

*Communications in  
Applied  
Mathematics and  
Computational  
Science*

**REVISIONIST INTEGRAL DEFERRED  
CORRECTION  
WITH ADAPTIVE STEP-SIZE CONTROL**

ANDREW J. CHRISTLIEB, COLIN B. MACDONALD,  
BENJAMIN W. ONG AND RAYMOND J. SPITERI

vol. 10 no. 1 2015



## REVISIONIST INTEGRAL DEFERRED CORRECTION WITH ADAPTIVE STEP-SIZE CONTROL

ANDREW J. CHRISTLIEB, COLIN B. MACDONALD,  
BENJAMIN W. ONG AND RAYMOND J. SPITERI

Adaptive step-size control is a critical feature for the robust and efficient numerical solution of initial-value problems in ordinary differential equations. In this paper, we show that adaptive step-size control can be incorporated within a family of parallel time integrators known as revisionist integral deferred correction (RIDC) methods. The RIDC framework allows for various strategies to implement step-size control, and we report results from exploring a few of them.

### 1. Introduction

The purpose of this paper is to show that local error estimation and adaptive step-size control can be incorporated in an effective manner within a family of parallel time integrators based on revisionist integral deferred correction (RIDC). RIDC methods, introduced in [10], are “parallel-across-the-step” integrators that can be efficiently implemented with multicore [10; 6], multi-GPGPU [4], and multinode [9] architectures. The “revisionist” terminology was adopted to highlight that (1) RIDC is a revision of the standard integral defect correction (IDC) formulation [12], and (2) successive corrections, running in parallel but (slightly) lagging in time, revise and improve the approximation to the solution.

RIDC methods have been shown to be effective parallel time-integration methods. They can typically produce a high-order solution in essentially the same amount of wall-clock time as the constituent lower-order methods. In general, for a given amount of wall-clock time, RIDC methods are able to produce a more accurate solution than conventional methods. These results have thus far been demonstrated with constant time steps. It has long been accepted that local error estimation and adaptive step-size control form a critical part of a robust and efficient strategy for solving initial-value problems in ordinary differential equations (ODEs), in particular problems with multiple timescales; see [15], for example. Accordingly, in order to assess the practical viability of RIDC methods, it is important to establish

---

*MSC2010:* 65H10, 65L05, 65Y05.

*Keywords:* initial-value problems, revisionist integral deferred correction, parallel time integrators, local error estimation, adaptive step-size control.

whether they can operate effectively with variable step sizes. It turns out that there are subtleties associated with modifying the RIDC framework to incorporate functionality for local error estimation and adaptive step-size control: there are a number of different implementation options, and some of them are more effective than others.

The remainder of this paper is organized as follows. In Section 2, we review the ideas behind RIDC as well as strategies for local error estimation and step-size control. We then combine these ideas to propose various strategies for RIDC methods with error and step-size control. In Section 3, we describe the implementation of these strategies within the RIDC framework and suggest avenues that can be explored for a production-level code. In Section 4, we demonstrate that the use of local error estimation and adaptive step-size control inside RIDC is computationally advantageous. Finally, in Section 5, we summarize the conclusions reached from this investigation and comment on some potential directions for future research.

## 2. Review of relevant background

We are interested in numerical solutions to initial-value problems (IVPs) of the form

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

where  $y(t) : \mathbb{R} \rightarrow \mathbb{R}^m$ ,  $y_a \in \mathbb{R}^m$ , and  $f : \mathbb{R} \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ . We first review RIDC methods, a family of parallel time integrators that can be applied to solve (1). Then, we review strategies for local error estimation and adaptive step-size control for IVP solvers.

**2.1. RIDC.** RIDC methods [10; 6; 4] are a class of time integrators based on integral deferred correction [12] that can be implemented in parallel via pipelining. RIDC methods first compute an initial (or *provisional*) solution, typically using a standard low-order scheme, followed by one or more corrections. Each correction revises the current solution and increases its formal order of accuracy. After initial startup costs, the predictor and all the correctors can be executed in parallel. It has been shown that parallel RIDC methods with uniform step-sizes give almost perfect parallel speedups [10]. In this section, we review RIDC algorithms, generalizing the overall framework slightly to allow for nonuniform step-sizes on the different correction levels.

We denote the nodes for correction level  $\ell$  by

$$a = t_0^{[\ell]} < t_1^{[\ell]} < \dots < t_{N^{[\ell]}}^{[\ell]} = b,$$

where  $N^{[\ell]}$  denotes the number of time steps on level  $\ell$ . In practice, the nodes on each level are obtained dynamically by the step-size controller.

**2.1.1. The predictor.** To generate a provisional solution, a low-order integrator is applied to solve the IVP (1). For example, a first-order forward Euler integrator applied to (1) gives

$$\eta_n^{[0]} = \eta_{n-1}^{[0]} + (t_n^{[0]} - t_{n-1}^{[0]})f(t_{n-1}^{[0]}, \eta_{n-1}^{[0]}), \quad (2)$$

for  $n = 1, 2, \dots, N^{[0]}$ , with  $\eta_0^{[0]} = y_a$ , and where we have indexed the prediction level as level 0. We denote  $\eta^{[\ell]}(t)$  as a continuous extension [15] of the numerical solution at level  $\ell$ , i.e., a piecewise polynomial  $\eta^{[0]}(t)$  that satisfies

$$\eta^{[0]}(t_n^{[0]}) = \eta_n^{[0]}.$$

The continuous extension of a numerical solution is often of the same order of accuracy as the underlying discrete solution [15]. Indeed, for the purposes of this study, we assume  $\eta^{[\ell]}(t)$  is of the same order as  $\eta_n^{[\ell]}$ .

**2.1.2. The correctors.** Suppose an approximate solution  $\eta(t)$  to IVP (1) is computed. Denote the exact solution by  $y(t)$ . Then, the error of the approximate solution is  $e(t) = y(t) - \eta(t)$ . If we define the defect as  $\delta(t) = f(t, \eta(t)) - \eta'(t)$ , then

$$e'(t) = y'(t) - \eta'(t) = f(t, \eta(t) + e(t)) - f(t, \eta(t)) + \delta(t).$$

The error equation can be written in the form

$$\left[ e(t) - \int_a^t \delta(\tau) d\tau \right]' = f(t, \eta(t) + e(t)) - f(t, \eta(t)), \quad (3)$$

subject to the initial condition  $e(a) = 0$ . In RIDC, the corrector at level  $\ell$  solves for the error  $e^{[\ell-1]}(t)$  of the solution  $\eta^{[\ell-1]}(t)$  at the previous level to generate the corrected solution  $\eta^{[\ell]}(t)$ ,

$$\eta^{[\ell]}(t) = \eta^{[\ell-1]}(t) + e^{[\ell-1]}(t).$$

For example, a corrector at level  $\ell$  that corrects  $\eta^{[\ell-1]}(t)$  by applying a first-order forward Euler integrator to the error equation (3) takes the form

$$e^{[\ell-1]}(t_n^{[\ell]}) - e^{[\ell-1]}(t_{n-1}^{[\ell]}) - \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} \delta^{[\ell-1]}(\tau) d\tau = \Delta t_n^{[\ell]} [f(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]}) + e^{[\ell-1]}(t_{n-1}^{[\ell]})) - f(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]}))],$$

where  $\Delta t_n^{[\ell]} = t_n^{[\ell]} - t_{n-1}^{[\ell]}$ . After some algebraic manipulation, one obtains

$$\eta_n^{[\ell]} = \eta_{n-1}^{[\ell]} + \Delta t_n^{[\ell]} [f(t_{n-1}^{[\ell]}, \eta^{[\ell]}(t_{n-1}^{[\ell]})) - f(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]}))] + \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} f(\tau, \eta^{[\ell-1]}(\tau)) d\tau. \quad (4)$$

The integral in (4) is approximated using quadrature,

$$\int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} f(\tau, \eta^{[\ell-1]}(\tau)) d\tau \approx \sum_{i=1}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \alpha_{n,i}^{[\ell-1]} f(\tau_i, \eta^{[\ell-1]}(\tau_i)), \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}, \quad (5)$$

where the set of quadrature nodes,  $\vec{\mathcal{T}}_n^{[\ell]}$ , for a first-order corrector satisfies

1.  $|\vec{\mathcal{T}}_n^{[\ell]}| = \ell + 1$ ,
2.  $\vec{\mathcal{T}}_n^{[\ell]} \subseteq \{t_n^{[\ell-1]}\}_{n=0}^{N^{[\ell-1]}}$ ,
3.  $\min(\vec{\mathcal{T}}_n^{[\ell]}) \leq t_{n-1}^{[\ell]}$ ,
4.  $\max(\vec{\mathcal{T}}_n^{[\ell]}) \geq t_n^{[\ell]}$ .

The quadrature weights,  $\alpha_{n,i}^{[\ell-1]}$ , are found by integrating the interpolating Lagrange polynomials exactly,

$$\alpha_{n,i}^{[\ell-1]} = \prod_{j=1, j \neq i}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \int_{t_{n-1}^{[\ell]}}^{t_n^{[\ell]}} \frac{(t - \tau_j)}{(\tau_i - \tau_j)} dt, \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}. \quad (6)$$

The term  $f(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]}))$  in (4) is approximated using Lagrange interpolation,

$$f(t_{n-1}^{[\ell]}, \eta^{[\ell-1]}(t_{n-1}^{[\ell]})) \approx \sum_{i=1}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \gamma_{n,i}^{[\ell-1]} f(\tau_i, \eta^{[\ell-1]}(\tau_i)), \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}, \quad (7)$$

where the same set of nodes,  $\vec{\mathcal{T}}_n^{[\ell]}$ , for the quadrature is used for the interpolation. The interpolation weights are given by

$$\gamma_{n,i}^{[\ell-1]} = \prod_{j=1, j \neq i}^{|\vec{\mathcal{T}}_n^{[\ell]}|} \frac{(t_{n-1}^{[\ell]} - \tau_j)}{(\tau_i - \tau_j)}, \quad \tau_i \in \vec{\mathcal{T}}_n^{[\ell]}. \quad (8)$$

**2.2. Adaptive step-size control.** Adaptive step-size control is typically used to achieve a user-specified error tolerance with minimal computational effort by varying the step-sizes used by an IVP integrator. This is commonly done based on a local error estimate. It is also generally desirable that the step-size vary smoothly over the course of the integration. We review common techniques for estimating the local error, followed by algorithms for optimal step-size selection.

**2.2.1. Error estimators.** Two common approaches for estimating the local truncation error of a single-step IVP solver are through the use of Richardson extrapolation (commonly used within a step-size selection framework known as step doubling) and embedded Runge–Kutta pairs [15]. Step doubling is perhaps the more intuitive technique. The solution after each step is estimated twice: once as a full step and

once as two half steps. The difference between the two numerical estimates gives an estimate of the truncation error. For example, denoting the exact solution to IVP (1) at time  $t_n + \Delta t$  as  $y(t_n + \Delta t)$ , the forward Euler step starting from the exact solution at time  $t_n$  and using a step-size of size  $\Delta t$  is

$$\eta_{1,n+1} = y(t_n) + \Delta t f(t_n, y_n),$$

and the forward Euler step using two steps of size  $\Delta t/2$  is

$$\eta_{2,n+1} = \left( y(t_n) + \frac{\Delta t}{2} f(t_n, y_n) \right) + \frac{\Delta t}{2} f\left( t_n + \frac{\Delta t}{2}, y(t_n) + \frac{\Delta t}{2} f(t_n, y_n) \right).$$

Because forward Euler is a first-order method (and thus has a local truncation error of  $\mathcal{O}(\Delta t^2)$ ), the two numerical approximations satisfy

$$\begin{aligned} y(t_n + \Delta t) &= \eta_{1,n+1} + (\Delta t)^2 \phi + \mathcal{O}(\Delta t^3) + \dots, \\ y(t_n + \Delta t) &= \eta_{2,n+1} + 2 \left( \frac{\Delta t}{2} \right)^2 \phi + \mathcal{O}(\Delta t^3) + \dots, \end{aligned}$$

where a Taylor series expansion gives that  $\phi$  is a constant proportional to  $y''(t_n)$ . The difference between the two numerical approximations gives an estimate for the local truncation error of  $\eta_{2,n+1}$ ,

$$e_{n+1} = \eta_{2,n+1} - \eta_{1,n+1} = \frac{\Delta t^2}{2} \phi + \mathcal{O}(\Delta t^3).$$

An alternative approach to estimating the local truncation error is to use embedded RK pairs [11]. An  $s$ -stage Runge–Kutta method is a single-step method that takes the form

$$\eta_{n+1} = \eta_n + \Delta t \sum_{i=1}^s b_i k_i,$$

where

$$k_i = f\left( t_i + c_i h, \eta_n + \Delta t \sum_{j=1}^s a_{ij} k_j \right), \quad i = 1, 2, \dots, s.$$

The idea is to find two single-step RK methods, typically one with order  $p$  and the other with order  $p - 1$ , that share most (if not all) of their stages but have different quadrature weights. This is represented compactly in the extended Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b \\ & \hat{b} \end{array}$$

Denoting the solution from the order- $p$  method as

$$\eta_{n+1}^* = \eta_n + \Delta t \sum_{i=1}^s \hat{b}_i k_i, \quad (9a)$$

and the solution from the order- $(p-1)$  method as

$$\eta_{n+1} = \eta_n + \Delta t \sum_{i=1}^s b_i k_i, \quad (9b)$$

the error estimate is

$$e_{n+1} = \eta_{n+1} - \eta_{n+1}^* = \Delta t \sum_{i=1}^s (b_i - \hat{b}_i) k_i, \quad (9c)$$

which is  $\mathcal{O}(\Delta t^p)$ .

A third approach for approximating the local truncation error is possible within the deferred correction framework. We observe that in solving the error equation (3), one is in fact obtaining an approximation to the error. As discussed in Section 3.3, it can be shown that the approximate error after  $\ell$  first-order corrections satisfies  $\mathcal{O}(\Delta t^{p_0+\ell+1})$ . We shall see in Section 3.3 that this error estimate proves to be a poor choice for optimal step-size selection because in our formulation the time step selection for level  $\ell$  does not allow for the refinement of time steps at earlier levels.

**2.2.2. Optimal step-size selection.** Given an error estimate from Section 2.2.1 for a step  $\Delta t$ , one would like to either accept or reject the step based on the error estimate and then estimate an optimal step-size for the next time step or retry the current step. Following [16], Algorithm 1 outlines optimal step-size selection given an estimate of the local truncation error. In lines 1–4, one computes a scaled error estimate. In line 5, an optimal time step is computed by scaling the current time step. In lines 6–10, a new time step is suggested; a more conservative step-size is suggested if the previous step was rejected.

### 3. RIDC with adaptive step-size control

There are numerous adaptive step-size control strategies that can be implemented within the RIDC framework. We consider three of them in this paper as well as discuss other strategies that are possible.

**3.1. Adaptive step-size control: prediction level only.** One simple approach to step-size control with RIDC is to perform adaptive step-size control on the prediction level only, e.g., using step doubling or embedded RK pairs as error estimators for the step-size control strategy. The subsequent correctors then use this grid unchanged (i.e., without performing further step-size control). With this strategy, corrector  $\ell$  is



**Input:**

$y_n$ : approximate solution at time  $t_n$ ;  
 $y_{n+1}$ : approximate solution at time  $t_{n+1}$ ;  
 $e_{n+1}$ : error estimate for  $y_{n+1}$ ;  
 $p$ : order of integrator;  
 $m$ : number of ODEs;  
 $atol, rtol$ : user specified tolerances;  
 $prev\_rej$ : flag that indicates whether the previous step was rejected;  
 $\alpha < 1$ : safety factor;  
 $\beta > 1$ : allowable change in step-size.

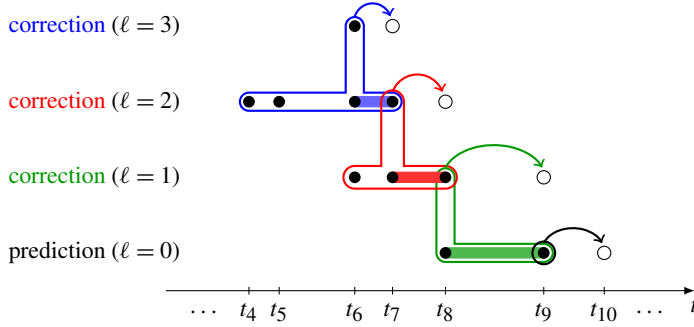
**Output:**

$accept\_flag$ : flag to accept or reject this step;  
 $\Delta t_{new}$ : optimal time step

- 1 Set  $a(i) = \max\{|y_n(i)|, |y_{n+1}(i)|\}$ ,  $i = 1, 2, \dots, m$ .
- 2 Compute  $\tau(i) = atol + rtol * a(i)$ ,  $i = 1, 2, \dots, m$ .
- 3 Compute  $\epsilon = \sqrt{\frac{\sum_{i=1}^m (e(i)/\tau(i))^2}{m}}$ .
- 4 Compute  $\Delta t_{opt} = \Delta t (\frac{1}{\epsilon})^{1/(p+1)}$ .
- 5 **if**  $prev\_rej$  **then**
- 6 |  $\Delta t_{new} = \alpha \min\{\Delta t, \max\{\Delta t_{opt}, \Delta t/\beta\}\}$
- 7 **else**
- 8 |  $\Delta t_{new} = \alpha \min\{\beta \Delta t, \max\{\Delta t_{opt}, \Delta t/\beta\}\}$
- 9 **end**
- 10 **if**  $\epsilon > 1$  **then**
- 11 |  $accept\_flag = 1$
- 12 **else**
- 13 |  $accept\_flag = 0$
- 14 **end**

**Algorithm 1:** Optimal step-size selection algorithm. The approximate solution, the error estimate, and its order are provided as inputs. For the numerical experiments in Section 4, we fix  $\alpha = 0.9$ ,  $\beta = 10$ .

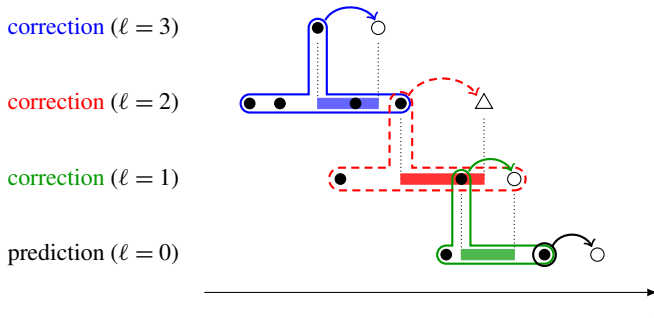
lagged behind corrector  $\ell - 1$  so that each node simultaneously computes an update on its level (after an initial startup period). This is illustrated graphically in Figure 1. In principle, near optimal parallel speedup is maintained with this approach provided the computational overhead for the RIDC method (i.e., the interpolation, quadrature, and linear combination of solutions) is small compared to the advance of predictor



**Figure 1.** Schematic diagram of step-size control on the prediction level only. The filled circles denote previously computed and stored solution values at particular times. The corrections are run in parallel (but lagging in time) and the open circles indicate which values are being simultaneously computed. The stencil of points required by each level is shown by the “bubbles” surrounding certain grid points; the thick horizontal shading indicates the integrals needed in (4).

from  $t_n$  to  $t_{n+1}$ ; in this implementation, a small memory footprint similar to [10] can be used. Additionally, an interpolation step is circumvented because the nodes are the same on each level. There are however a few potential drawbacks to this approach. First, it is not clear how to distribute the user-defined tolerance among the levels. Clearly, satisfying the user-specified tolerance on the prediction level defeats the purpose of the deferred correction approach. Estimating a reduced tolerance criterion may be possible a priori, but such an estimate would at present be ad hoc. Second, there is no reason to expect the corrector (4) should take the same steps to satisfy an error tolerance when computing a numerical approximation to the error equation (3).

**3.2. Adaptive step-size control: all levels.** A generalization of the above formulation is to utilize adaptive step-size control to solve the error equations (3) as well. The variant we consider is step doubling on all levels, where each predictor and corrector performs Algorithm 1; embedded RK pairs can also be used to estimate the error for step-size adaptivity on all levels. Intuitively, step-size control on every level gives more opportunity to detect and adapt to error than simply adapting using the (lowest-order) predictor. For example, this allows the corrector take a smaller step if necessary to satisfy an error tolerance when solving the error equation. Some drawbacks are: (i) an interpolation step is necessary because the nodes are generally no longer in the same locations on each level, (ii) more memory registers are required, and (iii) there is a potential loss of parallel efficiency because a corrector may be stalled waiting for an adequate stencil to become available to compute a quadrature approximation to the integral in (4). Another issue — both a potential benefit and a potential drawback — is the number of parameters that can be tuned



**Figure 2.** Schematic diagram of a scenario when step-size control is applied on all levels. Unlike in Figure 1, here each level has its own grid in time. Solid circles indicate particular times and levels where the solution is known. In this particular diagram, levels  $\ell = 0, 1, 3$  are all able to advance simultaneously to the open circles. However, correction level  $\ell = 2$  is unable to advance to the time indicated by the triangle symbol because correction level  $\ell = 1$  has not yet computed far enough. The stencil of points required by each level is shown by the “bubbles” surrounding certain grid points; the thick horizontal shading indicates the integrals needed in (4). Note in particular that the dashed stencil includes a open circle at level  $\ell = 1$  that is not yet computed.

for each problem. A discussion on the effect of tolerance choices for each level is provided in Section 4. One can in practice also tune step-size control parameters  $\alpha$ ,  $\beta$ ,  $\text{atol}$ , and  $\text{rtol}$  for Algorithm 1 separately on each level. Figure 2 highlights that some nodes might not be able to compute an updated solution on their current level if an adequate stencil is not available to approximate the integral in (4) using quadrature. In this example, the level  $\ell = 2$  correction is unable to proceed because it would require interpolated solution values not yet available from level  $\ell = 1$ , whereas the prediction level  $\ell = 0$  and corrections  $\ell = 1$  and  $\ell = 3$  are all able to advance the solution by one step.

**3.3. Adaptive step-size control: using the error equation.** A third strategy one might consider is adaptive step-size control for the error equation (3) using the solution to the error equation itself as the error estimate. (One still uses step doubling or embedded RK pairs to obtain an error estimate for step-size control on the predictor equation (1).) At first glance, this looks promising provided the order of the integrator can be established because it is used to determine an optimal step-size. One would expect computational savings from utilizing available error information, as opposed to estimating it via step doubling or an embedded RK pair.

If first-order predictor and first-order correctors are used to construct the RIDC method, the analysis in [17] can be easily extended to the proposed RIDC methods with adaptive step-size control. We note that the numerical quadrature approximation given in (5) and the numerical interpolation given in (7) are accurate to the order  $\mathcal{O}(\Delta t_n^{\ell+2})$ ; this is sufficient for the inductive proof in [17] to hold. Hence, one

can show that the method has a formal order of accuracy  $\mathcal{O}(\Delta t^{\ell+2})$ , where  $\Delta t = \max_{n,\ell}(t_n^{[\ell]} - t_{n-1}^{[\ell]})$ .

Although the formal order of accuracy can be established, using the error estimate from successive levels is a poor choice for optimal step-size selection. Consider step-size selection for level  $\ell$ , time step  $t_n^{[\ell]}$ , using  $\eta_n^{[\ell]} - \eta_n^{[\ell-1]}(t_n^{[\ell]})$  as the error estimator in Algorithm 1. The optimal step-size is chosen to control the local error estimate via the step-size  $\Delta t_n^{[\ell]} = t_n^{[\ell]} - t_{n-1}^{[\ell]}$ . However, the local error for the correctors generally contains contributions from the solutions at all the previous levels. The validity of the asymptotic local error expansion of the RIDC method in terms of  $\Delta t_n^{[\ell]}$  requires that  $\Delta t = \max_{n,\ell}(t_n^{[\ell]} - t_{n-1}^{[\ell]})$  be sufficiently small, and it is not normally possible to guarantee this in the context of an IVP solver. In other words, the step-size controller for a corrector at a given level cannot control the entire local error, and hence standard step-control strategies, which are predicated on the validity of error expansions in terms of only the step-size to be taken, cannot be expected to perform well. We present some numerical tests in Section 4.2.4 to illustrate the difficulties with using successive errors as the basis for step-size control.

**3.4. Further discussion.** There are many other strategies/implementation choices that affect the overall performance of the adaptive RIDC algorithm. Some have already been discussed in the previous section. We summarize some of the implementation choices that must be made:

- The choice of how to estimate the error of the discretization must be made. Three possibilities have already been mentioned: step doubling, embedded RK pairs, and solutions to the error equation (3). A combination of all three is also possible.
- If an IVP method with adaptive step-size control is used to solve (3), choices must be made as to how the tolerances and step-size control parameters,  $\alpha$  and  $\beta$ , are to be chosen for each correction level.

We also list a few implementation details that should be considered when designing adaptive RIDC schemes.

- If adaptive step-size control is implemented on all levels, some correction levels may sit idle because the information required to perform the quadrature and interpolation in (4) is not available. This idle time adversely affects the parallel efficiency of the algorithm. One possibility to decrease this idle time is instead of taking an “optimal step” (as suggested by the step-size control routine), one could take a smaller step for which the quadrature and interpolation stencil is available. There is some flexibility in choosing exactly which points are used in the quadrature stencil, and it might also be possible to choose a stencil to minimize the time that correction levels are sitting idle.

- Because values are needed from lower-order correction levels, the storage required by a RIDC scheme depends on when values can be overwritten (see, e.g., the stencils in Figures 1 and 2). Thus to avoid increasing the storage requirements, the prediction level and each correction level should not be allowed to get too far ahead of higher correction levels. Although this is also the case for the nonadaptive RIDC schemes [10; 6], if adaptive step-size control is implemented on all levels (Figure 2), the memory footprint is likely to increase. Some consideration should thus be given to a potential trade-off between parallel efficiency and the overall memory footprint of the scheme.
- It is important to reduce round-off error when computing the quadrature weights (6) and the interpolation weights (8). This can be done by through careful scaling and control of the order of the floating-point operations [3].
- If one wishes to use higher-order correctors and predictors to construct RIDC integrators, we note that the convergence analysis in [7; 8; 5] only holds for uniform steps. A nonuniform mesh introduces discrete “roughness” (see [8]); hence, an increase of only one order per correction level is guaranteed even though a high-order method is used to solve (3).
- RIDC methods necessarily incur computational overhead costs, for example, quadrature evaluation (5), interpolation evaluation (7), and the combination of these components in (4). Parallel speedup can only be expected if the computational overhead is small compared to the advance of predictor from  $t_n$  to  $t_{n+1}$ .

Additionally, the RIDC framework, by construction, solves a series of error equations to generate a successively more accurate solution. This framework can be potentially be exploited to generate *order-adaptive* RIDC methods. For example, one might control the number of corrector levels adaptively based on an error estimate.

#### 4. Numerical examples

We focus on the solutions to three nonlinear IVPs. The first is presented in [1]; we refer to it as the Auzinger IVP:

$$\begin{cases} y_1' = -y_2 + y_1(1 - y_1^2 - y_2^2), \\ y_2' = y_1 + 3y_2(1 - y_1^2 - y_2^2), \\ y(0) = (1, 0)^T, \quad t \in [0, 10], \end{cases} \quad (\text{AUZ})$$

that has the analytic solution  $y(t) = (\cos t, \sin t)^T$ .

The second is the IVP associated with the Lorenz attractor:

$$\begin{cases} y_1' = \sigma(y_2 - y_1), \\ y_2' = \rho y_1 - y_2 - y_1 y_3, \\ y_3' = y_1 y_2 - \beta y_3, \\ y(0) = (1, 1, 1)^T, \quad t \in [0, 1]. \end{cases} \quad (\text{LORENZ})$$

For the parameter settings  $\sigma = 10$ ,  $\rho = 28$ ,  $\beta = 8/3$ , this system is highly sensitive to perturbations, and an IVP integrator with adaptive step-size control may be advantageous.

The third is the restricted three-body problem from [15]; we refer to it as the Orbit IVP:

$$\begin{cases} y_1'' = y_1 + 2y_2' - \mu' \frac{y_1 + \mu}{D_1} - \mu \frac{y_1 - \mu'}{D_2}, \\ y_2'' = y_2 - 2y_1' - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}, \\ D_1 = ((y_1 + \mu)^2 + y_2^2)^{3/2}, \quad D_2 = ((y_1 - \mu')^2 + y_2^2)^{3/2}, \\ \mu = 0.012277471, \quad \mu' = 1 - \mu. \end{cases} \quad (\text{ORBIT})$$

Choosing the initial conditions

$$\begin{aligned} y_1(0) &= 0.994, & y_1'(0) &= 0, & y_2(0) &= 0, \\ y_2'(0) &= -2.00158510637908252240537862224, \end{aligned}$$

gives a periodic solution with period  $t_{\text{end}} = 17.065216560159625588917206249$ .

We now present numerical evidence to demonstrate that:

1. RIDC integrators with nonuniform step-sizes converge and achieve their designed orders of accuracy.
2. RIDC methods with adaptive step-size constructed using step doubling (on the prediction level only) and embedded RK error estimators (on the prediction level only) converge.
3. RIDC methods with adaptive step-size control based on step doubling to estimate the local error on the prediction and correction levels converge; however, the step-sizes selected are poor (many rejected steps), even for the smooth Auzinger problem.
4. RIDC methods with adaptive step-size control based on step doubling to estimate the local error on the prediction level but using the solution to the error equation for step-size control results is problematic.

The numerical examples chosen are canonical problems designed to illustrate the step-size adaptivity properties of the RIDC methods. Because the computational

overhead is significant compared to the advance of an Euler solution from time  $t_n$  to  $t_{n+1}$ , a runtime analysis does not reveal parallel speedup for any of these examples. Whereas the number of function evaluations is an effective parameter for comparing algorithms, we need a different metric to compare a parallel algorithm to a sequential algorithm. Where appropriate, we tabulate the number of *sets of concurrent function evaluations* as a proxy for measuring parallel speedup when the function evaluation costs dominate. A set of concurrent function evaluations consists of function evaluations that can be evaluated in parallel.

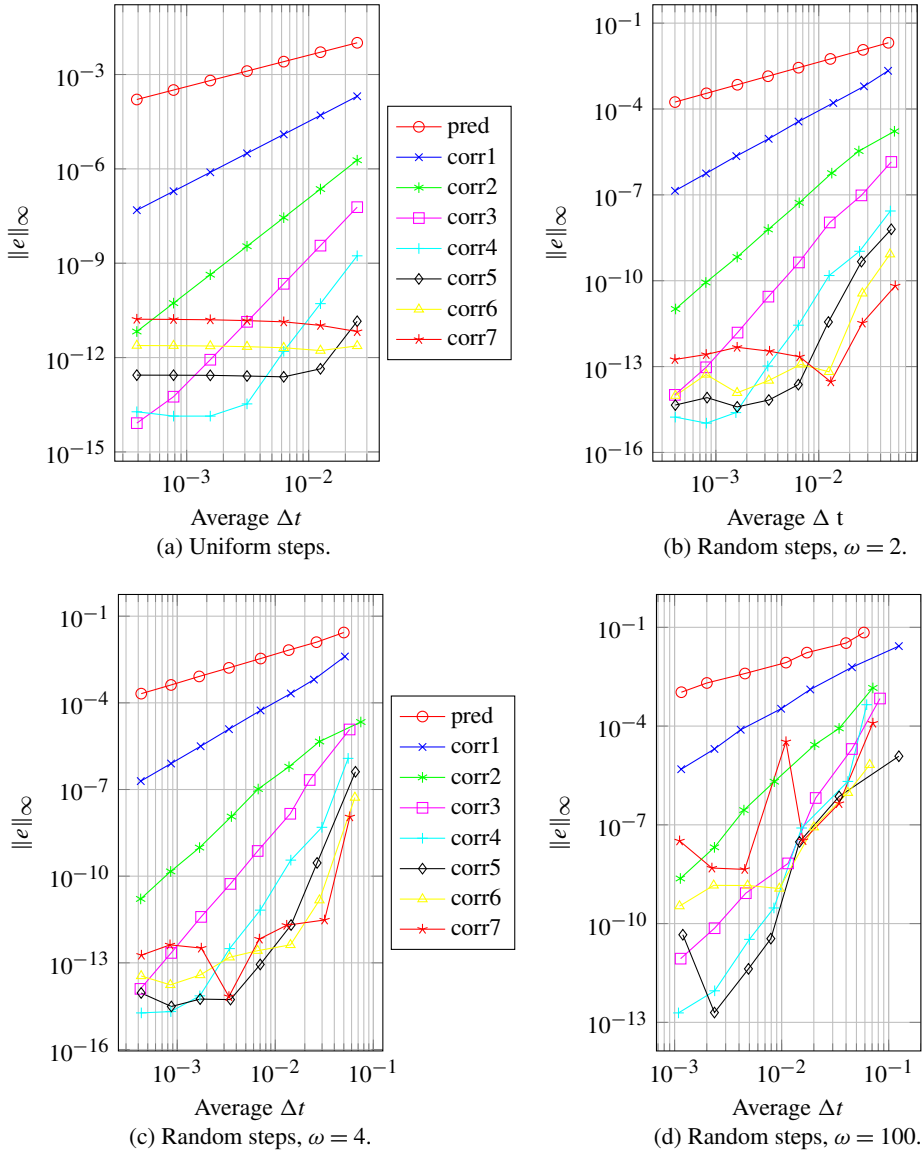
**4.1. RIDC with nonuniform step-sizes.** For our first numerical experiment, we demonstrate that RIDC integrators with nonuniform step-sizes converge and achieve their design orders of accuracy. Figure 3 shows the classical convergence study (error as a function of mean step-size) for the RIDC integrator applied to (AUZ). Figure 3(a) shows the convergence of RIDC integrators with uniform step-sizes; Figure 3(b)–(d) shows the convergence of RIDC integrators when random step-sizes are chosen. The random step-sizes are chosen so that

$$\Delta t_n^{[\ell]} \in \left[ \frac{1}{\omega} \Delta t_{n-1}^{[\ell]}, \omega \Delta t_{n-1}^{[\ell]} \right], \quad \omega \in \mathbb{R},$$

where  $\omega$  controls how rapidly a step-size is allowed to change. The figures show that RIDC integrators with nonuniform step-sizes achieve their designed order of accuracy (each additional correction improves the order of accuracy by one), at least up to order 6. In Figure 3 (corresponding to RIDC with uniform step-sizes), we observe that the error stagnates at a value significantly larger than machine precision. This is likely due to numerical issues associated with quadrature on equispaced nodes [14]. We note that  $\omega = 1$  gives the uniformly distributed case. We also observe that as the ratio of the largest to the smallest cell increases, the performance of higher-order RIDC methods degrades, likely due to round-off error associated with calculating the quadrature and interpolation weights.

Figure 4 shows the convergence study (error as a function of mean step-size) for (LORENZ). The reference solution is computed using an RK-45 integrator with a fine time step. Similar observations can be made that RIDC methods with nonuniform step-sizes converge with their designed orders of accuracy (at least up to order 6).

**4.2. Adaptive RIDC.** We study four different variants of RIDC methods with adaptive step-size control: (i) step doubling is used for adaptive step-size control on the prediction level only (Section 4.2.1); (ii) an embedded RK pair is used for adaptive step-size control on the prediction level only (Section 4.2.2); (iii) step doubling is used for adaptive step-size control on the prediction and correction levels (Section 4.2.3); and (iv) step doubling is used for adaptive step-size control

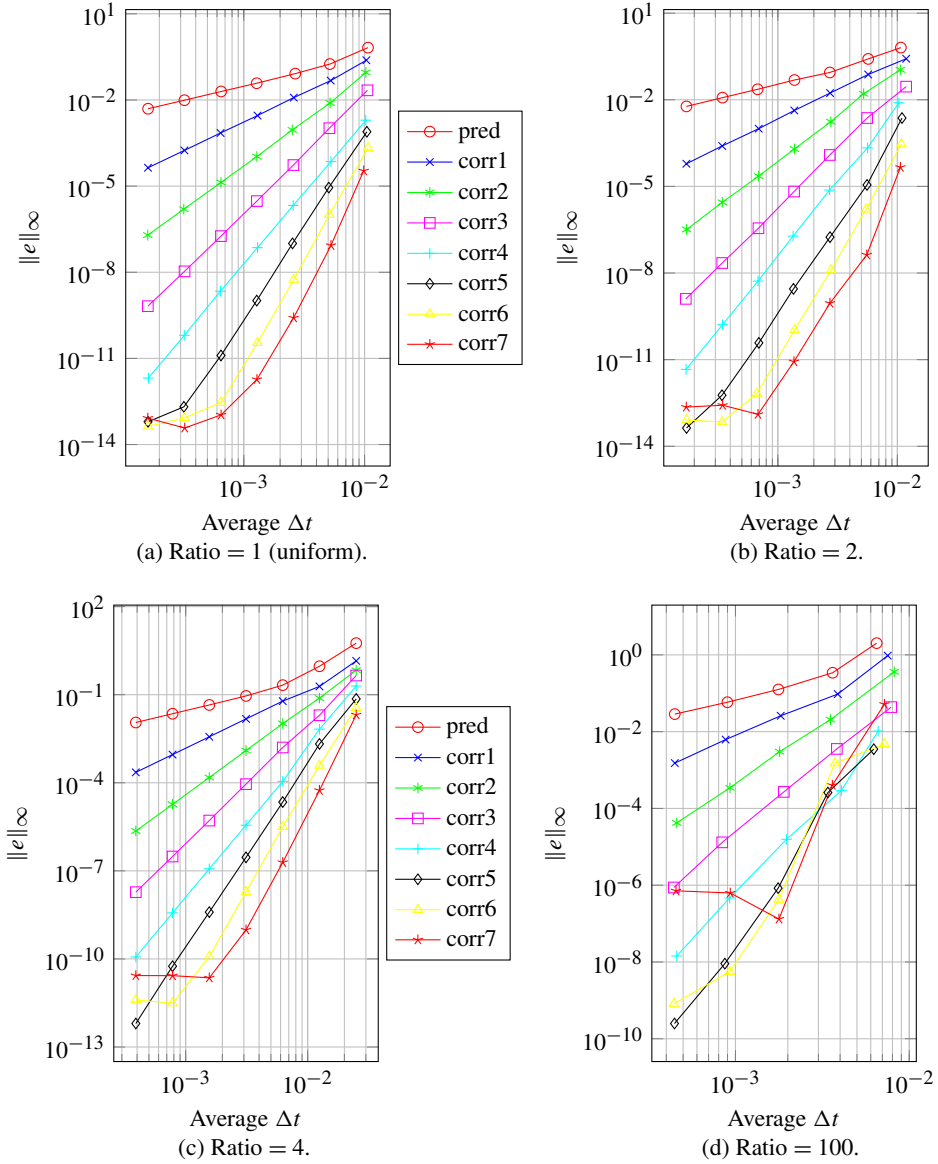


**Figure 3.** Auzinger IVP: The design order is illustrated for the RIDC methods.

on the prediction level, and the computed errors from the error equation (3) are used for adaptive step-size control on the correction levels.

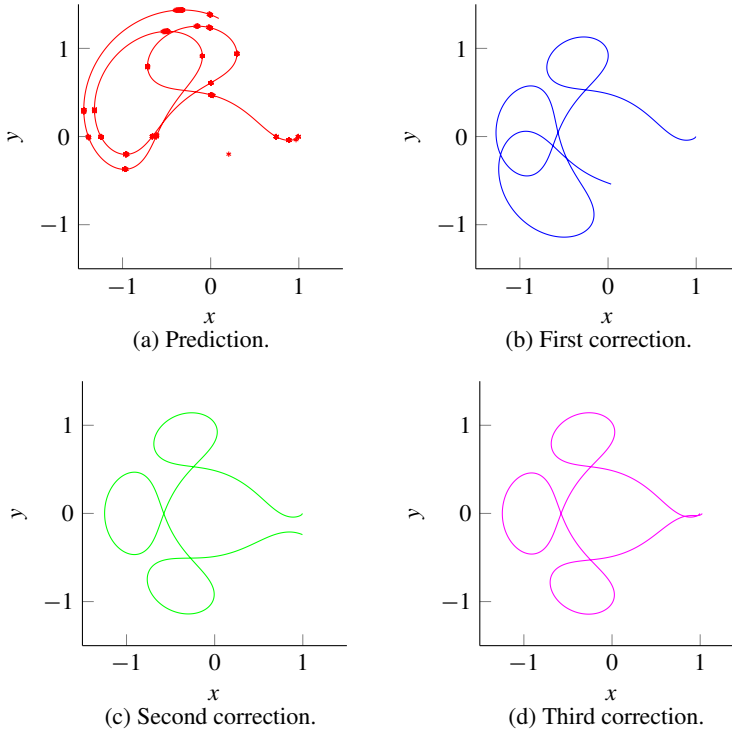
**4.2.1. Step doubling on the prediction level only.** In this numerical experiment, we solve the orbit problem (ORBIT) using a fourth-order RIDC method (constructed using forward Euler integrators), and adaptive step-size control on the prediction level only, where step doubling is used to provide the error estimate. As shown in





**Figure 4.** Lorenz IVP: the design order is illustrated for the RIDC methods.

Figure 5, successive correction loops are able to reduce the error in the solution and recover the desired orbit. The red circles in Figure 5(a) indicate rejected steps. Figure 6(a) shows that RIDC with step doubling only on the prediction level converges as the tolerance is reduced. In this experiment, the RIDC integrator is *reset* after every 100 accepted steps. By “reset” [10], we mean that the highest-order solution after every 100 steps is used as an initial condition to reinitialize the



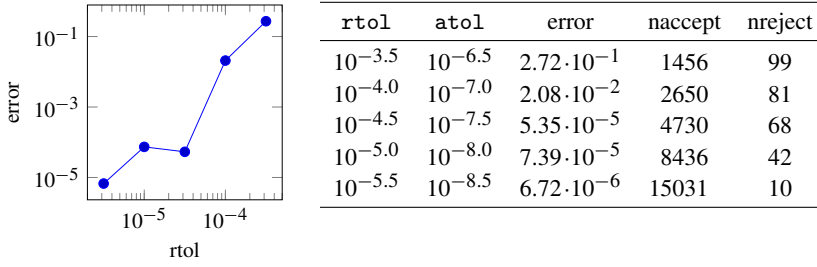
**Figure 5.** Orbit problem: although the prediction level gives a highly inaccurate solution, successive correction loops are able to reduce the error and produce the desired orbit. The red circles on the prediction level (a) indicate rejected steps.

provisional solution; e.g., instead of solving (1), one solves a sequence of problems

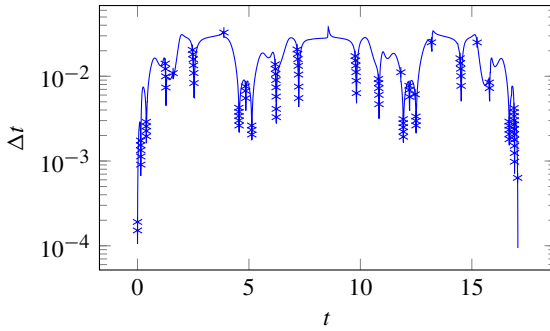
$$\begin{cases} y'(t) = f(t, y), & t \in [t_{100(i-1)}, \min(b, t_{100i})], \\ y(t_{100(i-1)}) = \eta_{100(i-1)}^{[P-1]}, \end{cases}$$

if  $(L-1)$  correctors are applied and  $\eta_0^{[L-1]} = y_a$ . The time steps chosen by the RIDC integrator with resets performed every 100 and 400 steps are shown in Figure 6(b) and (c).

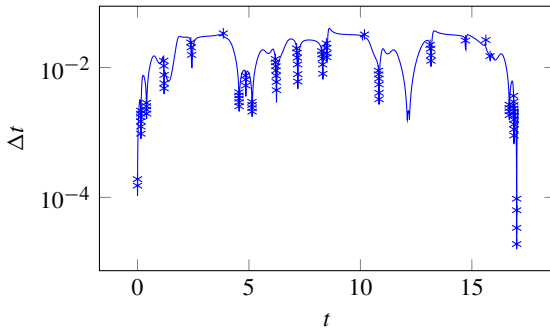
In Figure 6(b),  $\Delta t_{\min} = 1.06 \times 10^{-4}$ . If a nonadaptive fourth-order RIDC method was used with  $\Delta t_{\min}$ , 160814 uniform time steps would have been required. By adaptively selecting the time steps for this example and tolerance, the adaptive RIDC method required approximately one one-hundredth of the functional evaluations, corresponding to a one hundred-fold speedup. The effective parallel speedup can be computed by taking the ratio of the total number of function evaluations required and the number of sets of concurrent function evaluations required. For the computation in Figure 6(b) where a reset is performed after every 100 steps, the parallel speedup



(a) Convergence study.



(b) Adaptive step-sizes selected (reset every 100 steps).



(c) Adaptive step-sizes selected (reset every 400 steps).

**Figure 6.** Orbit problem: (a) convergence of a fourth-order RIDC method constructed with forward Euler integrators and adaptive step-size control on the prediction level (using step doubling). Convergence is measured relative to the exact solution as the tolerance is decreased. A reset is performed after every 100 accepted steps for this convergence study. In (b), the step-sizes selected for  $rtol = 10^{-3.5}$  and  $atol = 10^{-6.5}$  are displayed as the solid curve and rejected steps as  $\times$ ; a reset is performed after every 100 steps. In (c), the reset is performed after every 400 steps. Observe that although the number of rejected steps increases, the overall  $\Delta t$  chosen remains qualitatively similar.

(if four processors are available) can be computed using

$$\frac{(1456 \times 5) + 99}{(1456 \times 2) + (14 \times 6) + 99} = 2.38.$$

The numerator consists of the total number of function evaluations arising from the number of steps taken and the computation of the error estimate using step doubling and the number of function evaluations arising from the rejected steps. The denominator consists of the number of concurrent function evaluations (including startup costs for the RIDC method). Note that three of the processors sit idle while that step doubling computation is being processed. The parallel speedup can be improved if more levels are chosen, or if the number of resets are reduced. If a reset is performed after every 400 steps (Figure 6(c)), the parallel speedup is

$$\frac{(1591 \times 5) + 88}{(1591 \times 2) + (4 \times 6) + 88} = 2.44.$$

**4.2.2. Embedded RK on the prediction level only.** In this numerical experiment, we repeat the orbit problem (ORBIT) using a fourth-order RIDC method constructed again using forward Euler integrators, but the step-size adaptivity on the prediction level uses a Heun–Euler embedded RK pair. This simple scheme combines Heun’s method, which is second order, with the forward Euler method, which is first order. Figure 7(a) shows the convergence of this adaptive RIDC method as the tolerance is reduced. As the previous example, the RIDC integrator is reset after every 100 accepted steps for the convergence study. In Figure 7(b) and (c), we show the time steps chosen by the RIDC integrator with resets performed after 100 or 400 steps, respectively.

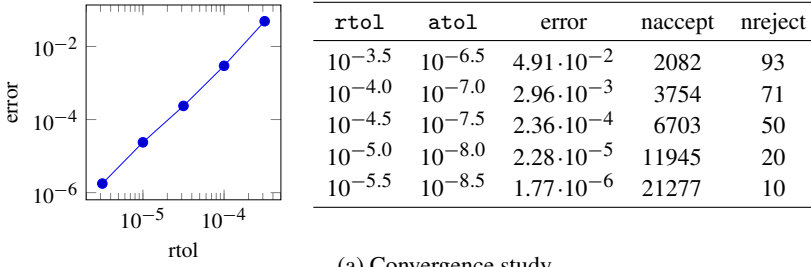
For the computation in Figure 7(b) where a reset is performed after every 100 steps, the parallel speedup (if four processors are available) is

$$\frac{(2441 \times 5) + 60}{(2441 \times 2) + (24 \times 6) + 60} = 2.41.$$

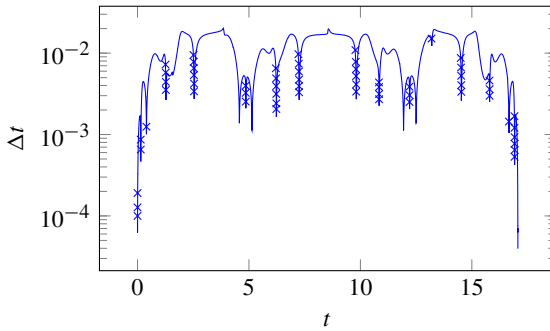
If a reset is performed after every 400 steps (Figure 7(c)), the parallel speedup is

$$\frac{(2276 \times 5) + 80}{(2276 \times 2) + (5 \times 6) + 80} = 2.46.$$

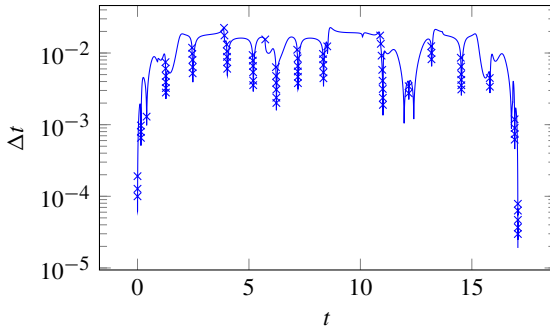
Not surprisingly, the time steps chosen by the RIDC method are dependent on the specified tolerances and the error estimator (and consequently the integrators used to obtain a provisional solution to (1)) used for the control strategy. One can easily construct a RIDC integrator using higher-order embedded RK pairs to solve for a provisional solution to (1), and then use the forward Euler method to solve the error equation (3) on subsequent levels. For example, Figure 8 shows the step-sizes chosen when the Bogacki–Shampine method [2] (a 3(2) embedded RK pair) and the popular Runge–Kutta–Fehlberg 4(5) pair [13] are used to compute the provisional solution (and error estimate) for the RIDC integrator. The same



(a) Convergence study.



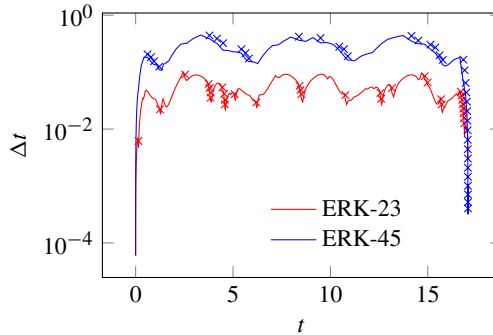
(b) Adaptive step-sizes selected (reset every 100 steps).



(c) Adaptive step-sizes selected (reset every 400 steps).

**Figure 7.** Orbit problem: (a) convergence of a fourth-order RIDC method constructed with forward Euler integrators and adaptive step-size control on the prediction level (using an embedded RK pair to estimate the error). Convergence is measured relative to the exact solution as the tolerance is decreased. A reset is performed after every 100 accepted steps for this convergence study. In (b), the step-sizes selected for  $rtol = 10^{-3.5}$  and  $atol = 10^{-6.5}$  are displayed as the solid curve and rejected steps as  $\times$ s; a reset is performed after every 100 steps. In (c), the reset is performed after every 400 steps.

tolerance of  $rtol = 10^{-3.5}$  is used to generate both graphs. As the order and accuracy of the predictor increases, one can take larger time steps. For this example, using higher-order embedded RK pairs as step-size control mechanisms for RIDC methods result in less variations in time steps.

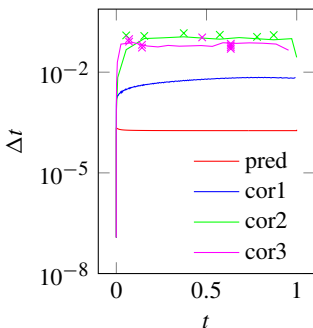


**Figure 8.** Step-sizes selected by RIDC methods constructed using a Bogacki–Shampine method, a 3(2) embedded pair (red) and the Runge–Kutta–Fehlberg 4(5) pair. Rejected steps are indicated with  $\times$ s.

**4.2.3. Step doubling on all levels.** As mentioned in Section 3.2, it might be advantageous to use adaptive step-size control when solving the error equations. This affords a myriad of parameters that can be used to tune the step-size control mechanism. In this set of numerical experiments, we explore how the choice of tolerances for the prediction/correction levels affect the step-size selection.

We first solve the Auzinger IVP using step doubling on all the levels, i.e., both predictor and corrector levels. In Figure 9, we show the computed step-sizes when we naively choose the same tolerances on each level. As expected, the predictor has to take many steps (to satisfy the stringent user-supplied tolerance), whereas life is easy for the correctors. The effective parallel speedup is

$$\frac{(5479 + 196 + 19 + 24) \times 2 + 15}{(5481 \times 2) + 15} = 1.04.$$



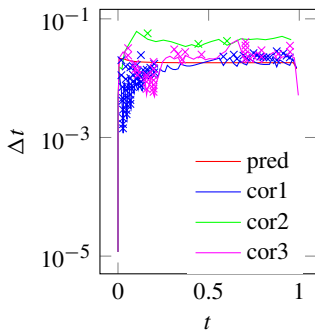
$\ell$	rtol	atol	error	naccept	nreject
0	$10^{-8}$	$10^{-10}$	$2.028 \cdot 10^{-5}$	5479	0
1	$10^{-8}$	$10^{-10}$	$8.793 \cdot 10^{-7}$	196	0
2	$10^{-8}$	$10^{-10}$	$2.618 \cdot 10^{-8}$	19	6
3	$10^{-8}$	$10^{-10}$	$1.486 \cdot 10^{-6}$	24	9

**Figure 9.** Auzinger IVP: step-size control is implemented on all prediction and correction levels. The same tolerances are used for each level. As expected, the predictor has a hard time (forward Euler must satisfy a stringent tolerance); on the other hand, life is easy for the correctors. Rejected steps are indicated with  $\times$ s. For this set of tolerances, 5481 sets of concurrent function evaluations are needed.

In principle, the correctors are not even needed. Equally important to note is that the error *increases* after the last correction loop. This might seem surprising at first glance but ultimately may not unreasonable because the steps selected to solve the third correction are not based on the solution to the error equation but rather the original IVP.

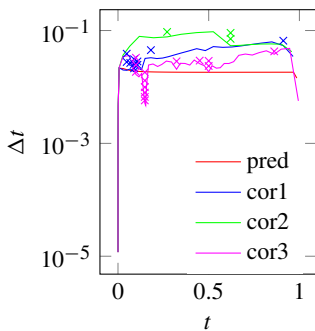
Instead of naively choosing the same tolerances on each level, we now change the tolerance at each level, as described in Figure 10. By making this simple change, the number of accepted steps on each level are now on the same order of magnitude. Not surprisingly, the predictor still selects good steps. Interestingly in Figure 10(a), the first correction is “noisy”, especially initially. For this set of tolerances, the effective parallel speedup is

$$\frac{(58 + 7 + 30 + 61) \times 2 + (52 + 7 + 24)}{(135 \times 2) + (52 + 7 + 24)} = 1.52.$$



(a) Set 1 of tolerances.

$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$2.026 \cdot 10^{-3}$	58	0
1	$1 \cdot 10^{-6}$	$1 \cdot 10^{-8}$	$6.945 \cdot 10^{-5}$	78	52
2	$1 \cdot 10^{-8}$	$1 \cdot 10^{-10}$	$1.265 \cdot 10^{-7}$	30	7
3	$1 \cdot 10^{-10}$	$1 \cdot 10^{-12}$	$9.579 \cdot 10^{-8}$	61	24



(b) Set 2 of tolerances.

$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$2.026 \cdot 10^{-3}$	58	0
1	$1 \cdot 10^{-5}$	$1 \cdot 10^{-7}$	$1.805 \cdot 10^{-4}$	29	12
2	$1 \cdot 10^{-7}$	$1 \cdot 10^{-9}$	$1.172 \cdot 10^{-6}$	20	6
3	$1 \cdot 10^{-9}$	$1 \cdot 10^{-11}$	$7.216 \cdot 10^{-7}$	39	11

**Figure 10.** Auzinger IVP: different tolerances at each level. With the first set of tolerances, the step-size controller for the predictor is well behaved, as it is for the second and third correctors. The step-size controller for the first corrector however is noisy. 135 sets of concurrent function evaluations are needed to generate (b). With the second set of tolerances, the step-size controller for all correctors is reasonably well behaved. Here, 64 sets of concurrent function evaluations are needed.

By picking a different set of tolerances, we can eliminate the noise, as shown in Figure 10(b). For this set of tolerances, the parallel speedup is

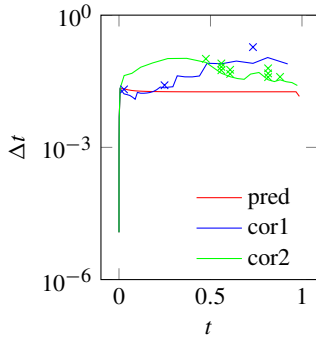
$$\frac{(58 + 24 + 20 + 39) \times 2 + (12 + 6 + 11)}{(64 \times 2) + (12 + 6 + 11)} = 1.98.$$

**4.2.4. Using solutions from the error equation.** As mentioned in Section 3.3, using the solution from the error equation (3) as the local error estimate for step-size control on a given level is potentially problematic because the step-size controller can only control the local error introduced on that level whereas the true local error generally contains contributions from all previous levels. For completeness, we present the results of this adaptive RIDC formulation applied to the Orbit problem (Figure 12) and the Auzinger problem (Figure 11). Step doubling is used for step-size adaptivity on the predictor level, solutions from the error equation are used to control step-sizes for the corrector levels. For the Auzinger problem, we observe in the top figure that if the tolerances are held fixed on each level, each correction level improves the solution. If the tolerance is reduced slightly on each level, the step-size controller gives a poor step-size selection (many rejected steps), even for this smoothly varying problem. For the Orbit IVP, Figure 12 shows that the corrector improves the solution if the tolerances are held fixed at all levels; however the corrector requires *many* steps. A second correction loop was not attempted. Reducing the tolerance for the first corrector resulted in inordinately many rejected steps.

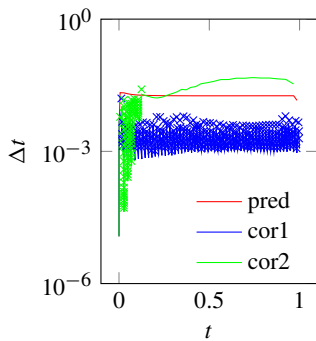
## 5. Conclusions

In this paper, we formulated RIDC methods that incorporate local error estimation and adaptive step-size control. Several formulations were discussed in detail: (i) step doubling on the prediction level, (ii) embedded RK pairs on the prediction level, (iii) step doubling on the prediction and error levels, and (iv) step doubling for the prediction level but using the solution from the error equation for step-size control; other formulations are also alluded to. A convergence theorem from [17] can be extended to RIDC methods that use adaptive step-size control on the prediction level. Numerical experiments demonstrate that RIDC methods with nonuniform steps converge as designed and illustrate the type of behavior that might be observed when adaptive step-size control is used on the prediction and correction levels. Based on our numerical study, we conclude that adaptive step-size control on the prediction level is viable for RIDC methods. In a practical application where a user gives a specified tolerance, this prescribed tolerance must be transformed to a specific tolerance that is fed to the predictor.

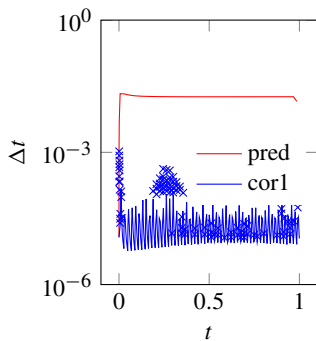




$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$2.031 \cdot 10^{-3}$	59	0
1	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$7.249 \cdot 10^{-4}$	33	3
2	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$6.513 \cdot 10^{-6}$	26	10



$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$2.031 \cdot 10^{-3}$	59	0
1	$1 \cdot 10^{-5}$	$1 \cdot 10^{-7}$	$1.063 \cdot 10^{-5}$	657	305
2	$1 \cdot 10^{-6}$	$1 \cdot 10^{-8}$	$9.446 \cdot 10^{-8}$	75	76

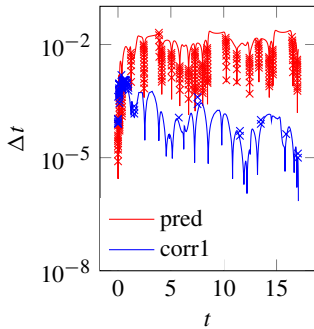


$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-6}$	$2.031 \cdot 10^{-3}$	59	0
1	$1 \cdot 10^{-7}$	$1 \cdot 10^{-9}$	$1.178 \cdot 10^{-7}$	60571	94

**Figure 11.** Auzinger problem: step doubling on prediction level, using successive levels for error estimation for step control on the error equation. Step-size controller for the corrector is noisy.

## Acknowledgments

This publication was based on work supported in part by award no. KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST), AFRL and AFOSR under contract and grants FA9550-12-1-0455, NSF grant number DMS-0934568, NSERC grant number RGPIN-228090-2013, and the Oxford Center for Collaborative and Applied Mathematics (OCCAM).



$\ell$	rtol	atol	error	naccept	nreject
0	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	2.405	2261	230
1	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$7.234 \cdot 10^{-1}$	475181	84

**Figure 12.** Orbit problem: step doubling on prediction level, using successive levels for error estimation for step control on the error equation.

## References

- [1] W. Auzinger, H. Hofstätter, W. Kreuzer, and E. Weinmüller, *Modified defect correction algorithms for ODEs, I: general theory*, Numer. Algorithms **36** (2004), no. 2, 135–155. MR 2005h:65096
- [2] P. Bogacki and L. F. Shampine, *A 3(2) pair of Runge–Kutta formulas*, Appl. Math. Lett. **2** (1989), no. 4, 321–325. MR 1025845 Zbl 0705.65055
- [3] B. Bradie, *A friendly introduction to numerical analysis: with C and MATLAB materials on website*, Pearson Education, Upper Saddle River, NJ, 2006.
- [4] A. Christlieb, A. Melfi, and B. Ong, *Distributed parallel semi-implicit time integrators*, preprint, 2012. arXiv 1209.4297v1
- [5] A. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods*, Commun. Math. Sci. **9** (2011), no. 3, 879–902. MR 2865808 Zbl 1271.65109
- [6] A. Christlieb and B. Ong, *Implicit parallel time integrators*, J. Sci. Comput. **49** (2011), no. 2, 167–179. MR 2012k:65067 Zbl 1243.65076
- [7] A. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **4** (2009), 27–56. MR 2010e:65094 Zbl 1167.65389
- [8] ———, *Integral deferred correction methods constructed with high order Runge–Kutta integrators*, Math. Comp. **79** (2010), no. 270, 761–783. MR 2011c:65122 Zbl 1209.65073
- [9] A. J. Christlieb, R. D. Haynes, and B. W. Ong, *A parallel space-time algorithm*, SIAM J. Sci. Comput. **34** (2012), no. 5, C233–C248. MR 3023735 Zbl 1259.65143
- [10] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835. MR 2011g:65105 Zbl 1211.65089
- [11] J. R. Dormand and P. J. Prince, *A family of embedded Runge–Kutta formulae*, J. Comput. Appl. Math. **6** (1980), no. 1, 19–26. MR 81g:65098 Zbl 0448.65045
- [12] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), no. 2, 241–266. MR 2001e:65104 Zbl 0959.65084
- [13] E. Fehlberg, *Low-order classical Runge–Kutta formulas with step size control and their application to some heat transfer problems*, technical report, R-315, NASA, 1969.

- [14] S. Güttel and G. Klein, *Efficient high-order rational integration and deferred correction with equispaced data*, *Electron. Trans. Numer. Anal.* **41** (2014), 443–464.
- [15] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations, I: Nonstiff problems*, 2nd ed., Springer Series in Computational Mathematics, no. 8, Springer, Berlin, 1993. MR 94c:65005
- [16] E. Hairer and G. Wanner, *Solving ordinary differential equations, II: Stiff and differential-algebraic problems*, 2nd ed., Springer Series in Computational Mathematics, no. 14, Springer, Berlin, 1996. MR 97m:65007 Zbl 0859.65067
- [17] Y. Xia, Y. Xu, and C.-W. Shu, *Efficient time discretization for local discontinuous Galerkin methods*, *Discrete Contin. Dyn. Syst. Ser. B* **8** (2007), no. 3, 677–693. MR 2008e:65307 Zbl 1141.65076

Received October 9, 2013. Revised December 10, 2014.

ANDREW J. CHRISTLIEB: *Department of Mathematics, Michigan State University, East Lansing, 48823, United States*

COLIN B. MACDONALD: [macdonald@maths.ox.ac.uk](mailto:macdonald@maths.ox.ac.uk)  
*Mathematical Institute, Oxford University, Oxford, OX2 6GG, United Kingdom*

BENJAMIN W. ONG: [ongbw@mtu.edu](mailto:ongbw@mtu.edu)  
*Department of Mathematics, Michigan Technological University, Houghton, MI 49931, United States*

RAYMOND J. SPITERI: [spiteri@cs.usask.ca](mailto:spiteri@cs.usask.ca)  
*Department of Computer Science, University of Saskatchewan, Saskatoon S7N 5C9, Canada*



# Communications in Applied Mathematics and Computational Science

msp.org/camcos

## EDITORS

### MANAGING EDITOR

John B. Bell  
Lawrence Berkeley National Laboratory, USA  
jbbell@lbl.gov

### BOARD OF EDITORS

Marsha Berger	New York University berger@cs.nyu.edu	Ahmed Ghoniem	Massachusetts Inst. of Technology, USA ghoniem@mit.edu
Alexandre Chorin	University of California, Berkeley, USA chorin@math.berkeley.edu	Raz Kupferman	The Hebrew University, Israel raz@math.huji.ac.il
Phil Colella	Lawrence Berkeley Nat. Lab., USA pcolella@lbl.gov	Randall J. LeVeque	University of Washington, USA rjl@amath.washington.edu
Peter Constantin	University of Chicago, USA const@cs.uchicago.edu	Mitchell Luskin	University of Minnesota, USA luskin@umn.edu
Maksymilian Dryja	Warsaw University, Poland maksymilian.dryja@acn.waw.pl	Yvon Maday	Université Pierre et Marie Curie, France maday@ann.jussieu.fr
M. Gregory Forest	University of North Carolina, USA forest@amath.unc.edu	James Sethian	University of California, Berkeley, USA sethian@math.berkeley.edu
Leslie Greengard	New York University, USA greengard@cims.nyu.edu	Juan Luis Vázquez	Universidad Autónoma de Madrid, Spain juanluis.vazquez@uam.es
Rupert Klein	Freie Universität Berlin, Germany rupert.klein@pik-potsdam.de	Alfio Quarteroni	Ecole Polytech. Féd. Lausanne, Switzerland alfio.quarteroni@epfl.ch
Nigel Goldenfeld	University of Illinois, USA nigel@uiuc.edu	Eitan Tadmor	University of Maryland, USA etadmor@cscamm.umd.edu
		Denis Talay	INRIA, France denis.talay@inria.fr

## PRODUCTION

production@msp.org

Silvio Levy, Scientific Editor

---

See inside back cover or [msp.org/camcos](http://msp.org/camcos) for submission instructions.

---

The subscription price for 2015 is US \$85/year for the electronic version, and \$120/year (+\$15, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscribers address should be sent to MSP.


---

Communications in Applied Mathematics and Computational Science (ISSN 2157-5452 electronic, 1559-3940 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840, is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

---

CAMCoS peer review and production are managed by EditFLOW® from MSP.

PUBLISHED BY

 **mathematical sciences publishers**  
nonprofit scientific publishing

<http://msp.org/>

© 2015 Mathematical Sciences Publishers

# *Communications in Applied Mathematics and Computational Science*

vol. 10

no. 1

2015

- 
- |  |    |
|--|----|
| Revisionist integral deferred correction with adaptive step-size control   | 1  |
| ANDREW J. CHRISTLIEB, COLIN B. MACDONALD, BENJAMIN W. ONG and RAYMOND J. SPITERI   |    |
| An adaptively weighted Galerkin finite element method for boundary value problems  | 27 |
| YIFEI SUN and CHAD R. WESTPHAL   |    |
| An adaptive finite volume method for the incompressible Navier–Stokes equations in complex geometries                                    | 43 |
| DAVID TREBOTICH and DANIEL T. GRAVES   |    |
| High-accuracy embedded boundary grid generation using the divergence theorem   | 83 |
| PETER SCHWARTZ, JULIE PERCELAY, TERRY J. LIGOCKI, HANS JOHANSEN, DANIEL T. GRAVES, DHARSHI DEVENDRAN, PHILLIP COLELLA and ELI ATELJEVICH |    |



1559-3940(2015)10:1;1-O