Open Mathematics

Research Article

Sunyoung Bu*

Time parallelization scheme with an adaptive time step size for solving stiff initial value problems

https://doi.org/10.1515/math-2018-0022
Received August 30, 2016; accepted December 18, 2017.

Abstract: In this paper, we introduce a practical strategy to select an adaptive time step size suitable for the parareal algorithm designed to parallelize a numerical scheme for solving stiff initial value problems. For the adaptive time step size, a technique to detect stiffness of a given system is first considered since the step size will be chosen according to the extent of stiffness. Finally, the stiffness detection technique is applied to an initial prediction step of the parareal algorithm, and select an adaptive step size to each time interval according to the stiffness. Several numerical experiments demonstrate the efficiency of the proposed method.

Keywords: Parareal Algorithm, Krylov Deferred Correction, Stiffness detection, Adaptive step size, Stiff System

MSC: 34A45, 65L04, 65L05, 65Y05

1 Introduction

We consider numerical techniques to solve stiff initial value problems (IVPs) given by

$$\frac{dy}{dt} = f(t, y(t)), \qquad t \in [t_0, t_f], \qquad y(t_0) = y_0, \tag{1}$$

where f has continuously bounded partial derivatives up to required order for the developed numerical method. The stiff systems are broadly classified into two categories - one is to have stiff components in just a given system and the other is to have the components in both the system and its solution. In the first case, solutions of the system behaves smoothly as time is increasing so it can be easily solved by any implicit scheme with an appropriate step size. On the other hand, the solutions of the given system in the second case have stiff components expressed as irregularities or sharp fronts in some or whole time intervals. In the intervals, we carefully handle a numerical scheme since the solutions are very rapidly changed. Most interesting research topics, induced from the real applications such as fluid dynamics, molecular dynamics, plasma or other physics, are related with the second case.

There are lots of numerical strategies to find efficient and accurate solutions of the stiff systems. In this paper, we focus on the parallelization scheme to find the efficient solutions of the stiff IVPs. Time parallelization scheme has received a lot of attention over the past few years and several parallelization schemes have been proposed [1–3]. Especially in 2001, a new algorithm which was named parareal algorithm for the solution of time dependent differential equations in parallel was introduced [4]. It can be defined by

$$y_{n+1}^k = G(t_{n+1}, t_n, y_n^k) + F(t_{n+1}, t_n, y_n^{k-1}) - G(t_{n+1}, t_n, y_n^{k-1}) \tag{2}$$

***Corresponding Author: Sunyoung Bu:** Hongik University, Seoul, South Korea, E-mail: syboo@hongik.ac.kr

where the subscript n refers to the time subdomain number, the superscript k refers to the iteration number. F represents a fine propagator, that is, a more accurate solution on a fine grid in time interval $[t_n, t_{n+1}]$ with an initial value y_n^{k-1}. G represents a coarse propagator, a less accurate approximation in a coarser grid. Note that the F-propagator determines the overall accuracy of the parareal method, whereas the convergence order of the method is decided by the order of the G-propagator and the number of iterations used when it is coupled with a sufficiently accurate F-propagator [5, 6]. Unfortunately, the traditional parareal scheme has the low parallel efficiency which is bounded by 1/K, where K is the parareal iterations needed to converge to the desired accuracy. In most case, 2 or more iterations are needed, so the efficiency of the traditional parareal scheme is less than 50 percents and even worst in practice. To hurdle this drawback, several advanced parareal techniques based on the deferred correction (DC) methods have been recently introduced [2, 5, 6], in which DC strategies are utilized within the parareal iteration for the F-propagator by using one or few DC iterations during each parareal iteration.

In the parareal algorithm, there is an important assumption that there are an infinite number of processors to use, so each processor is assigned to each different time interval with a uniform time step size. However, only a few finite number of processors can be provided in practice. Even if an infinite number of processors are provided, it is not efficient to assign the uniform step size on each processor without any consideration on the property of the problem, especially for stiff systems or partial differential equations (PDEs) having sharp front. That is, it is more efficient that a larger time step size is assigned for smooth or non-stiff regions in solutions, while a smaller time step size is needed for shock or stiff regions. Therefore, the usage of adaptive step sizes is very important issue to improve the efficiency of the parareal algorithm. Related to this issue, many researchers have attempted to find a suitable way to automatically detect stiffness [7].

The aims of this paper are to introduce a criterion to detect stiffness and to develop a scheme for finding an adaptive time step size according to the extent of stiffness in each interval. First of all, for the given system, we need to split the stiff and non-stiff parts in a given time domain. There are various ways to detect stiffness. For simplicity's sake, we examine a gradient ratio of a given system to split the stiff and non-stiff parts. Once the stiff regions is detected, the corresponding step size should be automatically controlled. So, the time intervals in stiff regions should be gradually shrank depending on the extent of the stiffness, while those in non-stiff regions are comparatively stretched. Based on these processes, an appropriate time step size for the parareal algorithm is chosen in the sequential step depending on the stiffness at each time interval. Note that in the traditional parareal algorithm, a G-propagator approximates initial values for all time intervals sequentially, with having a uniform step size at the initialization step.

Additionally, a theoretical analysis of the parareal algorithm shows that the stability of the method depends on the choice of G-propagator [5, 8]. Especially, for solving highly stiff problem, G-propagator should be satisfied an L-stability. Also, each time interval is determined in the initialization step with a G-propagator, the computational cost for G-propagator should be small enough. Overall, Backward Euler (BE) method will be a good candidate for the G-propagator, since BE is unconditionally stable, its computational costs is relatively small and it has L-stability [9], where it can unconditionally fulfill the stability condition of the parareal methods with less computational costs.

This paper is organized as follows. In Sec. 2, we briefly describe the original parareal technique and the improved parareal algorithms based on the original one. In Sec. 3, we introduce several parameters for detecting degree of stiffness in each interval and discuss a strategy to select adaptive time step size using stiffness detection to improve the overall efficiency of the parareal algorithm. In Sec. 4, preliminary numerical results are presented to show the efficiency of the proposed scheme. Finally in Sec. 5, future research directions are provided.

2 Parareal method

In this section, we briefly review the original parareal algorithm and the improved parareal algorithms based on the traditional algorithm.

2.1 Parareal algorithm

As in general parareal algorithm, we assume the time interval $[0, T]$ is divided into N_p intervals with each interval being assigned to a different processor denoted processors P_1 through P_{N_p}. On each interval, the parareal method iteratively computes a succession of approximation $y_{n+1}^k \approx y(t_{n+1})$, where k denotes the iteration number. It is defined using two propagation operators $G(t_{n+1}, t_n, y_n)$ and $F(t_{n+1}, t_n, y_n)$, for which the propagators search a solution from t_n to t_{n+1} using an initial value y_n. The $G(t_{n+1}, t_n, y_n)$ operator (denoted G) provides a rough approximation of $y(t_{n+1})$, the solutions of Eq. (1) with given initial conditions, whereas the $F(t_{n+1}, t_n, y_n)$ operator (denoted F) typically gives a highly accurate approximation of $y(t_{n+1})$ on the fine discretization of time interval $[t_n, t_{n+1}]$. Note that typically the G propagator is computationally less expensive than the F propagator, that is, the G propagator is usually a lower order method or computed on a much coarser discretization, while the F propagator is a higher-order method on a finer discretization. So, the parareal method is convergent to a solution of the F propagator applied in serial.

The parareal method begins by sequentially computing y_n^0 for $n = 1, \dots, N_p$, using G propagator,

$$y_{n+1}^0 = G(t_{n+1}, t_n, y_n^0). \tag{3}$$

Once each processor has a value y_n^0, the processors can in parallel compute the approximation $F(t_{n+1}, t_n, y_n^0)$. The parareal algorithm then computes the serial correction step for $n = 1, \dots, N_p$,

$$y_{n+1}^{k+1} = G(t_{n+1}, t_n, y_n^{k+1}) + F(t_{n+1}, t_n, y_n^k) - G(t_{n+1}, t_n, y_n^k). \tag{4}$$

The method proceeds iteratively alternating between the parallel computation of $F(t_{n+1}, t_n, y_n^k)$ and the serial computation of Eq. (4).

2.2 Improved parareal methods

In this subsection, we briefly introduce improved versions of parareal methods to develop for overcoming limitations of the original parareal algorithm.

Although the original parareal algorithm enables us to parallelize numerical algorithms for solving initial value problems, its efficiency is controversial since the low parallel efficiency which is bounded by 1/K, where K is the parareal iterations needed to converge to the desired accuracy. Note that K must be at least 2, the efficiency of the original parareal scheme is less than 50 percents and even worse in practice. To improve the low parallel efficiency for the parareal algorithm, several improved algorithms are developed [2, 5, 11], in which various deferred correction (DC) techniques are embedded into the parareal framework. In [2, 11], a hybrid parareal spectral deferred correction method was introduced in which spectral deferred correction (SDC) strategies are utilized within the parareal iteration, as a F-propagator. Also, in [5], two different deferred correction schemes, modified DC technique combined with Backward Euler (BE) method and Krylov deferred correction (KDC) [10], are used for G and F propagators in the parareal framework, similar to the hybrid parareal spectral deferred correction method. Commonly in [2, 5, 11], instead of directly using the SDC scheme or KDC scheme requiring several iterations (SDC sweeps in SDC or Newton-Krylov iterations in KDC) in serial, the F-propagator in each parareal iteration performs one or a few SDC sweeps or Newton-Krylov iterations on the solution from the previous parareal iteration. As the parareal iterations converge, the F solution still converges to the high-accuracy SDC or KDC solution.

The advantage of these schemes is that the F-propagator becomes much cheaper by combining the parareal iterations and DC iterations, compared to a full accurate solver. Because the DC iterations (SDC sweeps in SDC or Newton-Krylov iterations in KDC) are overlapped with the parareal iteration, so the hybrid parareal schemes can unite the two different iterations (DC and parareal iterations) [2, p. 281]. Typically, the original efficiency is bounded by $1/K$, where K is the number of iterations for the parallel iterations to converge. However, the parallel efficiency of the hybrid parareal SDC or KDC is about K_s/K, where K_s is the number of iterations required of the serial SDC or KDC method to converge to a given tolerance. Note that the hybrid parareal SDC method is a reasonably good choice for non-stiff systems and parareal KDC method is suitable for stiff systems since KDC was developed to overcome the limitation of SDC for stiffness.

3 Algorithm

In the parareal algorithm, all processors are typically initialized by using the coarse propagator in a serial way to yield a low accuracy initial condition on each interval which is assigned to each processor. So, all processors except the first one are idle until passed an initial condition from the previous processor and the idle time is inevasible.

3.1 Stiffness detection

We now describe a parameter which dictates whether there exists a stiffness of a given system in a given time interval. Since stiffness implies that the given system has two different time scales. That is, a rate of maximum and minimum value of derivatives for the system in some interval is quite big, then we prescribe that it is stiff in the interval. In this respect, this can be easily done by introducing the following measure related to the gradient for the given problem :

$$\kappa(t) = \begin{cases} \dfrac{\max \|y'(t)\|}{\min \|y'(t)\|}, & n > 1 \\ \|y'(t)\|, & n = 1 \end{cases} \tag{5}$$

where n is the dimension of the given system.

Note that we restrict the stiffness to the case when stiffness is on the problem and solution. That is, we need to check the change rate of $\kappa(t)$ since it says whether any big change exists between the previous interval and the current one. This allows us to approximate a conditioning parameter $\gamma(h)$ in each interval as follows:

$$\gamma(h) = \frac{\min(\kappa(t_m), \kappa(t_{m+1}))}{\max(\kappa(t_m), \kappa(t_{m+1}))}, \tag{6}$$

where $h = t_{m+1} - t_m$. We can easily check that $\gamma(h)$ goes to 0 when there is a big difference between $\kappa(t_m)$ and $\kappa(t_{m+1})$ due to big change in the interval $[t_m, t_{m+1}]$. Also, $\kappa(t_m)$ and $\kappa(t_{m+1})$ have quite similar values, $\gamma(h)$ goes to 1. That is, the parameter $\gamma(h)$ can be used to detect the stiffness in the interval $[t_m, t_{m+1}]$, so we say it “stiffness ratio". Therefore, the stiffness ratio $\gamma(h)$ is used to determine a time step size h according to the degree of the stiffness, since the ratio $\gamma(h)$ is relatively small when solutions in a given time interval are rapidly changed and $\gamma(h)$ is large when a change rate of solutions is increasingly small. Note that “the change rate of solutions is small" means that the solutions are smooth in that interval, so the time step size is allowed to be large. Hence the time step size can be selected adaptively small and increasingly large when the stiffness ratio $\gamma(h)$ is small and relatively large, respectively.

3.2 Adaptive step size selection

For a code implementation, stiffness criteria to choose time step sizes are needed to be set up. When the ratio $\gamma(h)$ is close to 1, the step size is expanded and when the ratio goes to 0, the step size should be shortened. For the sake of simplicity, we amplify the step size twice when the stiffness ration is 1, which means that there is no difference of $\kappa(t)$ in an interval. In addition, the step size is reduced by half when the stiffness ratio is halved. Using these conditions, we simply set up the following criterion to choose a new time step size h_{new} as follows:

$$h_{new} = \min(h_{max}, \max(h_{min}, 2h \cdot \gamma(h)^2)), \tag{7}$$

where h_{max} and h_{min} are constant factors to avoid too fast increase and decrease of the time step, respectively.

Based on the parameters and the discussion above, we get the following algorithm to select new step size as follows:

Algorithm for step size selection

1. Remark: The algorithm is designed to adaptively choose time step size for G-propagator in the parereal scheme. $[t_0, t_{final}]$ is the required integration interval, and y_0 is a given initial value.
2. Initialize h_0, $t_{old} := t_0$.
3. Set $t_{new} := t_{old} + h$. If $t_{new} > t_{final}$, then exit.
4. Perform the Backward Euler method with having h and y_{old} and approximate y_{new}.
5. Calculate Jacobian matrix and its eigenvalue of the matrix and get $\kappa(t)$ and $\gamma(h)$ defined in (5) and (6), respectively.
6. $h_{new} = \min(h_{max}, \max(h_{min}, 2h \cdot \gamma(h)^2))$
7. Setting $h = h_{new}$, $t_{old} := t_{new}$ and $y_{old} := y_{new}$, go to step 3.

3.3 Parareal algorithm with the proposed adaptive step size controller

Based on the new step size controller with the stiffness detection discussed in the previous subsections, we present the enhanced parareal methods with adaptive step size. First of all, we assume the time interval of interest $[0, T]$ is divided into N uniform intervals, and each interval $[t_i, t_{i+1}]$ is assigned to a corresponding processor P_i. Note that y_i^k denotes the approximation after the k-th parareal iteration at the i-th node t_i.

Predictor Step Decide initial values and step sizes in a serial way

Starting with the initial value y_0 and h_0,

- Get the initial value y_i^0 and h_i from the previous interval $[t_{i-1}, t_i]$
- Using G-propagator, calculate the initial approximation y_{i+1}^0 for $t = t_{i+1}$ on Processor P_i in serial.

$$G(t_{i+1}, t_i, y_i^0) = y_{i+1}^0. \tag{8}$$

- Using the step size selection technique discussed above, calculate a new step size h_{new} based on κ and γ.
- Send the y_{i+1}^0 and h_{new} to the next interval.

Corrector Step Parallel Iteration ($k + 1$ step) for $k = 0, \ldots, N - 1$

- Using a higher order method, compute $F(t_{n+1}, t_n, y_n^k)$ on the fine grid in parallel.
- After the approximation value y_i^{k+1} at t_i on each processor P_{i-1} is calculated, it is sending to the following processor P_i as a new initial value for t_{i+1}.
- Using G-propagator with a new initial value y_i^{k+1}, calculate the initial approximation y_{i+1}^{k+1} for $t = t_{i+1}$ on Processor P_i.
- Update

$$y_{n+1}^{k+1} = G(t_{n+1}, t_n, y_n^{k+1}) + F(t_{n+1}, t_n, y_n^k) - G(t_{n+1}, t_n, y_n^k), \tag{9}$$

  where $G(t_{n+1}, t_n, y_n^k)$ is the approximation from G-propagator.

4 Numerical results

In this section, preliminary numerical results are presented to examine the convergence behavior and efficiency of the enhanced parareal scheme, compared to the standard implementation of the original parareal scheme.

4.1 Robertson example

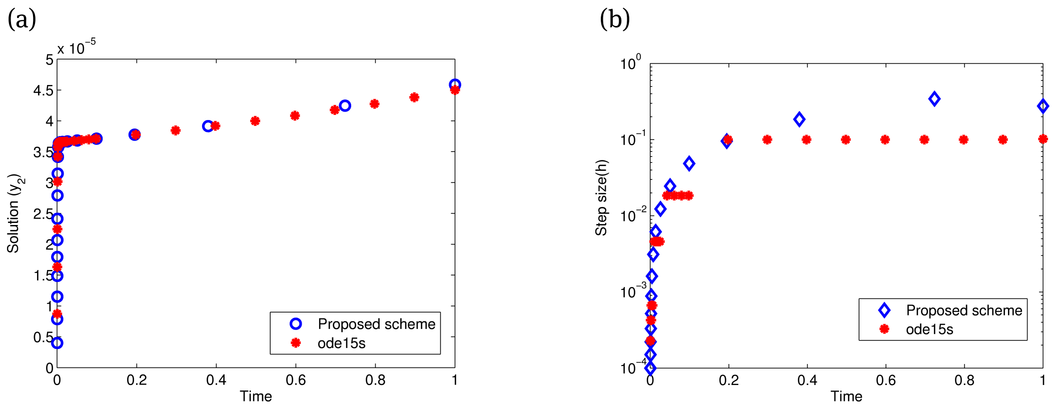
As the first example, we solve a classical problem due to Robertson which describes the kinetics of an autocatalytic reaction given by Robertson. Its system consists of a stiff system of 3 nonlinear ODEs given by

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3, \\ y_2' &= 0.04y_1 + 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2, \\ y_3' &= 3 \cdot 10^7 y_2^2. \end{aligned} \tag{10}$$

The initial vector y_0 is given by $[1, 0, 0]^T$. In this experiment, we march from $t = 0$ to $t_f = 1$ with initial step size $h_0 = 1e - 4$. To investigate the effectiveness of the proposed scheme, we simply experiment with the adaptive mesh selection using Backward Euler (BE) method and compare the accuracy of the proposed scheme that of the existing method (built-in Matlab function -ode15s).

Note that in this experiment, BE is used as a test method since BE is employed for G-propagator of the parareal algorithm in this paper. Fig. 1(a) shows that the proposed step size controller can solve the problem and its result is quite close to that from the existing method ode15s.

Fig. 1. Comparison of (a) solution and (b) corresponding time step size for proposed scheme and existing method (ode15s)



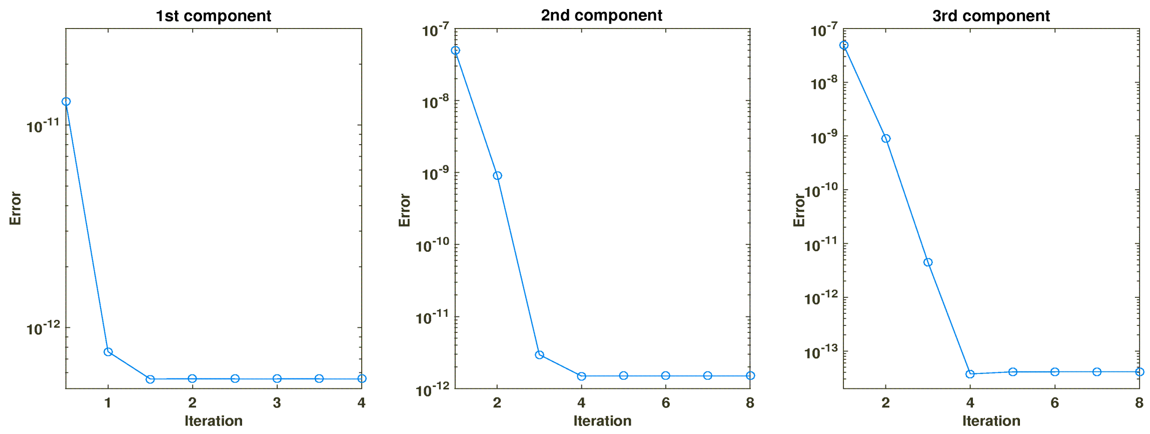
Note that we just plot the second component of the solution set since the stiffness is on the second component. Also the stiffness is near $t = 0$, so we expect the step size is chosen relatively small in this region. For this, we plot the time step size by the proposed scheme over the time domain and compare it with that obtained by the existing method.

Fig. 1(b) shows that only 21 time steps are needed to reach the final time with the proposed scheme, while 10^4 time steps are required with uniform grid used in the original parareal scheme and 30 time steps with the existing scheme using adaptive time step size. Also, as seen in the figure, larger time step sizes are allowed by the proposed technique in non-stiff parts as desired.

Now we apply the adaptive mesh selection strategy to the parareal algorithm with KDC as a F-propagator and BE as a G-propagator. Also, for the experiment, the KDC methods with 4 Radau II nodes are employed, and each parareal iteration performs the 2 outer Newton iterations for desired efficiency and the other conditions such as the tolerance for the Newton-Krylov methods or nonlinear solvers are fixed for all simulations. We use a reference solution obtained from KDC scheme with 8 Radau II node and full outer Newton iterations.

To examine the convergence behavior of stiff parts, only 10 processors are used and corresponding final time point is 0.00125. In Fig. 2, we plot the error at the final time (t = 0.00125) versus the parareal iterations with an initial step size 10^4. It can be seen that after a certain number of parareal iterations, the error levels reach a certain tolerance level even for stiff parts. It also shows that to reach the final time 0.00125 using uniform grid with a step size $h = 10^4$, it requires 10 processors which is the same number of processors needed for the adaptive mesh parareal scheme. Hence, it must be noted that even using adaptive step sizes, the step sizes in stiff parts should become small enough.

Fig. 2. Convergence behavior of parareal iterations for each component of the given system



4.2 Van der Pol problem

This problem, that models the behavior in an electronic circuit, can be described as a system of two equations given by

$$y'(t) = f\big(t, y(t)\big), \quad t \in (0,\ 3000]; \quad y(0) = y_0, \tag{11}$$

where $y(t) = [y_1(t),\ y_2(t)]^T \in \mathbb{R}^2$ and f is defined by

$$f\big(t, y(t)\big) = \begin{bmatrix} y_2(t), \\ \big((1 - y_1^2(t))y_2(t) - y_1(t)\big)/\epsilon \end{bmatrix}. \tag{12}$$

with initial condition $y_0 = [2,\ 0]^T$. From the second component of f, it can be seen that the smaller ϵ is, the stronger stiffness of problem is. For the test, we take $\epsilon = 1/1000$ and initial time step size $h_0 = 10^{-4}$.

To examine the effectiveness of the proposed scheme, we plot the solution y_1 over the time domain and its corresponding step size h in Fig. 5. The figure shows that the step size is adaptively chosen to be small in stiff parts and increasingly large in non-stiff parts. The figure also shows that only 591 time steps are needed to reach the final time with the proposed scheme. Note that regardless of stiffness, the original parareal scheme have to use the uniform step size using appropriate step size suitable for stiff components. For example, if the initial step size $h = 10^{-4}$, then 3×10^7 time steps are required. It is directly related to the computational time and the number of processes needed in parareal scheme.

Now we apply the adaptive mesh selection strategy to the parareal algorithm with KDC as a F-propagator and BE as a G-propagator. Also, for the experiment, the KDC methods with several Radau II nodes are employed, and each parareal iteration performs the 2 outer Newton iterations for desired efficiency and the other conditions such as the tolerance for the Newton-Krylov methods or nonlinear solvers are fixed for all simulations.

Since only a few processors (less than 100) can be available in current status, we test parareal algorithm with adaptive step sizes only for non-stiff parts.

Using 72 processors, we march $t = 0$ to $t_f = 569.6$ with adaptive time step size and plot the adaptive step sizes over the time domain in Fig. 5. The adaptive step size is almost same as seen in Fig. 5.

With the adaptive step size, we generate numerical results from the parareal algorithm with $3, 4$ and 6 Radau II nodes for F-propagator (KDC) to examine the convergence behavior. Note that we calculate a

Fig. 3. Approximated solution (y_1) behavior (above), approximated solution (y_2) behavior (middle) and corresponding time step size (below)

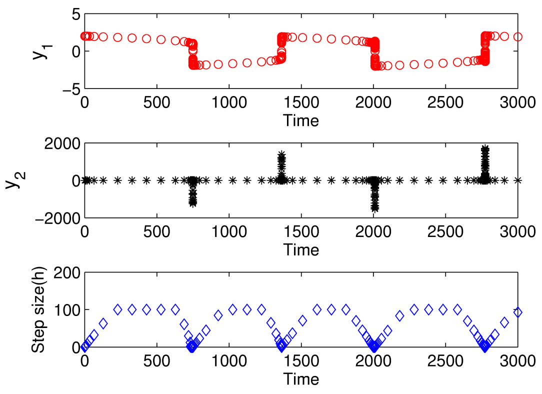
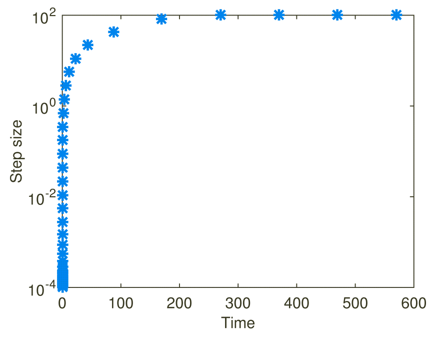


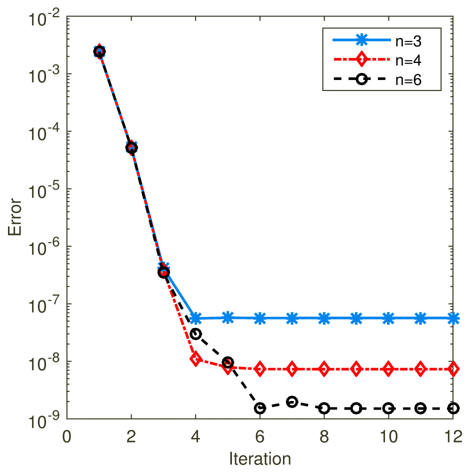
Fig. 4. Comparison of corresponding time step size for proposed scheme and existing method (ode15s)



numerical solution at time $t_f = 569.6$ for the KDC method with 8 Radau II node and full outer Newton iterations for a reference solution, since analytic solutions of this problem are unknown.

For the experiment, we plot the error based on the reference solutions for the parareal iteration in Fig. 5. It can be seen that the accuracy of the algorithm after convergence depends on the number of Radau IIa collocation nodes in the KDC methods. Note that the KDC methods using p Radau IIa nodes is converging with an approximate order of $2p - 1$ [5, 12, 13].

Fig. 5. Convergence behavior of parareal algorithm with 3, 4 and 6 Radau IIa nodes



5 Discussion

In this paper, a numerical technique to select an adaptive step size is introduced to improve the efficiency of the parareal algorithm for stiff systems. Unlike the traditional parareal scheme, the proposed scheme allows us to use much larger time step size in non-stiff parts so that it can reduce the number of processors assigned to the corresponding interval and lead to less computational costs without any damage on accuracy.

Currently, we are working on the generalization of adaptive step size selection for any G-propagator. In particular, the proposed technique is just applicable for explicit type ODE systems, but not Differential Algebraic Equations (DAEs). In relation to this, we are constructing other parameters to measure stiffness of the given systems. At the same time, we are applying the proposed scheme to time dependent PDEs. Preliminary results are quite promising. Results along these directions will be reported soon.

Acknowledgement: This work was supported by the Hongik University new faculty research support fund.

References

[1] Butcher J. C., Order and stability of parallel methods for stiff problems: parallel methods for odes, Adv. Comput. Math., 1997, **7**, 79–96.
[2] Minion L. M., A hybrid parareal spectral deferred corrections method, Comm. in App. Math. and Comput. Sc., 2011, **5**(2), 265–301.
[3] Vandewalle S., and Roose D, The parallel waveform relaxation multigrid method, in: Proceedings of the Third SIAM Conference con Parallel Processing for Sientific Computing, in: Soc. Indust. Appl. Math., 1989, 152–156.
[4] Lions J. J., Maday Y., and Turinici G., A parareal in time discretization of PDE's, C.R. Acad. Sci. Paris, Serie I, 2001, 332(1):16.
[5] Bu S., and Lee J., An enhanced parareal algorithm based on the deferred correction methods for a stiff system, J. Comput. Appl. Math. 2014, **255** 297–305.
[6] Emmett M., and Minion L. M., Toward an efficient parallel in time method for partial differential equations, Comm. in App. Math. and Comput. Sc., 2012, **7**(1), 105–132.
[7] Shampine F. L., Stiff and nonstiff differential equation solvers, II: Detecting stiffness with Runge-Kutta methods., ACM Trans. Math. Softw, 1977, **3(1)**, 44–53.
[8] Staff A. G., and, Ronquist M. E., Stability of the parareal algorithm, in: Proceedings of the 15th International Domain Decomposition Conference, in: Lect. Notes Comput. Sci., 2003.
[9] Hairer E., and Wanner G., Solving ordinary differential equations II, Springer, 1996.
[10] Bu S., Huang J., and Minion L. M., Semi-implicit Krylov deferred correction methods for differential algebraic equations, Math. Comput., 2012, **81**(280) 2127–2157.
[11] Minion L. M., and Willimas S., Parareal and spectral deferred corrections, In AIP Conference Proceedings, 2008, **1048**, 388–391.
[12] Huang J, Jia J., and Minion L. M., Accelerating the convergence of spectral deferred correction methods, J. Comput. Phys., 2006, **214**(2), 633–656.
[13] Huang J., Jia J., Minion L. M., Arbitrary order Krylov deferred correction methods for differential algebraic equations, J.Comput. Phys., 2007, **221**,(2), 739–760.