Parareal and Spectral Deferred Corrections

Michael L. Minion and Sarah A. Williams

Mathematics Dept., CB 3250, University of North Carolina, Chapel Hill, NC, 27599, USA

Abstract. A new class of iterative time parallel methods for initial value ordinary differential equations are developed. Methods based on a parallel variation of spectral deferred corrections (SDC) are compared and contrasted with the parareal method. It is shown that there is a strong similarity between the serial step in the parareal algorithm and the correction step in the SDC method. This observation is used to construct a hybrid strategy combining features of both the parareal and SDC methods which can significantly reduce the computational cost of each iteration compared to parareal. A numerical example is presented to compare the effectiveness of the hybrid strategies.

Keywords: parallel computing, ordinary differential equations, deferred corrections, parareal
PACS: 02.60.Lj

SPECTRAL DEFERRED CORRECTIONS

The spectral deferred correction method (SDC) is a variant of the traditional deferred and defect correction methods for ODEs introduced in the 1960s [1, 2, 3]. The original methods never gained the popularity of Runge-Kutta or linear multi-step methods; however, a recent series of papers beginning in 2000 has rekindled interest in using such methods for large scale physical simulations. The SDC method introduced in [4] couples a Picard integral formulation of the correction equation with spectral integration rules to achieve stable explicit and implicit methods with arbitrarily high formal order of accuracy. SDC also provides the flexibility to apply different time-stepping procedures to different terms in an equation (as in operator splitting methods) while maintaining the high formal order of accuracy. Specifically, the construction of schemes which treat non-stiff terms in the equation explicitly and multiple stiff terms implicitly but independently has been demonstrated [5]. In the current context, we show that the iterative nature of SDC methods is advantageous when used within parallel ODE methods which are themselves iterative.

An overview of the SDC method is now presented. Consider the ODE initial value problem

$$\mathbf{u}'(t) = f(t,\mathbf{u}(t)), \qquad \mathbf{u}(0) = \mathbf{u}_0 \tag{1}$$

for $t \in [0,T]$, where $\mathbf{u}_0, \mathbf{u}(t) \in \mathbb{C}^N$ and $f : \mathbb{R} \times \mathbb{C}^N \to \mathbb{C}^N$. This equation is equivalent to the Picard integral equation

$$\mathbf{u}(t) = \mathbf{u}_0 + \int_0^t f(\tau,\mathbf{u}(\tau))d\tau, \tag{2}$$

and this latter form is used extensively in the discussion that follows.

As with traditional deferred correction methods, a single time step $[t_n, t_{n+1}]$ is divided into a set of intermediate substeps defined by $\vec{t}_n = [t_{n,0}, \cdots, t_{n,P}]$; however, for SDC methods, $\vec{t}_n$ corresponds to Gaussian quadrature nodes. Here we use Gauss-Lobatto nodes so that $t_{n,0} = t_n$ and $t_{n,P} = t_{n+1}$. Next, a provisional approximation $\vec{U}_n^0 = [U_{n,0}^0, \cdots, U_{n,P}^0]$ is computed using a standard numerical method with $U_{n,0}^0 = U_n^0$ as the initial value. We denote generic substep values with the subscript m, and when the context is clear, the n subscript is omitted to avoid clutter, e.g., $U_{n,m}^0 = U_m^0$. Let $U_n^0(t)$ denote the continuous counterpart of $\vec{U}_n^0$ constructed by standard interpolation. Then, an integral equation similar to (2) for the error $\delta(t) = \mathbf{u}(t) - U_n^0(t)$ is

$$\delta(t) = \int_{t_n}^t [f(\tau, U_n^0(\tau) + \delta(\tau)) - f(\tau, U_n^0(\tau))]\, d\tau + \varepsilon_n(t), \tag{3}$$

where

$$\varepsilon_n(t) = U_n^0 + \int_{t_n}^t f(\tau, U_n^0(\tau))d\tau - U_n^0(t). \tag{4}$$

Note that $\varepsilon_n(t_m)$ can be accurately and stably approximated using spectral integration [6] since the provisional solution $U_n^0(t)$ is known at the Gaussian quadrature nodes. Hence we define the approximations

$$\varepsilon_{n,m}^0 = U_n^0 + S_n^m(f(\vec{t}, \vec{U}_n^0)) - U_n^0(t_m), \tag{5}$$

where S_n^m denotes a spectral integration rule which approximates the definite integral from t_n to $t_{n,m}$. A low-order time-stepping method is then applied to (3) to approximate $\delta(t)$ at the points t_m resulting in a correction to the provisional solution. For example, an explicit time-stepping scheme similar to the forward Euler method is

$$\delta_{m+1}^0 = \delta_m^0 + \Delta t_m[f(t_m, U_m^0 + \delta_m^0) - f(t_m, U_m^0)] + \varepsilon_{m+1}^0 - \varepsilon_m^0. \tag{6}$$

The provisional solution is then updated by adding to it the approximation of the correction, and the SDC method then proceeds iteratively, i.e., $U_m^{k+1} = U_m^k + \delta_m^k$. Each SDC iteration raises the formal order of accuracy of the numerical solution by the order of the approximation to (3) provided the quadrature rule in (5) is sufficiently accurate [4, 7]. Lastly, using (5), a direct update form of (6) can be derived

$$U_{m+1}^{k+1} = U_m^k + \Delta t_m[f(t_m, U_m^{k+1}) - f(t_m, U_m^k)] + S_m^{m+1}(\vec{t}_n, f(\vec{U}_n^k)). \tag{7}$$

This form is compared below to the serial step in the parareal algorithm.

THE PARAREAL METHOD

The parareal method was introduced in 2001 by Lions, Maday, and Turinici [8]. Unlike earlier attempts to parallelize the individual steps within a standard ODE solver (see, e.g., [9]), the parareal method employs an iterative strategy over the total interval of integration. The convergence of the parareal algorithm is considered in [10], and the stability of some variations is considered in [11]. Gander and Vandewalle have shown that the parareal method can be interpreted as either a space-time multigrid method or as a multiple shooting method [12]. After introducing the parareal method, we discuss the advantage of using an iterative solver like SDC within the parareal framework and examine the connection between the parareal method and the correction equation in SDC.

The general strategy for the parareal method is to divide the time interval of interest $[0, T]$ into N_p intervals with each interval being assigned to a different processor. To simplify the discussion, denote the processors $\mathbf{P}_0$ through $\mathbf{P}_{N_p-1}$, and consider time intervals of uniform size $\Delta t = T/N_p$ so that the nth processor computes the solution on the interval $[t_n, t_{n+1}]$ where $t_n = n\Delta t$. On each interval, the parareal method iteratively computes a succession of approximations $U_{n+1}^k \approx \mathbf{u}(t_{n+1})$, where k denotes the iteration number.

It is becoming standard to describe the parareal algorithm in terms of two numerical approximation methods denoted G and F. Both G and F propagate an initial value U_n by approximating the solution to (1) from t_n to t_{n+1}. For example, if G is defined by the forward Euler method, then

$$G(t_{n+1}, t_n, U_n) = U_n + \Delta t\, f(t_n, U_n). \tag{8}$$

Note that G or F can be defined to be more than one step of a particular numerical method on the interval $[t_n, t_{n+1}]$. As discussed below, in order for the parareal method to be efficient, it must be the case that the G propagator is computationally less expensive than the F propagator, hence in practice, G is usually a low-order method or computed on a much coarser time step than F. Since the overall accuracy of parareal is limited by the accuracy of the F propagator, F is typically higher-order.

The parareal method begins by sequentially computing U_n^0 for $n = 1 \ldots N_p$, often using G, i.e.,

$$U_{n+1}^0 = G(t_{n+1}, t_n, U_n^0). \tag{9}$$

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

$$U_{n+1}^{k+1} = G(t_{n+1}, t_n, U_n^{k+1}) + F(t_{n+1}, t_n, U_n^k) - G(t_{n+1}, t_n, U_n^k). \tag{10}$$

The method proceeds iteratively alternating between the parallel computation of $F(t_{n+1}, t_n, U_n^k)$ and the serial computation of (10) (which requires computing the G propagator).

Parareal is an iterative method and hence requires a stopping criteria. Note that after k iterations of the parareal method, the solution U_m^k for $m \leq k$ is exactly equal to the numerical solution obtained by using the F scheme sequentially. Hence after N_p iterations, the parareal solution is exactly equal to applying F sequentially. Each iteration of the parareal method requires the application of both F and G (plus the cost of communication between processors). Hence, the parareal method can only provide significant parallel efficiency compared to the sequential F scheme if the number of iterations required to converge to the specified criteria (denoted here by K) is significantly less than N_p and the cost of the serial propagator G is significantly less than that of F.

PARAREAL DEFERRED CORRECTIONS

Since the parareal algorithm can in principle use any numerical method for the F and G propagators, it would be straightforward to incorporate SDC into the parareal framework. However, after the first parareal iteration, it would be foolish to ignore the results of the F propagator from the parareal iteration $k-1$ when using SDC in iteration k. One could instead define the F propagator to be p SDC sweeps applied to the previous F solution (incorporating the updated boundary condition as well). As the parareal iterations converge, the F solution still converges to the high-accuracy SDC solution (in fact to the spectral collocation solution [7]), but the cost of applying the F propagator during each iteration is now only p times that of a low-order method. The numerical examples here suggest that $p = 1$ is sufficient for an efficient method (see the discussion below).

Furthermore we establish the connection between deferred corrections and the parareal step defined by (10). Both F and G are approximations to the exact update given by the Picard equation

$$\mathbf{u}(t_{n+1}) = \mathbf{u}(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, \mathbf{u}(\tau)) d\tau. \tag{11}$$

We symbolize this correspondence by implicitly defining Q_n^{n+1} and I_n^{n+1} to be approximations to the integral term above so that

$$F(t_{n+1}, t_n, U_n^k) = U_n^k + I_n^{n+1} f(t, U_n^k), \qquad \text{and} \qquad G(t_{n+1}, t_n, U_n^k) = U_n^k + Q_n^{n+1} f(t, U_n^k). \tag{12}$$

Using these definitions, (10) can be rewritten

$$U_{n+1}^{k+1} = U_n^k + Q_n^{n+1} f(t, U_n^{k+1}) - Q_n^{n+1} f(t, U_n^k) + I_n^{n+1} f(t, U_n^k). \tag{13}$$

Note the similarity between this equation and (7). In the discussion leading to (7), we discretize (3) using forward Euler to give a concrete example of a time stepping scheme. If G is similarly defined as a single step of forward Euler, then $Q_n^{n+1} = \Delta t f(t_n, U_n^k)$, and (13) becomes

$$U_{n+1}^{k+1} = U_n^k + \Delta t (f(t_n, U_n^{k+1}) - f(t_n, U_n^k)) + I_n^{n+1} f(t, U_n^k), \tag{14}$$

which is nearly identical to (7). Hence the parareal step (10) can be considered a particular incarnation of a deferred correction step. However, in the parareal method, (10) only provides an updated initial condition for the following parallel F propagator. Instead G can be applied as a coarse SDC sweep, which updates the initial condition for the next time step and also improves the current solution in the nth interval.

NUMERICAL EXAMPLE

Here the effectiveness of the hybrid parareal/SDC method is explored using the Lorenz equation test problem from [10]. Specifically, we use $N_p = 180$ processors to solve

$$x' = \sigma(y - x), \quad y' = x(\rho - z) - y, \quad z' = xy - \beta z. \tag{15}$$

in $t \in [0, 10]$ with the usual choice of parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$ and initial conditions $(x, y, z)(0) = (5, -5, 20)$. Two parareal methods using a Runge-Kutta (RK) method for the G and F propagator will be compared to two methods using one SDC sweep in place of F and G. Specifically, the first RK variant is the same as the example

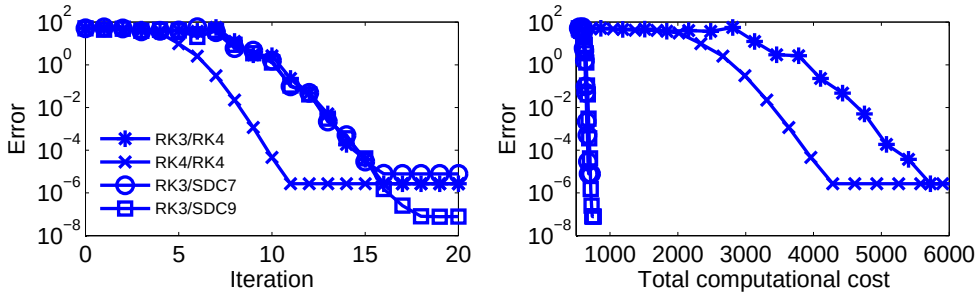

FIGURE 1. Comparison of parareal methods based on Runge-Kutta and SDC

in [10] where G is one step and F is 80 steps of fourth-order RK. The second RK method uses the same F, but G is third-order Runge-Kutta. The SDC variants both use one step of a third-order Runge-Kutta method applied to (3) in place of G, and the F propagator is replaced by one sweep of (6) using either 9 or 7 Gauss-Lobatto nodes in $[t_n, t_{n+1}]$ the interval. In the SDC methods, after the G propagator is computed, the correction to the solution is interpolated back to the Gauss-Lobatto nodes using the three stage values from the RK method.

To compare the cost of these methods, let τ_G and τ_F denote the cost of applying the G and F propagators respectively. Ignoring communication overhead, the total computational cost of K iterations of parareal using N_p processors is then $N_p\tau_G + K(\tau_G + \tau_F)$. This assumes independent processors, so that each processor can begin computation of F as soon as the preceding G procedure is completed. A reasonable approximation is to define τ_G and τ_F in terms of number of function evaluations. For the first RK method, $\tau_G = 4$, $\tau_F = 320$, and hence the total cost is $720 + K324$, while for the second it is $540 + K323$. For the SDC based methods here, $\tau_G = 3$, and τ_F is either 6 or 8. This means the total cost is $540 + K9$ and $540 + K11$ respectively. Note this ignores the cost of interpolating the correction and computing (5) (both of which are simple matrix multiplications).

Figure 1 displays the error versus both iteration number (left plot) and computational cost (right plot). To compute the error, the solutions are compared to a reference solution, and the L_2 norm in space, L_∞ norm in time of the difference is computed. The plot on the left demonstrates that the accuracy of the G propagator largely determines the number of iterations required to converge, while the accuracy of the F propagator determines the overall accuracy. The right-hand figure demonstrates the dramatic savings in cost of using the SDC procedure in place of the F propagator. It should be noted that a similar accuracy in the RK-based methods can be achieved using only 6 steps of the 8th-order 11-step method from [13], which reduces the cost of the F propagator from 320 to 66.

These results suggest that using an iterative process based on SDC rather than traditional deterministic methods within a parareal framework results in a significant decrease in the overall computational cost of the method. Careful numerical and analytical analysis of these methods is in progress.

REFERENCES

1. P. Zadunaisky, "A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations," in *The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings of International Astronomical Union, Symposium 25*, edited by G. Contopoulus, 1964.
2. V. Pereyra, *Numer. Math.* **8**, 376–391 (1966).
3. V. Pereyra, *Numer. Math.* **10**, 316–323 (1966).
4. A. Dutt, L. Greengard, and V. Rokhlin, *BIT* **40**, 241–266 (2000).
5. A. Bourlioux, A. T. Layton, and M. L. Minion, *J. Comput. Phys.* **189**, 351–376 (2003).
6. L. Greengard, *SIAM J. Numer. Anal.* **28**, 1071–1080 (1991).
7. J. Huang, J. Jia, and M. L. Minion, *J. Comput. Phys.* **214**, 633–656 (2006).
8. J.-L. Lions, Y. Maday, and G. Turinici, *C.R. Acad. Sci. Paris, Serie I* **332**, 661–668 (2001).
9. K. Burrage, *Applied Numerical Mathematics: Transactions of IMACS* **11**, 5–25 (1993).
10. M. J. Gander, and E. Hairer, "Nonlinear Convergence Analysis for the Parareal Algorithm," in *Proceedings of the 17th international domain decomposition conference*, edited by U. Langer, O. Widlund, and D. Keyes, Springer LNCSE, 2007.
11. G. Staff, and E. M. Rønquist, *Stability of the Parareal Algorithm*, Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2003.
12. M. J. Gander, and S. Vandewalle, *SIAM J. Sci. Comput.* **29**, 556–578 (2007).
13. G. J. Cooper, and J. H. Verner, *SIAM Journal of Numerical Analysis* **9**, 389–405 (1972).