\min\{J, M+1\}}, \tag{21}$$

where the constant c is independent of h.

Algorithm 2: Parallel Numerical Picard Iteration Method

Input: Domain $[a, b]$, initial condition y_a, number of intervals N, number of points on each interval M, number of Picard iterations J

Output: Evaluations on the endpoints of N intervals $\boldsymbol{\eta} := [\eta_0^{[J]}, \eta_1^{[J]}, \ldots, \eta_N^{[J]}]^\top$

(Initialize and precompute integration matrix);
Set $\eta_0^{[j]} = y_a$, $j = 1, 2\cdots, J$, and derive the matrix $\mathcal{S}_M$;
for $k = 2$ *to* $N + 1$ **do**
  for $j = 1$ *to* $\min(k - 1, J)$ **do**
    (Parallel Computation)
    $n = k - j$;
    $\mathbf{y}_n := [\eta_{n-1}^{[j]}, \eta_{n-1}^{[j]}, \ldots, \eta_{n-1}^{[j]}]^\top$;
    if $j = 1$ **then**
      $\mathbf{Y}_n^{[0]} = \mathbf{y}_n$;
      $\Delta\mathbf{y}_n := [0, 0, \ldots, 0]^\top$;
    else
      $\Delta\mathbf{y}_n := [\eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}, \eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}, \ldots, \eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}]^\top$;
    end
    $\mathbf{Y}_n^{[j]} = \mathbf{y}_n + \frac{h_n}{2}\mathcal{S}_M f(\mathbf{Y}_n^{[j-1]} + \Delta\mathbf{y}_n)$;
    $\eta_n^{[j]} = \eta_{n-1}^{[j]} + \frac{h_n}{2}\boldsymbol{\omega}' f(\mathbf{Y}_n^{[j-1]} + \Delta\mathbf{y}_n)$;
  end
end
for $k = N + 2$ *to* $N + J$ **do**
  for $n = \max(1, k - J)$ *to* N **do**
    (Parallel Computation)
    $j = k - n$;
    $\mathbf{y}_n := [\eta_{n-1}^{[j]}, \eta_{n-1}^{[j]}, \ldots, \eta_{n-1}^{[j]}]^\top$;
    if $j = 1$ **then**
      $\mathbf{Y}_n^{[0]} = \mathbf{y}_n$;
      $\Delta\mathbf{y}_n := [0, 0, \ldots, 0]^\top$;
    else
      $\Delta\mathbf{y}_n := [\eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}, \eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}, \ldots, \eta_{n-1}^{[j]} - \eta_{n-1}^{[j-1]}]^\top$;
    end
    $\mathbf{Y}_n^{[j]} = \mathbf{y}_n + \frac{h_n}{2}\mathcal{S}_M f(\mathbf{Y}_n^{[j-1]} + \Delta\mathbf{y}_n)$;
    $\eta_n^{[j]} = \eta_{n-1}^{[j]} + \frac{h_n}{2}\boldsymbol{\omega}' f(\mathbf{Y}_n^{[j-1]} + \Delta\mathbf{y}_n)$;
  end
end

Proof Denoting the error $e_n^{[i]}(t) := y(t) - Y_n^{[i]}(t)$ and $\xi_n^{[i]} := y(t_n) - \eta_n^{[i]}$, $n = 0, 1, \ldots, N-1$, $i = 0, 1, \ldots J - 1$, it is obtained that

$$e_n^{[i+1]}\left(t_n + \frac{h_n}{2}(s+1)\right) = \xi_n^{[i]} + \frac{h_n}{2}\int_{-1}^{s} \mathcal{I}_M(f(\mathbf{y}_n) - f(\tilde{\mathbf{Y}}_n^{[i]}), z)dz + \left(\frac{h_n}{2}\right)^{M+2}\int_{-1}^{s} H_n(z)dz,$$

with $i = 0, 1, \ldots, J-1$, $n = 0, 1, \ldots, N-1$ and $s \in [-1, 1]$. Using the Lipschitz condition, we have

$$\left| e_n^{[i+1]} \left(t_n + \frac{h_n}{2}(s+1) \right) \right| \leq \left| \xi_n^{[i]} \right| + ch \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i]} \right\| + ch^{M+2}, \tag{22}$$

where c is a generic constant independent of h. In particular, we have from Eq. (22) that

$$\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i+1]} \right\| \leq \left| \xi_n^{[i+1]} \right| + ch \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i]} \right\| + ch^{M+2}. \tag{23}$$

Noting that $\xi_{n+1}^{[i]} = e_n^{[i+1]}(t_{n+1})$, $n = 0, 1, 2, \ldots, N-1$, $i = 0, 1, \ldots, J-1$, it follows from Eq. (22) that

$$\left| \xi_{n+1}^{[i]} \right| \leq \left| \xi_n^{[i]} \right| + ch \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i]} \right\| + ch^{M+2}. \tag{24}$$

We next present the bounds for $\xi_n^{[0]}$ and $\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\|$. Since

$$\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\| \leq \| \mathbf{y}_n - y(t_n) \| + \left| \xi_n^{[0]} \right| \leq ch + \left| \xi_n^{[0]} \right|,$$

we obtain from Eq. (24) that

$$\left| \xi_{n+1}^{[0]} \right| \leq (1 + ch) \left| \xi_n^{[0]} \right| + ch^2 + ch^{M+2}, n = 0, 1, \ldots, N-1. \tag{25}$$

Since $\xi_0^{[0]} = 0$, Lemma 2 shows that

$$\max_{n=0,1,\ldots,N-1} \left| \xi_n^{[0]} \right| \leq ch, \text{ and } \max_{n=0,1,\ldots,N-1} \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\| \leq ch. \tag{26}$$

We then turn to the bound of $\left| \xi_{n+1}^{[i]} \right|$, $i \geq 1$. With the combination of Eqs. (23) and (24), it is obtained by induction that

$$\left| \xi_{n+1}^{[i]} \right| \leq \left| \xi_n^{[i]} \right| + \sum_{j=1}^{i} (ch)^{i+1-j} \left(\left| \xi_n^{[j]} \right| + (ch)^{M+2} \right) + (ch)^{i+1} \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\| + (ch)^{M+2}. \tag{27}$$

Combining the bound (26) for $\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\|$ and Eq. (27), we have, for a small enough h, that

$$\left| \xi_{n+1}^{[i]} \right| \leq (1 + ch) \left| \xi_n^{[i]} \right| + \sum_{j=1}^{i-1} (ch)^{i+1-j} \left| \xi_n^{[j]} \right| + ch^{\min\{i+2, M+2\}}. \tag{28}$$

Combining the inequality (28) and the fact that $\xi_0^{[j]} = 0$, $j = 0, 1, \ldots, J-1$, it can be verified by induction on i with the help of Lemma 2 that

$$\max_{n=1,2,\ldots,N-1} \left| \xi_n^{[i]} \right| \leq ch^{\min\{i+1, M+1\}}, \ i = 0, 1, \ldots, J-1. \tag{29}$$

Combining Eqs. (23) and (29), we have

$$\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i+1]} \right\| \leq ch \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i]} \right\| + ch^{\min\{i+2, M+1\}}.$$

By induction on i, it is obtained from $\left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[0]} \right\| < ch$ that

$$\max_{n=1,2,\ldots,N-1} \left\| \mathbf{y}_n - \tilde{\mathbf{Y}}_n^{[i]} \right\| \leq ch^{\min\{i+1, M+1\}}, \ i = 0, 1, \ldots, J. \tag{30}$$

It follows directly from Eqs. (22), (29), and (30) that

$$\left\| y - \eta_h^{[J]} \right\|_\infty = \max_{n=0,1,\ldots,N-1} \left| e_n^{[J]} \right| \le c h^{\min\{J, M+1\}},$$

which finishes the proof. □

We remark that the PNPI methods also possess the super-convergence property. We present this property as the following theorem, the proof of which is similar to that of Theorem 2.

Theorem 4 *Assuming that the solution $y(t)$ of (1) is $(2M+3)$-times continuously differentiable and $\eta_h^{[J]}$ is a parallel J-Picard solution, if the step size h is sufficiently small and the points $\{c_j : j = 0, 1, \ldots, M\}$ are chosen to be Legendre–Gauss points, then the following error estimate holds:*

$$\max_{t \in \mathcal{M}_h} |y(t) - \eta_h^{[J]}(t)| \le C h^{\min(J, 2M+2)}, \tag{31}$$

where the constant C is independent of h.

We end this section by presenting the speedup property of PNPI. To this end, we define the following index for speedup:

$$\text{Speedup} = \frac{T_{\text{serial}}}{T_{\text{parallel}}},$$

where T_{serial} and T_{parallel} are the computational times for the traditional method and PNPI method, respectively. We denote by $\mathbf{u}_0$ one unit time to evaluate one force function and by $\mathbf{u}_1$ one unit time to compute a dot product of two vectors of size $(M+1)$. To implement one numerical Picard iteration to obtain one $\mathbf{Y}_n^{[j]}$ and $\eta_n^{[j]}$, $(M+1)$ functions and $m(M+1)$ dot products must be evaluated, as shown in Algorithm 2. Since NJ Picard iterations in total are needed, it is clear that

$$T_{\text{serial}} = NJ(M+1)(\mathbf{u}_0 + m\mathbf{u}_1).$$

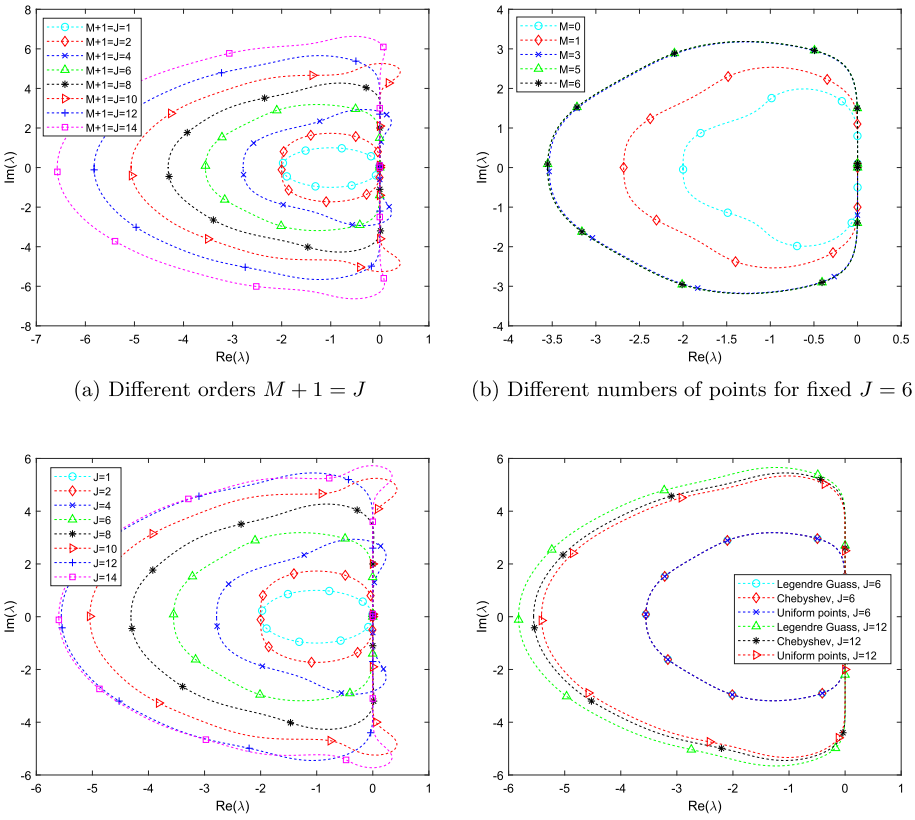
Supposing that P cores are adopted in Algorithm 2, for simplicity we assume that there are J numerical Picard iterations in each outer iteration. In each outer iteration, there are $(M+1)J$ functions and $m(M+1)J$ dot products to be evaluated independently, which can be computed in parallel. It then follows that

$$T_{\text{parallel}} \approx \frac{1}{P}(N + J - 1)(M+1)J(\mathbf{u}_0 + m\mathbf{u}_1).$$

Thus, the speedup of the parallel numerical Picard method is given by

$$\text{Speedup} = \frac{PN}{N + J - 1}.$$

Owing to the combination of parallelizations inside one numerical Picard iteration and between numerical Picard iterations on different sub-intervals, the parallel numerical Picard iteration method yields a larger speedup compared to the RIDC method [8]. In fact, the speedup of a Jth-order RIDC is at most J, while that for PNPI with the speedup can be $J(M+1)$, at least in theory.



(a) Different orders $M+1=J$ (b) Different numbers of points for fixed $J=6$

(c) Different iterations for fixed $M=5$ (d) Different kinds of points

Fig. 2 Stability regions for numerical Picard iteration methods. Dependence of stability on **a** orders, **b** number of points, **c** iteration, and **d** type of points

4 Stability Analysis

We now investigate the numerical stability region for the numerical Picard iteration method. First, we recall the following definitions.

Definition 2 The amplification factor for a numerical method, $Am(\lambda)$, can be interpreted as the numerical solution to Dalhquist's test equation,

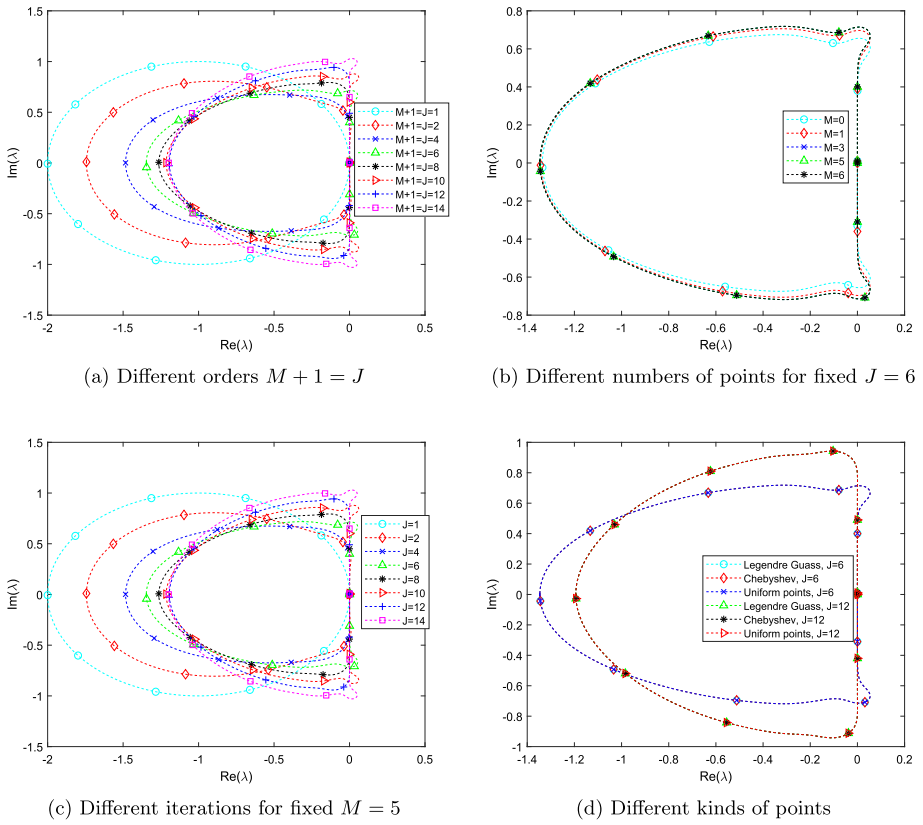
$$y'(t) = \lambda y(t), \;\; y(0) = 1, \tag{32}$$

after one time step of size 1 for $\lambda \in \mathbb{C}$, i.e., $Am(\lambda) = y(1)$.

Definition 3 The stability region S for a numerical method is the subset of the complex plane $\mathbb{C}$ consisting of all λ such that $Am(\lambda) \leq 1$,

$$S = \{\lambda : Am(\lambda) \leq 1\}.$$

The stability regions for the numerical Picard iteration methods with different settings are computed numerically and presented in Fig. 2. In Fig. 2a–c, we explore the dependence



(a) Different orders $M+1=J$ (b) Different numbers of points for fixed $J=6$

(c) Different iterations for fixed $M=5$ (d) Different kinds of points

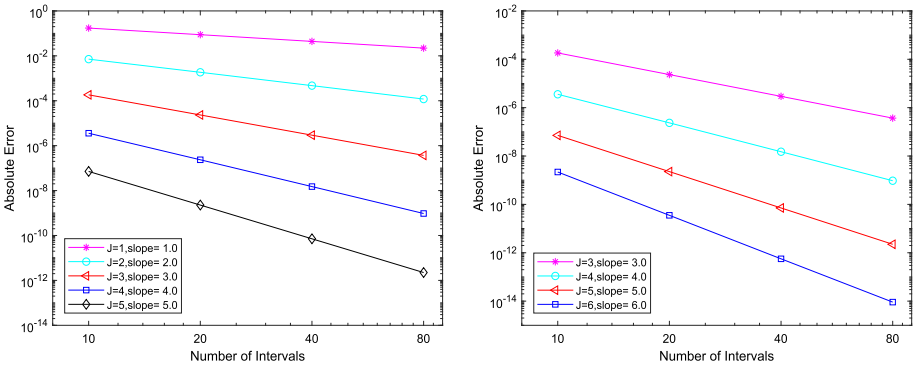
Fig. 3 Stability regions for parallel numerical Picard iteration methods. Dependence of stability on **a** orders, **b** number of points, **c** iteration, and **d** type of points

of the stability on the convergence order, number M of points, and iteration number J with Chebyshev points. One observes a circle of radius 1 for the stability region after one Picard iteration, which is the same as the forward Euler integrator used. This is shown clearly in Fig. 2a, i.e., the area of the stability regions increases with increasing convergence order of the numerical Picard iteration methods. When the convergence order is unchanged, Fig. 2b shows that the stability regions would not increase with increasing number M of points, while Fig. 2c shows that the stability regions still increase with increasing iteration number J before the value J attaches $2(M+1)$. We also study the dependence of the stability regions on the choice of points. We chose three different types of points: Chebyshev, Legendre–Gauss, and uniform points. Interestingly, FIg. 2d shows that the stability regions almost do not depend on the choice of the points.

To study the stability regions of the PNPI methods, we introduce another definition of the stability region based on a uniform mesh.

Definition 4 The stability region S for a numerical method with a uniform mesh that has a size of Δt is the subset of the complex plane $\mathbb{C}$ consisting of all $\lambda\Delta t$ such that $Am(\lambda) \leq 1$,

$$S = \{\lambda\Delta t : Am(\lambda) \leq 1\}.$$



(a) Five Chebyshev-Gauss nodes (b) Three Legendre-Gauss nodes

Fig. 4 Absolute error as function of number of intervals for different numbers of iterations, J,. NPM with J iterations adopts **a** five Chebyshev-Gauss nodes per interval and **b** three Legendre–Gauss nodes per interval. Expected orders of accuracy are observed

The stability regions for the parallel numerical Picard iteration methods with different settings are displayed in Fig. 3. In the first three Fig. 3a–c, we explore the dependence of the stability of the methods on the number M of points and the iteration number J while using Chebyshev points. One can also observe a circle of radius 1 for the stability region after one Picard iteration. Figure 3a shows that the area of the stability regions decreases with the convergence order of the numerical Picard iteration methods, but they encompass an increasing amount of the imaginary axis. This result is consistent with the stability regions of the RIDC [8]. We also note in Fig. 3b that the stability regions do not nearly depend on the number M of points, which is quite different from that for the numerical Picard iteration methods. The stability regions in Fig. 3c are nearly the same as those in Fig. 3a, which indicates that the stability regions depend mainly on the number of the iterations. Similarly, Fig. 3d shows that the stability regions do not nearly depend on the choice of points.

5 Numerical Examples

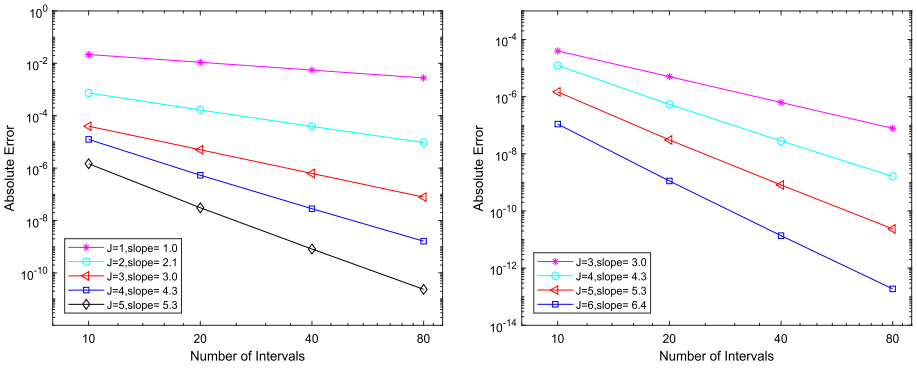
In this section, we illustrate the performance of numerical Picard iteration (NPI) and PNPI methods by numerical examples.

Example 1 Our first example is taken from [21]

$$\begin{cases} y_1'(t) = ty_2(t) + y_1(t), \\ y_2'(t) = -ty_1(t) + y_2(t), \\ y(0) = (1,1)^\top, t \in [0,T]. \end{cases} \tag{33}$$

The analytic solution is $y(t) = (e^t(\cos\frac{t^2}{2} + \sin\frac{t^2}{2}), e^t(\cos\frac{t^2}{2} - \sin\frac{t^2}{2}))^\top$. We use this example to show the convergence rate of NPI methods.

We set $T = 1$ and adopt five Chebyshev–Gauss points, i.e., $\{\cos\frac{i\pi}{n-1} : i = 0, 1, \ldots, n-1.\}$ in the NPI methods. The numerical errors with respect to the number of intervals is presented in Fig. 4a for different numbers of iterations. It is noted that the rates of convergence confirm the theoretical results. The super-convergence property with the Legendre–Gauss points is



(a) Five Chebyshev-Gauss nodes (b) Three Legendre-Gauss nodes

Fig. 5 Absolute error as function of number of intervals for different number of iterations, J. PNPI methods with J iterations adopting **a** five Chebyshev-Gauss nodes per interval and **b** three Legendre–Gauss nodes per interval. Expected orders of accuracy are observed

Table 1 Rate of convergence of NPI methods with five different nodes ($J = 5$, $T = 1$)

N	Chebyshev points		Uniform points		Irregular points	
	Error	Order	Error	Order	Error	Order
10	$7.1188e{-}08$	–	$1.1426e{-}07$	–	$2.2807e{-}07$	–
20	$2.2623e{-}09$	4.98	$3.6240e{-}09$	4.98	$7.2287e{-}09$	4.98
40	$7.1275e{-}11$	4.99	$1.1408e{-}10$	4.99	$2.2747e{-}10$	4.99
80	$2.2387e{-}12$	4.99	$3.5794e{-}12$	4.99	$7.1458e{-}12$	4.99
160	$7.0166e{-}14$	5.00	$1.1369e{-}13$	4.98	$2.4025e{-}13$	4.89

Table 2 Rate of convergence of NPI methods with 10 nodes ($J = 10$, $T = 6$)

N	Chebyshev points		Uniform points		Irregular points	
	Error	Order	Error	Order	Error	Order
10	$9.2939e{+}00$	–	$9.2920e{+}00$	–	$9.2528e{+}00$	–
20	$1.2409e{-}02$	9.55	$1.2407e{-}02$	9.55	$1.2361e{-}02$	9.55
40	$1.4560e{-}05$	9.74	$1.4561e{-}05$	9.73	$1.4739e{-}05$	9.71
80	$1.2176e{-}08$	10.22	$1.3068e{-}08$	10.12	$1.1963e{-}07$	6.94
160	$1.3188e{-}11$	9.85	$8.7309e{-}10$	3.90	$8.7145e{-}08$	0.46

also presented in Fig. 4b, where three Legendre–Gauss nodes are used. It is clearly shown in Fig. 4b that the order increases as the number of iterations increases before it reaches two times the number of nodes.

The NPI methods with different choice of collocation points were also tested. More precisely, we compared three different types of nodes: Chebyshev–Gauss points, uniformly spaced nodes, and irregularly spaced nodes. Table 1 shows the numerical errors and rates of convergence for the NPI methods M, $J = 5$, while Table 2 shows similar results with

Table 3 Comparison of convergence of PNPI and NPI methods with five Chebyshev points and $J = 5$

N	PNPI methods		NPI methods	
	Error	Order	Error	Order
10	1.4723e−06	–	1.1684e−06	–
20	3.0713e−08	5.58	3.1909e−08	5.19
40	8.1510e−10	5.24	9.2687e−10	5.11
80	2.3644e−11	5.11	2.7884e−11	5.05
160	7.1321e−13	5.05	8.5487e−13	5.03

Table 4 Rate of convergence of PNPI methods with 10 nodes ($J = 10$, $T = 5$)

N	Chebyshev points		Uniform points		Irregular points	
	Error	Order	Error	Order	Error	Order
20	5.3941e−04	–	5.3941e−04	–	5.3941e−04	–
40	1.7467e−08	14.91	1.7467e−08	14.91	1.7457e−08	14.92
80	6.2746e−12	11.44	6.1571e−12	11.47	3.0896e−11	9.14
160	3.8858e−15	10.66	3.9563e−13	3.96	2.8869e−11	0.10

$M, J = 10$. Here, the uniform points are chosen as

$$\left\{ -1 + \frac{2i}{n-1} : i = 0, 1, \ldots, n-1. \right\},$$

while the irregular points are selected as [−0.9 −0.6 −0.5 0.2 0.25] in Table 1 and as [−0.9 −0.8 −0.6 −0.55 −0.5 −0.15 0.2 0.225 0.25 0.4] in Table 2. It is noted that, when the numbers of nodes and iterations are not too large (Table 1), the NPI methods steadily yield a fifth-order rate of convergence of no matter what kinds of nodes are adopted. However, when the numbers of nodes and iterations are relatively large (Table 2), the NPI methods with irregular points and uniform points demonstrate a reduction of the convergence order, while the NPI methods with Chebyshev points behave more stably.

Example 2 We next consider the following nonlinear ODE system [8]

$$\begin{cases} y_1'(t) = -y_2(t) + y_1(t)(1 - y_1^2(t) - y_2^2(t)), \\ y_2'(t) = y_1(t) + 3y_2(t)(1 - y_1^2(t) - y_2^2(t)), \\ y(0) = (1, 0)^\top, t \in [0, T]. \end{cases} \tag{34}$$

The analytic solution is given by $y(t) = (\cos t, \sin t)^\top$. We use this example to validate the order of accuracy of PNPI methods. Figure 5a shows the error with respect to the number of intervals for different numbers of iterations. The desired order of accuracy is observed and the super-convergence of the PNPI methods is also observed in Fig. 5b when the Legendre–Gauss points are used. Comparison of the performance between the NPI and PNPI methods is presented in Table 3. The numerical result shows that the PNPI methods can preserve the convergence order of the NPI methods. As seen in Table 4, the PNPI methods exhibit similar behaviors as the NPI methods.

Example 3 The last example is the one-dimensional N-body problem [8]. Supposing that N_+ ions are uniformly spaced on the interval [0, 1] and that N_- electrons are also uniformly

Table 5 Performance of PNPI methods with different numbers of iterations and of cores for fixed numbers of intervals, $N = 1000$, and of points, $M + 1 = 12$, on each interval

J	Error	ST (s)	Cores	PT (s)	Speedup		Efficiency
					Theoretical	Actual	
2	$4.1069e{-}02$	154.88	2	59.86	2.00	2.59	1.29
4	$1.7261e{-}04$	238.93	4	60.37	3.99	3.96	0.99
6	$2.7795e{-}07$	358.56	6	63.59	5.97	5.64	0.94
8	$1.2337e{-}10$	477.79	8	73.88	7.94	6.47	0.81
10	$1.2218e{-}13$	597.19	10	79.23	9.91	7.54	0.75
12	$2.6756e{-}14$	716.68	12	87.02	11.86	8.24	0.69

spaced on the interval [0, 1], the motion of particles is then dominated by the equations

$$\begin{bmatrix} x_i^+ \\ v_i^+ \end{bmatrix}_t = \begin{bmatrix} v_i^+ \\ \frac{q_+}{m_+}\left(\sum_{j=1}^{N_+} \frac{q_+(x_i^+ - x_j^+)}{\sqrt{(x_i^+ - x_j^+)^2 + d^2}} + \sum_{j=1}^{N_-} \frac{q_-(x_i^+ - x_j^-)}{\sqrt{(x_i^+ - x_j^-)^2 + d^2}}\right) \end{bmatrix}, \ i = 1, \ldots, N_+,$$

$$\begin{bmatrix} x_i^- \\ v_i^- \end{bmatrix}_t = \begin{bmatrix} v_i^- \\ \frac{q_-}{m_-}\left(\sum_{j=1}^{N_+} \frac{q_+(x_i^- - x_j^+)}{\sqrt{(x_i^- - x_j^+)^2 + d^2}} + \sum_{j=1}^{N_-} \frac{q_-(x_i^- - x_j^-)}{\sqrt{(x_i^- - x_j^-)^2 + d^2}}\right) \end{bmatrix}, \ i = 1, \ldots, N_-,$$

where $t \in [0, T]$, $\{x_i^+, v_i^+\}_{i=1}^{N_+}$ and $\{x_i^-, v_i^-\}_{i=1}^{N_-}$ denote the locations and velocities of the ions and the electrons, respectively. The charge of each ion and electron is set to $q_+ = \frac{1}{N_+}$ and $q_- = -\frac{1}{N_-}$, respectively. The mass of each ion and electron is set to $m_+ = \frac{1000}{N_+}$ and $m_- = \frac{1}{N_-}$, respectively, which are physically reasonable mass ratios to work with. d is a regularization constant, which is set to 0.05 in this simulation. The initial conditions are set to

$$x_i^+(0) = \frac{i - 0.5}{N_+}, \quad v_i^+(0) = 0, \ i = 1, \ldots, N_+,$$
$$x_i^-(0) = \frac{i - 0.5}{N_-}, \quad v_i^-(0) = \sin(6\pi x_i^-(0)), \ i = 1, \ldots, N_-.$$

In this example, we set $T = 20$ and $N_+ = N_- = 200$, and the reference solution is obtained using the built-in solver ode45 in MatLab (MathWorks, Natick, MA, USA). We consider the maximum norm errors at the terminal time. For a fixed number of intervals, $N = 1000$, and a fixed number of collocation points, $M + 1 = 12$, in each interval, we tested the efficiency of the PNPI methods with different iterations J on different numbers of cores ranging from 2 to 12. The associated numerical results are presented in Table 5. It is noted that the error decays rapidly when the number of iterations, J, increases. To show the efficiency, we present both the CPU time of sequential computation (denoted ST) and of parallel computation (denoted PT) with different cores. We also introduce two indexes, speedup and efficiency, which are defined, respectively, by

$$\text{Speedup} = \frac{\text{ST}}{\text{PT}}, \quad \text{Efficiency} = \frac{\text{Speedup}}{\text{number of cores}}.$$

Table 6 Performance of PNPI methods with different numbers of cores for fixed numbers of intervals, $N = 1000$; of iterations, $J = 12$; and of points, $M + 1 = 12$, on each interval

No. of cores	Time (s)	Speedup		Efficiency
		Theoretical	Actual	
1	716.68	–	–	–
2	361.03	1.98	1.99	0.99
4	185.01	3.96	3.87	0.97
6	133.10	5.94	5.38	0.90
8	111.68	7.92	6.42	0.80
10	97.46	9.90	7.35	0.73
12	87.02	11.88	8.24	0.69
$ode45$	174.07	(Error=1.0819$e-13$)		

Table 7 Performance comparison between PNPI methods with $J = M + 1 = 8$ and RIDC8 based on forward Euler method with different cores

$Cores$	PNPI methods ($Error = 2.45e-10$)			RIDC8 ($Error = 5.47e-10$)		
	Time (s)	Speedup	Efficiency	Time (s)	Speedup	Efficiency
1	318.18	–	–	442.55	–	–
2	159.40	2.00	1.00	236.54	1.87	0.94
4	81.94	3.88	0.97	122.80	3.60	0.90
8	48.93	6.50	0.81	70.07	6.31	0.79
12	40.64	7.83	0.65	70.45	6.28	0.52

To achieve similar accuracy, number of intervals is set to $N = 1000$ for PNPI methods and to $N = 6000$ for RIDC8

The results are listed in Table 5. It is noted that the speedup well matches that of theory and that the parallel efficiency is approximately 69% when 12 cores are used. According to theory, the CPU time is expected to decrease if one increases the number of cores until up to $12J$. We also adopted different cores to implement the PNPI methods with $J = 12$ and $M + 1 = 12$ in parallel, and the resulting computation times are listed in Table 6. Speedup in this table is defined by

$$\text{Speedup} = \frac{\text{CPU time of one core}}{\text{CPU time of multiple cores}}.$$

Again, the results match the theoretical results very well and the efficiency can be maintained above 69%. As a comparison, we present the computation time of the built-in function ode45 in MatLab. It is noted that that the PNPI methods with more than six cores outperform the ode45 solver.

We also performed a parallel performance comparison between the PNPI methods and the eighth-order RIDC method (denoted RIDC8). We set $J = M + 1 = 8$ and $N = 1000$ for the PNPI methods. To achieve similar accuracy, the number of intervals was set as $N = 6000$ for RIDC8. The numerical results are shown in Table 7, from which it can be seen that the PNPI methods take less time to achieve comparable accuracy and exhibit better speedup and efficiency.

6 Concluding Remarks

In this paper, we propose a new class of parallel time integrators for initial-value problems based on Picard iterations. We demonstrate that the parallel solvers yield the same convergence rate as the traditional numerical Picard iteration methods. The main features of our approach are as follows. (1) Instead of computing the solution point by point (as in RIDC methods), the proposed methods proceed segment by segment. (2) The proposed approach leads to a higher speedup: the speedup is shown to be $J(M+1)$ (while the speedup of the Jth-order RIDC is, at most, J). (3) The proposed approach is applicable to non-uniform points, such as Chebyshev points. We also present a stability analysis and numerical examples to verify the theoretical findings. In future work, we plan to apply the proposed PNPI method to more challenging engineering problems.

Acknowledgements The authors would like to express their most sincere thanks to the anonymous referees for their valuable comments. This research was partially supported by the National Natural Science Foundation of China (Grant No. 12201635, 62231026, 12271523), the Natural Science Foundation of Hunan Province, China (Grant No. 2022JJ40541) and the Research Fund of National University of Defense Technology (Grant No. ZK19-19).

Funding The authors have not disclosed any funding.

Availability of Data and Materials The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Atkinson, K.: An Introduction to Numerical Analysis. Wiley, Hoboken (1989)
2. Bai, X., Junkins, J.L.: Modified Chebyshev–Picard iteration methods for orbit propagation. J. Astronaut. Sci. **58**(4), 583–613 (2011)
3. Brenan, K.E., Campbell, S.L., Petzold, L.R.: Numerical Solution of Initial Value Problems in Differential-Algebraic Equations. SIAM, Philadelphia (1995)
4. Burrage, K.: Parallel and Sequential Methods for Ordinary Differential Equations. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York (1995)
5. Butcher, J.: The Numerical Analysis of Ordinary Differential Equations: Runge–Kutta and General Linear Methods. Wiley, New Fork (1987)
6. Christlieb, A., Ong, B.W., Qiu, J.: Integral deferred correction methods constructed with high order Runge–Kutta integrators. Math. Comput. **79**(270), 761–783 (2010)
7. Christlieb, A.J., Haynes, R.D., Ong, B.W.: A parallel space-time algorithm. SIAM J. Sci. Comput. **34**(5), C233–C248 (2012)
8. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM J. Sci. Comput. **32**(2), 818–835 (2010)
9. Clenshaw, C.W.: The numerical solution of linear differential equations in Chebyshev series. Math. Proc. Camb. Philos. Soc. **53**, 134–149 (1957)
10. Clenshaw, C.W., Curtis, A.R.: A method for numerical integration on an automatic computer. Numer. Math. **2**(1), 197–205 (1960)
11. Clenshaw, C.W., Norton, H.J.: The solution of nonlinear ordinary differential equations in Chebyshev series. Comput. J. **6**(1), 88–92 (1963)
12. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT **40**(2), 241–266 (2000)

13. Gander, M.J.: 50 years of time parallel time integration. In: Carraro, T., Geiger, M., Körkel, S., Rannacher, R. (eds.) Multiple Shooting and Time Domain Decomposition Methods, pp. 69–113. Springer International Publishing, Cham (2015)
14. Gander, M.J., Liu, J., Wu, S.L., Yue, X., Zhou, T.: Paradiag: Parallel-in-Time Algorithms Based on the Diagonalization Technique (2020). arXiv:2005.09158
15. Gear, C.W.: Numerical Initial Problems in Ordinary Differential Equations. Prentice-Hall, Englewood Cliffs (1971)
16. Hairer, E., Nøsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I. Non-stiff Problems. Springer, Berlin (1993)
17. Lambert, J.D.: Numerical Methods for Ordinary Differential Equations. Wiley, New York (1991)
18. Lindelöf, E.: Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. C R Hebd Séances Acad Sci **114**, 454–457 (1894)
19. Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. C. R. Acad. Sci. Paris Sér. I-Math. **332**, 661–668 (2001)
20. Nievergelt, J.: Parallel methods for integrating ordinary differential equations. Commun. ACM **7**(12), 731–733 (1964)
21. Ong, B.W., Spiteri, R.J.: Deferred correction methods for ordinary differential equations. J. Sci. Comput. **83**(60), 1–29 (2020)
22. Ong, B.W., Schroder, J.B.: Applications of time parallelization. Comput. Vis. Sci. **23**, 1–4 (2020)
23. Peano, G.: Resto nelle formule di quadrature, espresso con un integrale definito. Rom. Acc. L. Rend. **22**, 562–569 (1913)
24. Tang, T., Xie, H., Yin, X.: High-order convergence of spectral deferred correction methods on general quadrature nodes. J. Sci. Comput. **56**(1), 1–13 (2012)
25. Tang, T., Xu, X.: Accuracy enhancement using spectral postprocessing for differential equations and integral equations. Commun. Comput. Phys. **2–4**, 779–792 (2009)
26. Woollands, R., Bani Younes, A., Junkins, J.: New solutions for the perturbed Lambert problem using regularization and Picard iteration. J. Guid. Control. Dyn. **38**(9), 1548–1562 (2011)
27. Woollands, R., Junkins, J.L.: Nonlinear differential equation solvers via adaptive Picard Chebyshev iteration: applications in astrodynamics. J. Guid. Control. Dyn. **42**(5), 1007–1022 (2019)