Interlibrary Loan Services – FSU Community
James Elliott – Resource Sharing Manager
The Florida State University
R.M. Strozier Library
116 Honors Way
Tallahasee, Florida 32306-2047
Email: lib-borrowing@fsu.edu
Website: https://www.lib.fsu.edu/service/interlibrary-loan
Phone: 850.644.4466

Non-FSU Institutions:
Lib-Lending@fsu.edu
850.644.4171

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Mathematics and Computers in Simulation 82 (2012) 2167–2181

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

Original Articles

A parareal algorithm based on waveform relaxation☆

Jun Liu [a,b], Yao-Lin Jiang [a,*]

[a] *Department of Mathematical Sciences, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China*
[b] *College of Science, China University of Petroleum, Qingdao, Shandong 266580, China*

Received 16 January 2009; received in revised form 23 October 2011; accepted 3 May 2012
Available online 21 June 2012

Abstract

We report a new parallel iterative algorithm for time-dependent differential equations by combining the known waveform relaxation (WR) technique with the classical parareal algorithm. The parallelism can be simultaneously exploited in both sub-systems by WR and time by parareal. We also provide a sharp estimation on errors for the new algorithm. The iterations of parareal and WR are balanced to optimize the performance of the algorithm. Furthermore, the parallel speedup and efficiency of the new approach are analyzed by comparing with the classical parareal algorithm and the WR technique, respectively. Numerical experiments are carried out to verify the effectiveness of the theoretic work.

MSC: 45L05; 65L05; 65L20

Keywords: Parareal algorithm; Waveform relaxation; Parallelism in sub-systems and time; Convergence analysis

1. Introduction

The parareal algorithm is a time-parallel iterative process for solving differential equations, which was first proposed by Lions et al. [18]. It has been applied to many practical problems [3,5,6,20–22,27]. Several efficient variants in such parallel framework have been presented [9,12,13,23]. The parareal algorithm consists of some individual sub-problems with their own initial values which are updated during iterations. Fortunately, after few iterations its global accuracy of computations is often comparable to that given by a sequential numerical method used on a fine discretization in time.

Waveform relaxation (WR), also known as dynamic iteration, was first presented as a practical computational tool by Lelarasmee et al. to simulate MOS VLSI circuits [17]. It is also a highly parallel iterative technique for numerically solving large scale time-dependent systems, such as ordinary differential equations (ODEs) [4,8,10] and differential algebraic equations (DAEs) [14–16]. The WR technique consists of a "divide-and-conquer" approach, which means it decouples a large system into a number of simplified sub-systems on time-interval, and these sub-systems can be solved independently with their own time integrations.

☆ This work was supported by the Natural Science Foundation of China (NSFC) under grant 11071192, the International Science and Technology Cooperation Program of China under grant 2010DFA14700. This work was partly supported by a scholarship from the China Scholarship Council (J. Liu).

* Corresponding author. Tel.: +86 29 82665512.
E-mail address: yljiang@mail.xjtu.edu.cn (Y.-L. Jiang).

0378-4754/$36.00 © 2012 IMACS. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.matcom.2012.05.017

Recently, spectral deferred correction is incorporated into parareal framework [24,25] to reduce the computational cost of the parareal algorithm significantly. Such a strategy motivates us to combine the WR technique and the classical parareal algorithm together to construct a new parareal WR algorithm. That is, we use the WR iterations to replace the fine propagators of parareal. By adjusting the number of WR iterations in each parareal iteration, we can balance the two different kinds of iterations. The new algorithm here is very flexible since its parallelism can be exploited both in sub-systems and time. Moreover, such an approach makes the computational cost reduced greatly.

The remainder of the paper is organized as follows. In Section 2, we first introduce the WR technique into the parareal framework. An estimation on errors is then presented, which leads to a superlinear convergence for the new algorithm. In Section 3, the iterations of parareal and WR are balanced to obtain a better algorithm performance. Furthermore, we analyze the parallel speedup and efficiency of the hybrid algorithm in Section 4. Numerical experiments in Section 5 are provided to verify the effectiveness of the theoretic work. Finally, some conclusions are given.

2. Parareal by waveform relaxation

2.1. The review of the parareal approach

The parareal algorithm can be identified as a variant of some other methods, such as multiple shooting method and time-multigrid method [11,26]. To describe clearly, we consider the parareal process for a general nonlinear system of ODEs as follows:

$$\begin{cases} \dfrac{du(t)}{dt} = f(u(t), t), \\ u(0) = u_0, \quad t \in [0, T], \end{cases} \tag{1}$$

where the nonlinear function $f: \mathbf{R}^n \times \mathbf{R} \to \mathbf{R}^n$ is Lipschitz on $\mathbf{R}^n$, and the function $u: \mathbf{R} \to \mathbf{R}^n$ is to be computed.

We decompose the time interval $[0, T]$ into N time sub-intervals $[T_n, T_{n+1}]$, $n = 0, 1, \ldots, N-1$, with $0 = T_0 < T_1 < \cdots < T_{N-1} < T_N = T$. We suppose that all the time sub-intervals have the same length, namely $\Delta T = T_{n+1} - T_n = T/N$. Then, system (1) is converted to the following evolution systems:

$$\begin{cases} \dfrac{du_n(t)}{dt} = f(u_n(t), t), \quad t \in [T_n, T_{n+1}], \\ u_n(T_n) = U_n, \quad n = 0, 1, \ldots, N-1. \end{cases} \tag{2}$$

Here, the initial values U_n need to be determined such that the solutions on the time sub-intervals $[T_n, T_{n+1}]$ coincide with the restriction of the solution of system (1) to $[T_n, T_{n+1}]$. It means that the values U_n have to satisfy

$$U_0 = u_0, \quad U_{n+1} = S_{\Delta T}(U_n), \quad n = 0, 1, \ldots, N-1, \tag{3}$$

where $S_{\Delta T}(U_n)$ is the solution of system (1) at T_{n+1} with an initial value U_n at T_n. We denote $U = (U_0^{\mathrm{T}}, \ldots, U_N^{\mathrm{T}})^{\mathrm{T}}$, then system (3) can be rewritten in the form

$$H(U) = \begin{pmatrix} U_0 - u_0 \\ U_1 - S_{\Delta T}(U_0) \\ \vdots \\ U_N - S_{\Delta T}(U_{N-1}) \end{pmatrix} = 0. \tag{4}$$

Applying Newton's method to (4) yields

$$U_0^{k+1} = u_0, \quad U_{n+1}^{k+1} = S_{\Delta T}(U_n^k) + S'_{\Delta T}(U_n^k)(U_n^{k+1} - U_n^k), \tag{5}$$

where $n = 0, 1, \ldots, N-1$. However, it is very expensive to compute the Jacobian terms in (5). Two propagators are employed to deal with this problem.

Let $G(T_{n+1}, T_n, U_n)$ be a coarse approximation to $S_{\Delta T}(U_n)$, and let $F(T_{n+1}, T_n, U_n)$ be a more accurate one. We approximate the terms in system (5) with $S_{\Delta T}(U_n^k) \approx F(T_{n+1}, T_n, U_n^k)$ and $S'_{\Delta T}(U_n^k)(U_n^{k+1} - U_n^k) \approx G(T_{n+1}, T_n, U_n^{k+1}) - G(T_{n+1}, T_n, U_n^k)$, respectively. It leads to the classical parareal algorithm as follows:

$$\begin{cases} U_0^{k+1} = u_0, \\ U_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1}) + F(T_{n+1}, T_n, U_n^k) - G(T_{n+1}, T_n, U_n^k). \end{cases} \tag{6}$$

The initial iteration is often chosen as $U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0)$ for $n = 0, 1, \ldots, N-1$.

2.2. Waveform relaxation on time sub-interval

There are two classical convergence results for the WR technique [4]: (i) for nonlinear systems of ODEs on bounded time intervals, WR has superlinear convergence under a Lipschitz condition on the splitting function; (ii) for linear systems of ODEs on unbounded time intervals, WR has linear convergence under some dissipation assumption on the splitting function.

In this sub-section, we present a detailed error bound of WR for the following ODEs:

$$\begin{cases} \dfrac{du_n(t)}{dt} = f(u_n(t), t), \\ u_n(T_n) = U_n, \quad t \in [T_n, T_{n+1}]. \end{cases} \tag{7}$$

The WR technique of (7) can be written as

$$\begin{cases} \dfrac{du_n^{(l+1)}(t)}{dt} = \tilde{f}(u_n^{(l+1)}(t), u_n^{(l)}(t), t), \\ u_n^{(l+1)}(T_n) = U_n, \quad t \in [T_n, T_{n+1}], \quad l = 0, 1, \ldots, \end{cases} \tag{8}$$

where the initial guess $u_n^{(0)}(t)$ is chosen as $u_n^{(0)}(t) \equiv U_n$ on $[T_n, T_{n+1}]$. The nonlinear splitting function $\tilde{f}$ satisfies

$$\tilde{f}(u, u) = f(u), \quad u \in \mathbf{R}^n. \tag{9}$$

Moreover, we assume that there exist constants L_1 and L_2 such that

$$\|\tilde{f}(u_1, v_1) - \tilde{f}(u_2, v_2)\| \le L_1\|u_1 - u_2\| + L_2\|v_1 - v_2\|, \tag{10}$$

where $u_1, u_2, v_1, v_2 \in \mathbf{R}^n$. Here we give a detailed error bound of the WR approximation, which is useful for the forthcoming Theorem 2.1.

Lemma 2.1. *We suppose that the function f is Lipschitz on* $\mathbf{R}^n$*, with a Lipschitz constant L, and the splitting function* $\tilde{f}$ *satisfies (9) and (10), then the error of the WR technique (8) on the time sub-interval* $[T_n, T_{n+1}]$ *after l iterations is bounded by*

$$\|u_n(T_{n+1}) - u_n^{(l)}(T_{n+1})\| \le \frac{(L_2 \Delta T e^{L_1 \Delta T})^l}{l!} C \Delta T,$$

where C is a constant.

Proof. The solutions of systems (7) and (8) satisfy

$$u_n(t) = U_n + \int_{T_n}^t f(u_n(s), s)ds, \quad u_n^{(l+1)}(t) = U_n + \int_{T_n}^t \tilde{f}(u_n^{(l+1)}(s), u_n^{(l)}(s), s)ds,$$

respectively. Subtracting $u_n^{(l+1)}(t)$ from $u_n(t)$ and taking norms, we obtain

$$\begin{aligned} \|u_n(t) - u_n^{(l+1)}(t)\| &= \| \int_{T_n}^t [\tilde{f}(u_n(s), u_n(s), s) - \tilde{f}(u_n^{(l+1)}(s), u_n^{(l)}(s), s)]ds\| \\ &\le L_1 \int_{T_n}^t \|u_n(s) - u_n^{(l+1)}(s)\|ds + L_2 \int_{T_n}^t \|u_n(s) - u_n^{(l)}(s)\|ds. \end{aligned}$$

From the inequality, we know that

$$\int_{T_n}^{t} \|u_n(s) - u_n^{(l+1)}(s)\| ds \le L_2 \int_{T_n}^{t} \int_{T_n}^{s} e^{L_1(t-s)} \|u_n(\tau) - u_n^{(l)}(\tau)\| d\tau ds$$

and

$$\begin{aligned}
\|u_n(t) - u_n^{(l+1)}(t)\| &\le L_1 L_2 \int_{T_n}^{t} \int_{T_n}^{s} e^{L_1(t-s)} \|u_n(\tau) - u_n^{(l)}(\tau)\| d\tau ds + L_2 \int_{T_n}^{t} \|u_n(s) - u_n^{(l)}(s)\| ds \\
&\le L_2 \int_{T_n}^{t} e^{L_1(t-s)} \|u_n(s) - u_n^{(l)}(s)\| ds \\
&\le L_2 e^{L_1 \Delta T} \int_{T_n}^{t} \|u_n(s) - u_n^{(l)}(s)\| ds.
\end{aligned}$$

After the recurrence of operations, the quantity $\|u_n(t) - u_n^{(l)}(t)\|$ can be bounded by

$$\|u_n(t) - u_n^{(l)}(t)\| \le \frac{(L_2 e^{L_1 \Delta T})^l (t - T_n)^l}{l!} \|u_n(t) - u_n^{(0)}(t)\|.$$

Let $t = T_{n+1}$, it follows that

$$\begin{aligned}
\|u_n(T_{n+1}) - u_n^{(l)}(T_{n+1})\| &\le \frac{(L_2 \Delta T e^{L_1 \Delta T})^l}{l!} \|u_n(T_{n+1}) - u_n^{(0)}(T_{n+1})\| \\
&= \frac{(L_2 \Delta T e^{L_1 \Delta T})^l}{l!} \|u_n(T_{n+1}) - u_n(T_n)\| \\
&\le \frac{(L_2 \Delta T e^{L_1 \Delta T})^l}{l!} \Delta T \| f(u_n(\bar{s}))\|,
\end{aligned}$$

where $\bar{s} \in [T_n, T_{n+1}]$. Considering the continuity of the functions $f(u)$ and $u_n(t)$, we assume that $\|f(u(t))\|$ has an upper bound C. Thus, the desired result holds. □

Lemma 2.1 implies that, the sequence of $\{\|u_n(T_{n+1}) - u_n^{(l)}(T_{n+1})\|\}$ tends to 0 with superlinear rate of convergence as l goes to infinity. To end this sub-section, we denote

$$\epsilon(l) = \frac{(L_2 \Delta T e^{L_1 \Delta T})^l}{l!} C \Delta T.$$

We can see that, the smaller ΔT is, the smaller $\epsilon(l)$ is.

2.3. The construction of the parareal WR algorithm

The paper [24,25] used an iterative ODE solver to replace both the fine and coarse propagators. Motivated by the strategy, we combine the parareal algorithm with WR in this paper. With the initial value U_n at T_n, we denote the WR solution at T_{n+1} by $U_{n+1} = W_{s(k)}(T_{n+1}, T_n, U_n)$, where $s(k)$ is the iteration number of WR. Replacing the F propagator by the $W_{s(k)}$ propagator, we obtain the parareal WR algorithm as follows:

$$U_0^{k+1} = u_0, \quad U_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1}) + W_{s(k+1)}(T_{n+1}, T_n, U_n^k) - G(T_{n+1}, T_n, U_n^k). \tag{11}$$

It means that, we perform $s(k+1)$ iterations of WR on $[T_n, T_{n+1}]$ to generate $W_{s(k+1)}(t_{n+1}, t_n, U_n^k)$ for the $(k+1)$th parareal iteration. For the first several parareal iterations, $s(k)$ can be chosen small to avoid much computations. Given a small tolerance, there exists an integer k_0, such that the error by k_0 iterations of WR for system (7) on $[T_n, T_{n+1}]$ is less than the tolerance. We call the integer k_0 as the critical iteration number of WR on $[T_n, T_{n+1}]$. Therefore, it is reasonable to define $s(k) = \min\{m_0 k, k_0\}$, where m_0 is a parameter.

2.4. The estimation on the iteration error for the parareal WR algorithm

Bal [1], Gander and Hairer [7] gave the convergence analysis for the classical parareal algorithm. Based on their work, we present the convergence analysis for the parareal WR algorithm. Let $S(t)$ be the solution semigroup of system (1). Suppose that the difference between the approximate solution given by the G propagator and the exact solution can be expanded for small ΔT as

$$S(\Delta T)x - G(T_{n+1}, T_n, x) = c_{p+1}(x)\Delta T^{p+1} + c_{p+2}(x)\Delta T^{p+2} + \cdots, \tag{12}$$

and G satisfies the Lipschitz condition

$$\|G(t + \Delta T, t, x) - G(t + \Delta T, t, y)\| \le (1 + C_2\Delta T)\|x - y\|. \tag{13}$$

With these conditions, we have the following theorem.

Theorem 2.1. *Assume the conditions (12) and (13) hold, where the functions c_j, $j = p+1, p+2, \ldots$, are continuously differentiable, and C_2 is a positive constant, then the parareal WR algorithm (11) for system (1) is convergent.*

Proof. For simplicity, we denote the two propagations $G(T_{n+1}, T_n, U)$ and $F(T_{n+1}, T_n, U)$ as $G(U)$ and $F(U)$ in this proof, respectively. On the time sub-interval $[T_n, T_{n+1}]$, the difference between the exact solution and the parareal WR solution is given by

$$\begin{aligned} u(T_{n+1}) - U_{n+1}^{k+1} &= S(\Delta T)u(T_n) - G(U_n^{k+1}) - W_{s(k+1)}(U_n^k) + G(U_n^k) \\ &= S(\Delta T)u(T_n) - G(u(T_n)) - (W_{s(k+1)}(U_n^k) - G(U_n^k)) \\ &\quad + G(u(T_n)) - G(U_n^{k+1}) \\ &= S(\Delta T)u(T_n) - G(u(T_n)) + [S(\Delta T)U_n^k - W_{s(k+1)}(U_n^k)] \\ &\quad + [G(U_n^k) - S(\Delta T)U_n^k] + G(u(T_n)) - G(U_n^{k+1}). \end{aligned}$$

Using the condition (12), we have

$$\begin{aligned} u(T_{n+1}) - U_{n+1}^{k+1} &= [S(\Delta T)U_n^k - W_{s(k+1)}(U_n^k)] + G(u(T_n)) - G(U_n^{k+1}) + c_{p+1}(u(T_n))\Delta T^{p+1} \\ &\quad - c_{p+1}(U_n^k)\Delta T^{p+1} + c_{p+2}(u(T_n))\Delta T^{p+2} - c_{p+2}(U_n^k)\Delta T^{p+2} + \cdots. \end{aligned}$$

Since the functions c_j, $j = p+1, p+2, \ldots$, are continuously differentiable, there exists a constant C_1 such that

$$\begin{aligned} \|u(T_{n+1}) - U_{n+1}^{k+1}\| &\le C_1\Delta T^{p+1}\|u(T_n) - U_n^k\| + (1 + C_2\Delta T)\|u(T_n) - U_n^{k+1}\| \\ &\quad + \|S(\Delta T)U_n^k - W_{s(k+1)}(U_n^k)\|. \end{aligned}$$

From Lemma 2.1, we have $\|S(\Delta T)U_n^k - W_{s(k+1)}(U_n^k)\| < \epsilon(s(k+1))$. By the above inequality, we consider the recurrence relation

$$e_{n+1}^{k+1} = \alpha e_n^k + \beta e_n^{k+1} + \epsilon(s(k+1)), \quad e_{n+1}^0 = \gamma + \beta e_n^0, \tag{14}$$

where $\alpha = C_1\Delta T^{p+1}$, $\beta = 1 + C_2\Delta T$, and $\gamma = C_3\Delta T^{p+1}$. We choose $e_0^j = 0$, for $j = 0, 1, \ldots$. It is easy to check that, e_{n+1}^{k+1} is an upper bound on $\|u(T_{n+1}) - U_{n+1}^{k+1}\|$. Multiplying (14) by ζ^{n+1} and summing over n, where $|\zeta| < 1$, we obtain

$$\begin{aligned} \sum_{n\ge0} e_{n+1}^{k+1}\zeta^{n+1} &= \sum_{n\ge0}\alpha e_n^k\zeta^{n+1} + \sum_{n\ge0}\beta e_n^{k+1}\zeta^{n+1} + \sum_{n\ge0}\epsilon(s(k+1))\zeta^{n+1}, \\ \sum_{n\ge0} e_{n+1}^{0}\zeta^{n+1} &= \sum_{n\ge0}\gamma\zeta^{n+1} + \sum_{n\ge0}\beta e_n^0\zeta^{n+1}. \end{aligned}$$

We define $\rho_k(\zeta) \triangleq \sum_{n\ge0} e_{n+1}^k\zeta^{n+1}$. This function satisfies the recurrence relation

$$\rho_{k+1}(\zeta) = \alpha\zeta\rho_k(\zeta) + \beta\zeta\rho_{k+1}(\zeta) + \epsilon(s(k+1))\frac{\zeta}{1-\zeta}, \quad \rho_0(\zeta) = \gamma\frac{\zeta}{1-\zeta} + \beta\zeta\rho_0(\zeta).$$

Then, we have

$$\rho_k(\zeta) = \frac{\alpha\zeta}{1-\beta\zeta}\rho_{k-1}(\zeta) + \frac{\epsilon(s(k))\zeta}{(1-\zeta)(1-\beta\zeta)}, \quad \rho_0(\zeta) = \frac{\gamma\zeta}{(1-\zeta)(1-\beta\zeta)}.$$

By calculation, we get

$$\begin{aligned}\rho_k(\zeta) &= \left(\frac{\alpha\zeta}{1-\beta\zeta}\right)^k \rho_0(\zeta) + \frac{\zeta}{(1-\zeta)(1-\beta\zeta)}(\epsilon(s(k)) + a\epsilon(s(k-1)) + \cdots + a^{k-1}\epsilon(s(1))) \\ &= \frac{\gamma\alpha^k\zeta^{k+1}}{(1-\beta\zeta)^{k+1}(1-\zeta)} + \frac{\zeta}{(1-\zeta)(1-\beta\zeta)}(\epsilon(s(k)) + a\epsilon(s(k-1)) + \cdots + a^{k-1}\epsilon(s(1))),\end{aligned}$$

where $a = \alpha\zeta/(1-\beta\zeta)$.

Since replacing the factor $1-\zeta$ by $1-\beta\zeta$ in the above denominator increases only the coefficients in the power series of $\rho_k(\zeta)$. It means that the coefficients in the power series of $\rho_k(\zeta)$ are less than the counterparts in the power series of $\widetilde{\rho}_k(\zeta)$, where

$$\widetilde{\rho}_k(\zeta) = \frac{\gamma\alpha^k\zeta^{k+1}}{(1-\beta\zeta)^{k+2}} + \frac{\zeta}{(1-\beta\zeta)^2}(\epsilon(s(k)) + a\epsilon(s(k-1)) + \cdots + a^{k-1}\epsilon(s(1))).$$

Using the binomial series expansion

$$\frac{1}{(1-\beta\zeta)^{k+2}} = \sum_{j\geq 0}\binom{k+1+j}{j}\beta^j\zeta^j,$$

the coefficient of the term ζ^n in $\widetilde{\rho}_k(\zeta)$ is

$$\begin{aligned}v_n &\triangleq \gamma\alpha^k\binom{n}{k+1}\beta^{n-k-1} + \sum_{i=0}^{k-1}\binom{n}{i+1}\alpha^i\beta^{n-i-1}\epsilon(s(k-i)) \\ &= C_3\Delta T^{p+1}C_1^k(\Delta T^{p+1})^k\frac{n!}{(n-k-1)!(k+1)!}(1+C_2\Delta T)^{n-k-1} \\ &\quad + \sum_{i=0}^{k-1}\frac{n!}{(n-i-1)!(i+1)!}C_1^i(\Delta T^{p+1})^i(1+C_2\Delta T)^{n-i-1}\epsilon(s(k-i)).\end{aligned}$$

Due to $1+x \leq e^x$ for $x>0$, it follows

$$\begin{aligned}v_n &\leq \frac{C_3}{C_1}\frac{(C_1\Delta T^{p+1})^{k+1}}{(k+1)!}(1+C_2\Delta T)^{n-k-1}\prod_{j=0}^{k}(n-j) \\ &\quad + \sum_{i=0}^{k-1}\frac{C_1^i(\Delta T^{p+1})^i}{(i+1)!}(1+C_2\Delta T)^{n-i-1}\epsilon(s(k-i))\prod_{j=0}^{i}(n-j) \\ &\leq \frac{C_3}{C_1}\frac{(C_1T_n)^{k+1}}{(k+1)!}e^{C_2(T_n-T_{k+1})}\Delta T^{p(k+1)} \\ &\quad + \sum_{i=0}^{k-1}\frac{C_1^i(\Delta T^p)^i(T_n)^i}{i!}\frac{T_n}{\Delta T}e^{C_2(T_n-T_{i+1})}\epsilon(s(k-i)).\end{aligned}$$

Now, we discuss the upper bound on the iteration error in two situations. First, if $k < \lceil \frac{k_0}{m_0} \rceil$, where $\lceil a \rceil$ means the smallest integer which is not smaller than a, then $m_0 k < k_0$. For any $i = 0, \ldots, k-1$, $\epsilon(s(k-i)) = \frac{(L_2 \Delta T e^{L_1 \Delta T})^{m_0(k-i)}}{(m_0(k-i))!} C \Delta T$, and the quantity v_n is bounded by

$$\begin{aligned} v_n \quad \leq \quad & \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} \Delta T^{p(k+1)} \\ & + \sum_{i=0}^{k-1} \frac{C_1^i (\Delta T^p)^i (T_n)^i}{i!} T_n e^{C_2(T_n - T_{i+1})} \frac{(L_2 \Delta T e^{L_1 \Delta T})^{m_0(k-i)}}{(m_0(k-i))!} C. \end{aligned} \tag{15}$$

If $k \geq \lceil \frac{k_0}{m_0} \rceil$, we notice that $m_0(k-i) \geq k_0$ and $s(k-i) = k_0$ for $i \leq k - \lceil \frac{k_0}{m_0} \rceil$. Therefore, it has

$$\begin{aligned} v_n \quad \leq \quad & \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} \Delta T^{p(k+1)} \\ & + \sum_{i=0}^{k - \lceil \frac{k_0}{m_0} \rceil} \frac{C_1^i (\Delta T^p)^i (T_n)^i}{i!} \frac{T_n}{\Delta T} e^{C_2(T_n - T_{i+1})} \frac{(L_2 \Delta T e^{L_1 \Delta T})^{k_0}}{k_0!} C \Delta T \\ & + \sum_{i=k - \lceil \frac{k_0}{m_0} \rceil + 1}^{k-1} \frac{C_1^i (\Delta T^p)^i (T_n)^i}{i!} T_n e^{C_2(T_n - T_{i+1})} \frac{(L_2 \Delta T e^{L_1 \Delta T})^{m_0(k-i)}}{(m_0(k-i))!} C. \end{aligned} \tag{16}$$

We can see that the first term in the right side of (16) converges to 0 superlinearly, as k goes to infinity. By the definition of k_0, the term $\frac{(L_2 \Delta T e^{L_1 \Delta T})^{k_0}}{k_0!}$ is always smaller than a numerical tolerance ϵ_0, and the second term in the right side of (16) has an upper bound $C \epsilon_0 T_n e^{(C_2 + C_1 \Delta T^p) T_n}$. The third term in the right side of (16) includes a couple of components, and each of them converges to 0 superlinearly. Thus, it completes the proof of the theorem. □

By checking (15) and (16) we can see that, each component in the upper bounds converges to 0 superlinearly, though more components appear as k increases. Therefore, we can regard the iteration error of the parareal WR algorithm converges superlinearly to the tolerance in a rough meaning.

3. Balance of iterations by parareal and WR

The parareal WR algorithm includes two kinds of iterations, i.e., parareal iterations and WR iterations, and they usually converge at different rates. In this section, we will show how to find a proper integer m_0 to balance the two different kinds of iterations by the following linear ODEs:

$$\begin{cases} \dfrac{du(t)}{dt} = Mu(t), \quad t \in [0, T], \\ u(0) = U_0 \in \mathbf{R}^J. \end{cases} \tag{17}$$

First, the scheme of the parareal WR algorithm for system (17) is

$$\begin{cases} U_{n+1}^0 = G(T_{n+1}, T_n, U_n^0), \\ U_{n+1}^{k+1} = G(T_{n+1}, T_n, U_n^{k+1}) + W_{s(k+1)}(T_{n+1}, T_n, U_n^k) - G(T_{n+1}, T_n, U_n^k), \end{cases}$$

where $W_{s(k)}$ propagator is produced by

$$\begin{cases} \dfrac{d}{dt} u^{(l+1)}(t) = M_1 u^{(l+1)}(t) + M_2 u^{(l)}(t), \quad t \in [T_n, T_{n+1}], \\ u^{(l+1)}(T_n) = U_n^k, \quad l = 0, 1, \ldots, s(k) - 1 \end{cases} \tag{18}$$

with the splitting $M = M_1 + M_2$. Here, $s(k) = \min\{m_0 k, k_0\}$ and k_0 is the critical iteration number of WR.

We employ the recurrence relation (14) for system (17), and rewrite it in a vector form, i.e.,

$$\begin{pmatrix} e_1^{k+1} \\ e_2^{k+1} \\ \vdots \\ e_N^{k+1} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ -\beta & 1 & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & -\beta & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} 0 & & & \\ \alpha & 0 & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & \alpha & 0 \end{pmatrix} \begin{pmatrix} e_1^k \\ e_2^k \\ \vdots \\ e_N^k \end{pmatrix} + \begin{pmatrix} \epsilon_1(s(k+1)) \\ \epsilon_2(s(k+1)) \\ \vdots \\ \epsilon_N(s(k+1)) \end{pmatrix} \right]$$

$$= \begin{pmatrix} 1 & & & \\ -\beta & 1 & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & -\beta & 1 \end{pmatrix}^{-1} \left[\begin{pmatrix} 0 & & & \\ \alpha & 0 & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & \alpha & 0 \end{pmatrix} \begin{pmatrix} e_1^k \\ e_2^k \\ \vdots \\ e_N^k \end{pmatrix} + \begin{pmatrix} \rho\epsilon_1(s(k)) \\ \rho\epsilon_2(s(k)) \\ \vdots \\ \rho\epsilon_N(s(k)) \end{pmatrix} \right],$$

where $\epsilon_n(s(k))$ is the error at $n\Delta T$ caused by $s(k)$ iterations of WR, and ρ is the convergence factor of WR between two adjacent parareal iterations on each time sub-interval. It is expected that the factor ρ equals to the factor α which describes the truncation error of the coarse propagator.

Now, we investigate the convergence factor of the discrete WR process. The discretization of system (18) is

$$\frac{U_{n,m+1}^{(l+1)} - U_{n,m}^{(l+1)}}{\delta t} = M_1 U_{n,m+1}^{(l+1)} + M_2 U_{n,m+1}^{(l)},$$

where $m = 0, 1, \ldots, \overline{q} - 1$, and $\overline{q}\delta t = \Delta T$. The discretization of system (17) is

$$\frac{U_{n,m+1} - U_{n,m}}{\delta t} = M_1 U_{n,m+1} + M_2 U_{n,m+1}.$$

We define $\epsilon_{n,m}^{(l)} = U_{n,m}^{(l)} - U_{n,m}$, it then has

$$\epsilon_{n,m+1}^{(l+1)} = (I - M_1\delta t)^{-1}\epsilon_{n,m}^{(l+1)} + (I - M_1\delta t)^{-1} M_2 \delta t \epsilon_{n,m+1}^{(l)}.$$

Let $\vec{\epsilon}_n^{(l)} = [(\epsilon_{n,1}^{(l)})^{\mathrm{T}}, \ldots, (\epsilon_{n,\overline{q}}^{(l)})^{\mathrm{T}}]^{\mathrm{T}}$, and $A = (I - M_1\delta t)^{-1}$, it follows

$$\vec{\epsilon}_n^{(l+1)} = \begin{pmatrix} 0 & & & \\ A & 0 & & \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & A & 0 \end{pmatrix} \vec{\epsilon}_n^{(l+1)} + \begin{pmatrix} AM_2\delta t & & & \\ & AM_2\delta t & & \\ & & \ddots & \\ & & & AM_2\delta t \end{pmatrix} \vec{\epsilon}_n^{(l)},$$

and then

$$\vec{\epsilon}_n^{(l+1)} = \delta t \begin{pmatrix} AM_2 & & & \\ A^2M_2 & AM_2 & & \\ \vdots & \ddots & \ddots & \\ A^{\overline{q}}M_2 & \cdots & A^2M_2 & AM_2 \end{pmatrix} \vec{\epsilon}_n^{(l)}.$$

We denote by $\overline{a}$ the infinity norm for the coefficient matrix in the above expression. The convergence factor ρ of the discrete WR process equals $\overline{a}^{m_0}$.

Next, we consider the factor α. The coarse propagator for system (17) can be derived by the scheme

$$\frac{U_{n,m+1} - U_{n,m}}{\Delta t} = MU_{n,m+1},$$

where $m = 0, 1, \ldots, q-1$, and $q\Delta t = \Delta T$. Then we have the expression

$$U_{n,m+1} = (I - \Delta tM)^{-1} U_{n,m},$$

and we set

$$G(T_{n+1}, T_n, U_n^k) = (I - \Delta tM)^{-q} U_n^k.$$

Since a fine approximation to the solution of system (17) is

$$S_{\delta t}(T_{n+1}, T_n, U_n^k) = (I - \delta tM)^{-\overline{q}} U_n^k, \quad \overline{q}\delta t = \Delta T,$$

therefore, the factor α can be estimated roughly by

$$\alpha = \|(I - \Delta tM)^{-q} - (I - \delta tM)^{-\overline{q}}\|_\infty.$$

With the two convergence factors ρ and α, we obtain the balance equation

$$\alpha = \overline{a}^{m_0},$$

which leads to

$$m_0 = \log_{\overline{a}} \|(I - \Delta tM)^{-q} - (I - \delta tM)^{-\overline{q}}\|_\infty.$$

4. Parallel speedup and efficiency

Similar to the analysis for the parallel speedup and efficiency of the classical parareal algorithm presented in [2,24], we ignore the communication overhead, and adopt the following notations:

- T is the length of the time interval.
- ΔT is the length of each time sub-interval.
- Δt is the time increment for the coarse propagator.
- δt is the time increment for the discrete WR propagator.
- N is the number of time sub-intervals on $[0, T]$, i.e., $T = N\Delta T$.
- k_0 is the critical iteration number of WR on each time sub-interval.
- k_1 is the iteration number of the classical parareal algorithm.
- k_2 is the iteration number of the parareal WR algorithm.
- S is the parallel speedup. Subscripts are used for different algorithms.
- E is the parallel efficiency with $E = S/N$. Subscripts are used for different algorithms.

We assume that the classical parareal algorithm needs k_1 iterations, and the parareal WR algorithm needs k_2 iterations, to achieve a desired accuracy. In addition, the window WR algorithm takes k_0 iterations on each time sub-interval to the same accuracy. We further assume that the time complexity for solving an n-dimensional linear system is $\mathcal{O}(n^{\tilde{p}})$ with $\tilde{p} \geq 1$, and denote by τ the time cost for solving such a linear system.

According to the analysis in [24], the parallel speedup of the classical parareal algorithm in the pipelined fashion is

$$S_{\text{cla}} = \frac{N\frac{\Delta T}{\delta t}\tau}{N\frac{\Delta T}{\Delta t}\tau + k_1\left(\frac{\Delta T}{\delta t} + \frac{\Delta T}{\Delta t}\right)\tau} = \frac{N\frac{\Delta T}{\delta t}}{(k_1 + N)\frac{\Delta T}{\Delta t} + k_1\frac{\Delta T}{\delta t}} = \frac{N}{(k_1 + N)\frac{\delta t}{\Delta t} + k_1},$$

and the parallel efficiency is

$$E_{\mathrm{cla}} = \frac{S_{\mathrm{cla}}}{N} = \frac{1}{(k_1+N)\dfrac{\delta t}{\Delta t}+k_1}.$$

To present the parallel efficiency for the parareal WR algorithm clearly, we consider here a special situation, i.e., each WR propagation is computed by one processor. If the n-dimensional linear system is decoupled into r sub-systems by WR, the time complexity for each WR iteration averaged at each time step will be $r\mathcal{O}\left(\frac{n}{r}\right)^{\tilde{p}}$, and the corresponding time cost at each time step will be $s(k)\frac{\tau}{r^{\tilde{p}-1}}$. Therefore, the parallel speedup of the parareal WR algorithm is

$$S_{\mathrm{PWR}} = \frac{N\dfrac{\Delta T}{\delta t}\tau}{N\dfrac{\Delta T}{\Delta t}\tau+\sum_{i=1}^{k_2}\left(s(i)\dfrac{\Delta T}{\delta t}\dfrac{1}{r^{\tilde{p}-1}}\tau+\dfrac{\Delta T}{\Delta t}\tau\right)} = \frac{N}{(N+k_2)\dfrac{\delta t}{\Delta t}+\sum_{i=1}^{k_2}s(i)\dfrac{1}{r^{\tilde{p}-1}}},$$

and the parallel efficiency is

$$E_{\mathrm{PWR}} = \frac{1}{(N+k_2)\dfrac{\delta t}{\Delta t}+\sum_{i=1}^{k_2}s(i)\dfrac{1}{r^{\tilde{p}-1}}}.$$

Especially, if system (1) is from the method of lines for some linear PDEs, the linear systems involved in all propagations can be solved by some efficient methods, such as multigrid technique with $\mathcal{O}(n)$ work. Then the corresponding parallel efficiency will be

$$\overline{E}_{PWR} = \frac{1}{(N+k_2)\dfrac{\delta t}{\Delta t}+\sum_{i=1}^{k_2}s(i)}.$$

Now, we compare the parallel speedup and efficiency of the parareal WR algorithm to these of the WR technique. On the time sub-interval $[T_n, T_{n+1}]$, we denote by $U_{n,m}^{(l)}$ the approximation at $(T_n + m\delta t)$ on the lth discrete waveform, where $l = 1, 2, \ldots, k_0$, and $m = 1, 2, \ldots, \frac{\Delta T}{\delta t}$. We assign the first processor to compute the first discrete waveform, and $U_{n,m}^{(1)}$ is transferred to the second processor as soon as $U_{n,m}^{(1)}$ is obtained. The second processor has to wait a work unit (WU) for a time step and starts its job after receiving $U_{n,1}^{(1)}$ from the first processor. Here, $U_{n,m}^{(2)}$ is transferred to the third processor as soon as it is obtained by the second processor. Similarly, the third processor has to wait two WUs before $U_{n,1}^{(2)}$ is transferred from the second processor. Likewise, the k_0th discrete waveform is computed by the k_0th processor, and the job starts $(k_0 - 1)$ WUs later after the first processor begins its job. The details of the parallelism of WR technique can be found in [19]. Therefore, the cost of WR on each time sub-interval is $\left(\frac{\Delta T}{\delta t}+k_0-1\right)\frac{1}{r^{\tilde{p}-1}}\tau$, and the speedup with k_0 processors is

$$S_{\mathrm{WR}} = \frac{N\dfrac{\Delta T}{\delta t}\tau}{N\left(\dfrac{\Delta T}{\delta t}+k_0-1\right)\dfrac{1}{r^{\tilde{p}-1}}\tau} = \frac{r^{\tilde{p}-1}}{1+\dfrac{(k_0-1)\delta t}{\Delta T}},$$

and the parallel efficiency is

$$E_{\mathrm{WR}} = \frac{r^{\tilde{p}-1}}{k_0\left(1+\dfrac{(k_0-1)\delta t}{\Delta T}\right)}.$$

In particular, if $\tilde{p} = 1$, the parallel efficiency of the WR technique will be

$$\overline{E}_{\mathrm{WR}} = \frac{1}{k_0\left(1+\dfrac{(k_0-1)\delta t}{\Delta T}\right)}.$$

Since $\delta t \ll \Delta T$, it is likely that $\overline{E}_{\mathrm{PWR}} < \overline{E}_{\mathrm{WR}}$. However, the parareal WR algorithm explores the parallelism of the WR technique.

5. Numerical experiments

Example 1. We consider the Lorenz equation from [7,25] as follows:

$$\begin{cases} \dot{x} = -10x + 10y, \\ \dot{y} = -28x - y - xz, \\ \dot{z} = xy - \dfrac{8}{3}z, \end{cases}$$

on the time interval [0, 10] with the initial condition $(x, y, z)(0) = (5, -5, 20)$. For the classical parareal algorithm, the interval [0, 10] is decomposed into 180 sub-intervals, the G and F propagators are chosen as one step and 80 steps of the fourth-order Runge–Kutta method on each sub-interval, respectively. For the parareal WR algorithm, we employ the same G propagator, and take the $W_{s(k)}$ propagator by the Jacobi WR process as

$$\begin{cases} \dot{x}^{(l+1)} = -10x^{(l+1)} + 10y^{(l)}, \\ \dot{y}^{(l+1)} = -28x^{(l)} - y^{(l+1)} - x^{(l)}z^{(l)}, \\ \dot{z}^{(l+1)} = x^{(l)}y^{(l)} - \dfrac{8}{3}z^{(l+1)}, \quad l = 0, 1, \ldots, s(k) \end{cases} \tag{19}$$

with the initial guess $(x^{(0)}, y^{(0)}, z^{(0)})(t) \equiv (x_n^k, y_n^k, z_n^k)$ for $t \in [T_n, T_{n+1}]$. Take the first equation of system (19) as an example, its numerical solution on $[T_n, T_{n+1}]$ can be obtained by

$$x_{n,m+1}^{(l+1)} = e^{-10(t_{n,m+1}-t_{n,m})}x_{n,m}^{(l+1)} + \int_{t_{n,m}}^{t_{n,m+1}} 10y^{(l)}(s)e^{-10(t_{n+1}-s)}ds. \tag{20}$$

The integral term of (20) is computed by the Simpson's rule, which not only needs less computational cost than the fourth-order Runge–Kutta method, but also preserves the local error $\mathcal{O}(h^5)$ and the global error $\mathcal{O}(h^4)$. In order to be clearer, we also call this kind of parareal WR algorithm as parareal WR (Jacobi) algorithm, and choose $k_0^{\mathrm{Jac}} = 12$ as the critical iteration number. In order to accelerate the convergence of scheme (19), we take the window Jacobi WR technique. In detail, each time sub-interval is divided into 4 windows and Jacobi WR scheme is implemented on the 4 windows serially. The hybrid algorithm by parareal and window Jacobi WR is named as parareal window WR (Jacobi) algorithm, where the critical iteration number can be chosen as $k_{4,0}^{\mathrm{Jac}} = 9$, and the subscript "4" implies the number of windows. The comparisons between the behaviors of the classical parareal algorithm, the parareal WR (Jacobi) algorithm and the parareal window WR (Jacobi) algorithm are shown in Fig. 1.

We can see from Fig. 1 that, the parareal WR (Jacobi) algorithm with $m_0 = 12$ costs almost the same running time as the classical parareal algorithm. Therefore, it is reasonable to assume that the classical F propagator and the $W_{s(k)}$ propagator by scheme (19) have the same time cost, which is measured in terms of the number of function evaluations as [24] addressed. Furthermore, we simply define the time cost as 1 for computing (20). Then the time cost of the $W_{s(k)}$ propagator at each time step is nk_0^{Jac}, where $n = 3$ is the number of sub-systems. The time cost for the classical F propagator at each time step is also set to be nk_0^{Jac}.

Assuming that $3N_0$ processors are available, we first solve the example by the classical parareal algorithm. We denote by τ_G and τ_F the cost of the coarse and the fine propagations of the classical parareal algorithm, respectively. The total cost of k parareal iterations is $(k + 3N_0)\tau_G + k\tau_F$, where $N_0 = 180$, $\tau_G = nk_0^{\mathrm{Jac}} = 36$ and $\tau_F = 36 \times 27$ since F is taken as $27(\approx 80/3)$ steps of the fourth-order Runge–Kutta method on each time sub-interval. For the parareal WR (Jacobi) algorithm with $m_0 = 12$, we decompose the time interval into 180 sub-intervals and use 3 processors to compute the WR propagations. The total cost of k iterations of parareal WR (Jacobi) is $(k + N_0)\tau_G + k\tau_W$, where τ_W is the cost for computing the WR propagation and $\tau_W = k_0^{\mathrm{Jac}} \times 80$. Therefore, the cost of the classical parareal algorithm is 1008k + 19,440, and the cost of the parareal WR (Jacobi) algorithm is 996k + 6480. The comparisons between the behaviors of the two algorithms are shown in Fig. 2.

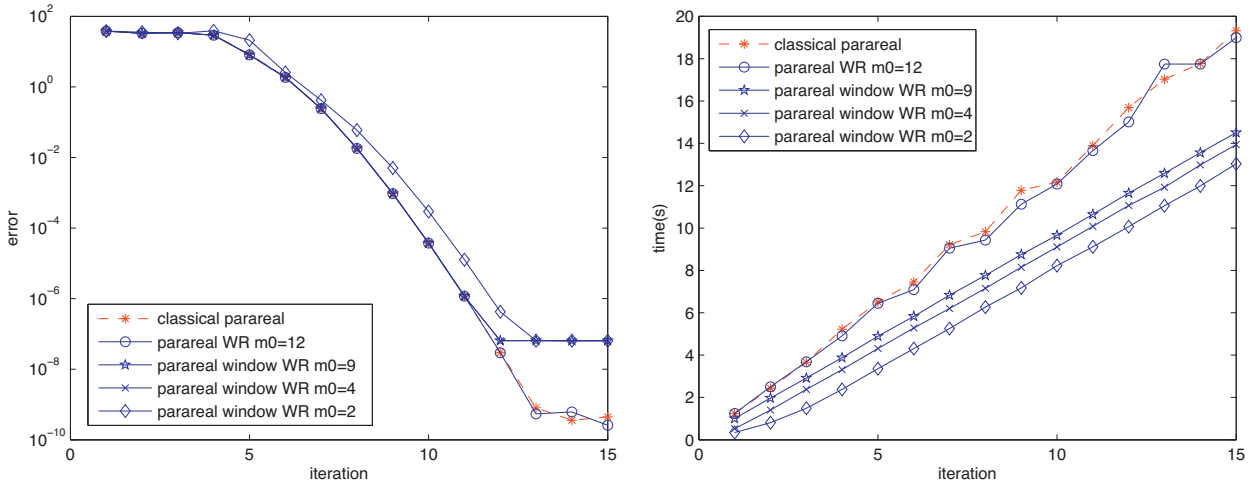

Fig. 1. Error and time versus iteration by the classical parareal algorithm and the parareal WR (Jacobi) algorithm with $k_0^{\mathrm{Jac}} = 12$, as well as the parareal window WR (Jacobi) algorithm with $k_{4,0}^{\mathrm{Jac}} = 9$.

In addition, the Gauss–Seidel WR algorithm is also very popular. The scheme of the Gauss–Seidel WR algorithm for the Lorenz equation is

$$\begin{cases} \dot{x}^{(l+1)} = -10x^{(l+1)} + 10y^{(l)}, \\ \dot{y}^{(l+1)} = -28x^{(l+1)} - y^{(l+1)} - x^{(l+1)}z^{(l)}, \\ \dot{z}^{(l+1)} = x^{(l+1)}y^{(l+1)} - \dfrac{8}{3}z^{(l+1)}, \quad l = 0, 1, \ldots, s(k) \end{cases} \tag{21}$$

with the initial guess $(x^{(0)}, y^{(0)}, z^{(0)})(t) \equiv (x_n^k, y_n^k, z_n^k)$ for $t \in [T_n, T_{n+1}]$. If the $W_{s(k)}$ propagator is produced by scheme (21), then the hybrid algorithm by G and $W_{s(k)}$ is called the parareal WR (Gauss–Seidel) algorithm, for which we choose $k_0^{GS} = 9$ as the critical iteration number. We assume that each Gauss–Seidel WR propagation is computed by one processor, and the decoupled system (21) is dealt with in the same way as (20). The cost for k iterations of the classical parareal algorithm by 180 processors can be estimated as $(k+180) \times 36 + 80 \times 36 \times k$. The cost of k iterations of the parareal WR (Gauss–Seidel) algorithm is $(k + 180) \times 36 + 80 \times 3 \times \sum_{i=1}^{k} \min\{m_0 i, k_0^{GS}\}$. Moreover, if we divide

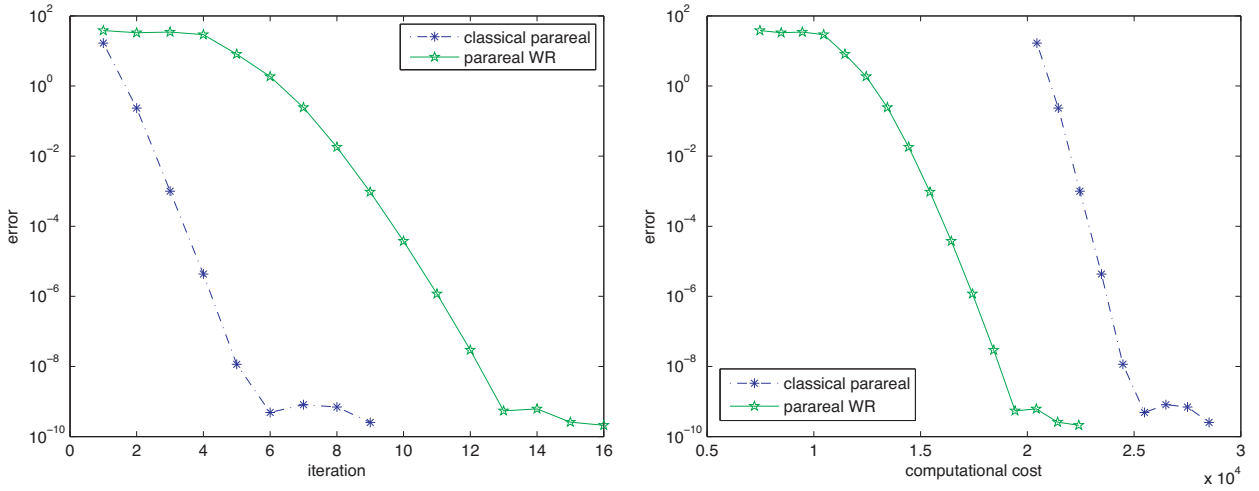

Fig. 2. Comparisons of the classical parareal algorithm and the parareal WR (Jacobi) algorithm with $k_0^{\mathrm{Jac}} = 12$, where the left plot shows the relationship between the error and the iteration and the right plot shows the relationship between the error and the computational cost.

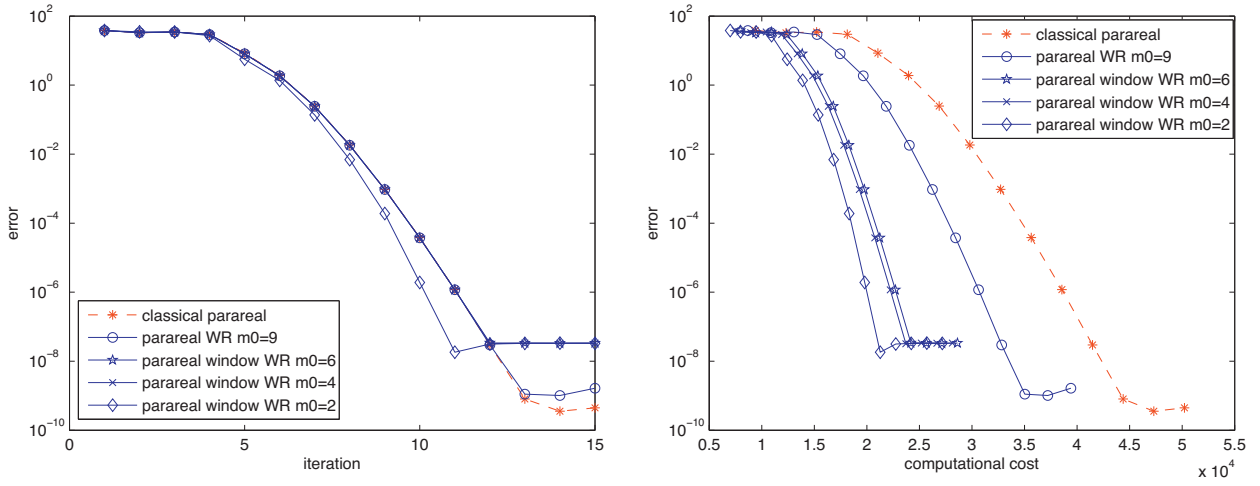

Fig. 3. Error versus iteration and computational cost by the classical parareal algorithm and the parareal WR (Gauss–Seidel) algorithm with $k_0^{GS} = 9$, as well as the parareal window WR (Gauss–Seidel) algorithm with $k_{4,0}^{GS} = 6$.

each time sub-interval into 4 windows and implement the Gauss–Seidel WR algorithm (21) on the 4 windows serially, we can obtain the window Gauss–Seidel WR propagation. The hybrid algorithm by parareal and window Gauss–Seidel WR is named as the parareal window WR (Gauss–Seidel) algorithm, with the critical iteration number $k_{4,0}^{GS} = 6$. The cost of k iterations of the parareal window WR (Gauss–Seidel) algorithm is $(k + 180) \times 36 + 80 \times 3 \times \sum_{i=1}^{k} \min\{m_0 i, k_{4,0}^{GS}\}$. The comparisons between the behaviors of the classical parareal algorithm, the parareal WR (Gauss–Seidel) algorithm and the parareal window WR (Gauss–Seidel) algorithm are shown in Fig. 3.

Example 2. We solve the following system of nonlinear ODEs discretized from the combustion theory,

$$\frac{du(t)}{dt} = \frac{1}{\Delta x^2} A u(t) - f(u(t)), \quad t \in [0, 1], \tag{22}$$

where, $u(0)$ is the reshaped vector from the approximation of the initial function $\sin \frac{\pi(x+1)}{2} \times \sin \frac{\pi(y+1)}{2}$ on a 50×50 grid of the square area $[-1, 1] \times [-1, 1]$ with grid size $\Delta x = \Delta y = 2/50$, $f(u) = \exp(\frac{u}{10+10u})$,

$$A = \begin{pmatrix} B & I & \cdots & 0 \\ I & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & I \\ 0 & \cdots & I & B \end{pmatrix}_{49^2 \times 49^2}, \quad B = \begin{pmatrix} -4 & 1 & \cdots & 0 \\ 1 & -4 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 1 & -4 \end{pmatrix}_{49 \times 49},$$

and $I \in \mathbf{R}^{49 \times 49}$ is an identity matrix.

We decompose the time interval $[0, 1]$ into 40 sub-intervals, then take one step and 40 steps of the following scheme to be the coarse and the fine propagators of the classical parareal algorithm,

$$U_{m+1} - U_m = \frac{\Delta t}{\Delta x^2} A U_{m+1} + \Delta t f(U_m).$$

For the $W_{s(k)}$ propagator, we divide each time sub-interval into 40 windows, and carry out one step of the discrete Gauss–Seidel scheme on each window, i.e., implement the scheme

$$U_{i,m+1}^{(l+1)} - U_{i,m}^{(l+1)} = \frac{\Delta t}{\Delta x^2} B U_{i,m+1}^{(l+1)} + \frac{\Delta t}{\Delta x^2} U_{i-1,m+1}^{(l+1)} + \frac{\Delta t}{\Delta x^2} U_{i+1,m+1}^{(l)} + \Delta t f(U_{i,m}^{(l+1)}),$$

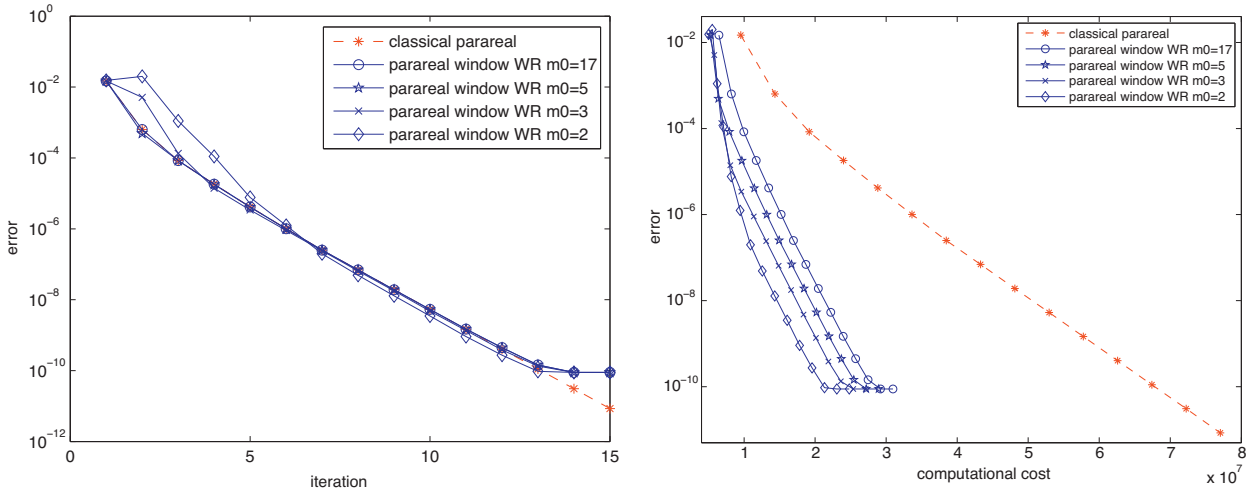

Fig. 4. Comparisons of the classical parareal algorithm and the parareal window WR (Gauss–Seidel) algorithm with $k_{40,0}^{GS} = 17$, where the left plot shows the relationships between the error and the iteration and the right plot shows the relationships between the error and the computational cost.

with $i = 1, 2, \ldots, 49$, $l = 0, 1, \ldots, s(k)$, for $m = 0, 1, \ldots, 39$. Here, $U_{i,m}^{(l)} \in \mathbf{R}^{49}$ is the approximation at $T_n + m\delta t$ of the ith sub-system on the lth waveform. We call the hybrid algorithm by the coarse and $W_{s(k)}$ propagators as the parareal window WR (Gauss–Seidel) algorithm, and set $k_{40,0}^{GS} = 17$ as the critical iteration number.

In the classical parareal algorithm, a linear banded system with band width $\sqrt{n}$ is solved at each time step. The time complexity for solving such a linear system is $\mathcal{O}(n^{3/2})$, where $n = 49 \times 49$ is the dimension of the linear system. In the WR process, all the linear systems are tridiagonal with time complexity $\mathcal{O}(n)$, and the number of such linear systems is $49 \times s(k)$. We suppose the time cost of a 49-dimensional tridiagonal linear system to be 49. The cost of $W_{s(k)}$ is $49 \times s(k) \times 40 \times 49$, and the cost of F is $40 \times (49 \times 49)^{3/2}$. Then the cost of the parareal window WR (Gauss–Seidel) algorithm is $(k + 40) \times (49 \times 49)^{3/2} + 40 \times 49 \times 49 \times \sum_{i=1}^{k} s(i)$, and that of the classical parareal algorithm is $(k + 40) \times (49 \times 49)^{\frac{3}{2}} + k \times 40 \times (49 \times 49)^{3/2}$. The comparisons of the behaviors of the classical parareal algorithm and the parareal window WR (Gauss–Seidel) algorithm for system (22) are shown in Fig. 4. We can see that the parareal window WR (Gauss–Seidel) algorithm takes much less computational cost than the classical parareal algorithm to achieve a desired accuracy.

6. Conclusions and future work

The known parareal and WR techniques have been combined to develop a new parallel algorithm, which could be carried out in parallel in sub-systems and time. A sharp upper bound on errors was presented, which indicates the superlinear convergence of the algorithm. In order to optimize the performance of the new algorithm, we showed a simple way to balance the parareal iterations and the WR iterations. In addition, we observed by numerical experiments that much less computational cost was needed than the classical parareal algorithm to achieve the same accuracy. However, it would worth for us to further consider the deferred correction strategy in future to promote the efficiency of the combination of the two different parallel approaches.

Acknowledgements

The authors would like to thank Professor Martin J. Gander and Professor Kenneth R. Jackson for their valuable suggestions on the paper. We are also grateful to the anonymous referees for their helpful comments to improve the paper significantly.

References

[1] G. Bal, On the convergence and the stability of the parareal algorithm to solve partial differential equations, in: 15th International Conference on Domain Decomposition Methods on Science and Engineering, July 21–25, Berlin, 2003.
[2] G. Bal, Parallelization in time of (stochastic) ordinary differential equations, available: www.columbia.edu/gb2030/PAPERS/paralleltime.pdf.
[3] G. Bal, Y. Maday, A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put, in: Workshop on Domain Decomposition, June 7–8, Zurich, 2001.
[4] K. Burrage, Parallel and Sequential Methods for Ordinary Differential Equations, Clarendon Press, Oxford, 1995.
[5] P.F. Fischer, F. Hecht, Y. Maday, A parareal in time semi-implicit approximation of the Navier-Stokes equations, in: 15th International Conference on Domain Decomposition Methods in Science and Engineering, July 21-25, Berlin, 2003.
[6] M.J. Gander, Analysis of the parareal algorithm to hyperbolic problems using characteristics, Boletin de la Sociedad Espanola de Matemática Aplicada (2008) 5–19.
[7] M.J. Gander, E. Hairer, Nonlinear convergence analysis for the parareal algorithm, in: 17th International Conference on Domain Decomposition Methods on Science and Engineering, July 3–7, St. Wolfgang/Strobl, 2006.
[8] M.J. Gander, A.K. Mohammad, A.E. Ruehli, Waveform relaxation technique for longitudinal partitioning of transmission lines, in: IEEE 15th Topical Meeting on Electrical Performance of Electronic Packaging, 2006, pp. 207–210.
[9] M.J. Gander, M. Petcu, Analysis of a Krylov subspace enhanced parareal algorithm for linear problems, ESAIM Proceedings 25 (2008) 114–129.
[10] M.J. Gander, A.E. Ruehli, Optimized waveform relaxation methods for RC type circuits, IEEE Transactions on Circuits and Systems I-Regular Papers 51 (2004) 755–768.
[11] M.J. Gander, S. Vandewalle, Analysis of the parareal time-parallel time-integration method, SIAM Journal of Scientific Computing 29 (2) (2007) 556–578.
[12] D. Guibert, D. Tromeur-Dervout, Adaptive parareal for systems of ODEs, in: 16th International Conference on Domain Decomposition Methods on Science and Engineering, January 12–15, New York, 2005.
[13] C. Harden, J. Peterson, Combining the parareal algorithm and reduced order modeling for time dependent partial differential equations, available: www.scs.fsu.edu/charden/myresearch.
[14] Y.L. Jiang, On time-domain simulation of lossless transmission lines with nonlinear terminations, SIAM Journal on Numerical Analysis 42 (3) (2004) 1018–1031.
[15] Y.L. Jiang, O. Wing, A note on convergence conditions of waveform relaxation algorithms for nonlinear differential-algebraic equations, Applied Numerical Mathematics 36 (2–3) (2001) 281–297.
[16] Y.L. Jiang, O. Wing, A note on the spectra and pseudospectra of waveform relaxation operators for linear differential-algebraic equations, SIAM Journal on Numerical Analysis 38 (1) (2000) 186–201.
[17] E. Lelarasmee, A.E. Ruehli, A.L. Sangiovanni-Vincentelli, The waveform relaxation method for time-domain analysis of large scale integrated circuits, IEEE Transactions on Computer-Aided Design 1 (3) (1982) 131–145.
[18] J.L. Lions, Y. Maday, G. Turinici, A "parareal" in time discretization of PDE's, Comptes Rendus del Academie des Sciences Serie I-Mathematique 332 (7) (2001) 661–668.
[19] J. Liu, Y.L. Jiang, Waveform relaxation for reaction-diffusion equations, Journal of Computational and Applied Mathematics 235 (2011) 5040–5055.
[20] Y. Maday, J. Salomon, G. Turinici, Monotonic parareal control for quantum systems, SIAM Journal Numerical Analysis 45 (6) (2007) 2468–2482.
[21] Y. Maday, G. Turinici, A parareal in time procedure for the control of partial differential equations, Comptes Rendus Mathematique 335 (4) (2002) 387–392.
[22] Y. Maday, G. Turinici, Parallel in time algorithms for quantum control: parareal time discretization scheme, International Journal of Quantum Chemistry 93 (2003) 223–228.
[23] Y. Maday, G. Turinici, The parareal in time iterative solver: a further direction to parallel implementation, in: 15th International Conference on Domain Decomposition Methods in Science and Engineering, July 21–25, Berlin, 2003.
[24] M. Minion, A hybrid parareal spectral deferred correction method, Communications in Applied Mathematics and Computational Science 5 (2010) 265–301.
[25] M. Minion, S. Williams, Parareal and spectral deferred corrections, AIP Conference Proceedings 1048 (2008) 388–391.
[26] S. Vandewalle, Multilevel Time-integration: Analysis of the Parareal Algorithm, Numerical Analysis Seminar Series, Oxford Computing Laboratory, University of Oxford, New York, 2007, February 8.
[27] S. Vandewalle, M.J. Gander, The parareal algorithm in a historical perspective, in: 16th International Conference on Domain Decomposition Methods on Science and Engineering, January 12–15, New York, 2005.