

Implications of the Choice of Quadrature Nodes for Picard Integral Deferred Corrections Methods for Ordinary Differential Equations *

Anita T. Layton¹ and Michael L. Minion² †

¹*Department of Mathematics, Duke University, Box 90320, Durham, NC 27708, U.S.A.
email: alayton@math.duke.edu*

²*Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599, U.S.A.
email: minion@amath.unc.edu*

Abstract.

This paper concerns a class of deferred correction methods recently developed for initial value ordinary differential equations; such methods are based on a Picard integral form of the correction equation. These methods divide a given timestep $[t_n, t_{n+1}]$ into substeps, and use function values computed at these substeps to approximate the Picard integral by means of a numerical quadrature. The main purpose of this paper is to present a detailed analysis of the implications of the location of quadrature nodes on the accuracy and stability of the overall method. Comparisons between Gauss-Legendre, Gauss-Lobatto, Gauss-Radau, and uniformly spaced points are presented. Also, for a given set of quadrature nodes, quadrature rules may be formulated that include or exclude function values computed at the left-hand endpoint t_n . Quadrature rules that do not depend on the left-hand endpoint are shown to lead to $L(\alpha)$ -stable implicit methods with $\alpha \approx \pi/2$. The semi-implicit analog of this property is also discussed. Numerical results suggest that the use of uniform quadrature nodes, as opposed to nodes based on Gaussian quadratures, does not significantly affect the stability or accuracy of these methods for orders less than ten. In contrast, a study of the reduction of order for stiff equations shows that the form and extent of order-reduction changes considerably with the use of uniform quadrature nodes.

AMS subject classification (2000): 65B05.

Key words: semi-implicit methods, deferred correction methods, order reduction, Picard integral

1 Introduction

In the last several years, a series of papers have appeared introducing numerical methods for the solution of ordinary differential equations (ODEs) based on a new variant of deferred corrections [10, 23, 25, 5, 21, 24]. These methods have been shown to have attractive stability and accuracy properties even for versions

*Received xxxx 2004. Revised xxxx 200x. Communicated by xxxx xxxx.

†This work was supported in part under contract DE-AC03-76SF00098 by the Director, Department of Energy (DOE) Office of Science; Office of Advanced Scientific Computing Research; Office of Mathematics, Information, and Computational Sciences; Applied Mathematics Sciences Program; and by the Alexander von Humboldt Foundation.

with very high order of accuracy. In general, the deferred correction strategy for producing higher-order approximations is to first compute a provisional solution at a set of points in the integration interval and then to use the provisional solution to form an equation for the error or correction in the provisional solution.

The deferred correction strategy presented in [10] differs from previous approaches in that the equation for the correction is not strictly an ODE, but instead is based on an approximation to the Picard integral solution to the initial value problem

$$(1.1) \quad \phi'(t) = F(t, \phi)$$

$$(1.2) \quad \phi(t_0) = \phi_0$$

given by

$$(1.3) \quad \phi(t) = \phi_0 + \int_{t_0}^t F(\tau, \phi(\tau)) d\tau.$$

An advantage to this approach is that the values of the provisional solution are used to approximate this integral rather than to approximate derivatives of the provisional solution. This allows the construction of methods with very high order of accuracy while maintaining the stability properties of lower order methods.

In these methods, a given timestep is divided into intermediate substeps, and function values computed at these substeps are used to approximate the Picard integral in Eq. (1.3) by means of a numerical quadrature. In [10], the choice of the size of the intermediate substeps in the deferred correction method are chosen so that the quadrature nodes used to approximate the Picard integral correspond to the standard Gauss-Legendre quadrature nodes. It is for this reason that the methods are referred to as *spectral* deferred correction (SDC) methods. This moniker is somewhat misleading because, although the quadrature rule over the entire interval corresponds to a spectral integration rule, the intermediate quadratures over subintervals necessary in the SDC method do not. Consequently, the maximum order of temporal accuracy of a method using p Gaussian nodes is $p + 1$ and not $2p + 1$ as one might expect.

One area in which SDC methods have shown promise is for the numerical solution of ODEs which can be separated into stiff and non-stiff terms. A prominent example of such ODEs, which are the main motivation of this work, results from the spatial discretization of a partial differential equation (PDE) with multiple spatial terms (e.g. advection-diffusion-reaction equations). SDC methods can be extended in a reasonably straightforward manner to treat non-stiff terms explicitly and stiff terms implicitly. Semi-implicit versions of SDC for equations with one stiff and one non-stiff term are introduced in [23, 25], and multi-implicit methods for ODEs with multiple stiff components in [5]. In these works, Gauss-Lobatto quadrature nodes are used which, unlike Gauss-Legendre nodes, include the endpoints of the interval. This choice avoids the need to extrapolate the numerical solution from internal nodes to the right endpoint of the temporal

domain as is done in [10]. Numerical and analytical analysis in these papers show that this choice allows one to construct semi- and multi-implicit methods with very high order of accuracy whose stability properties do not degrade as the order is increased.

However, there are situations when it is either convenient or necessary to choose substeps that do not correspond to Gaussian quadrature nodes. For example, the function $F(t, \phi)$ in Eq. (1.1) may depend on data that is known only at a discrete set of points in time. This data may come for example from observation or from the output of a wholly independent numerical process. Another possibility is that the ODE method is used for the temporal integration of a PDE, and the spatial approximation procedure may ultimately be more convenient or efficient when uniform time steps are used. It is straightforward to implement deferred correction methods based on the Picard integral for an arbitrary choice of substep size, and this choice will likely effect the stability and accuracy properties of the method. When the choice of substeps is, for example, uniformly spaced, the resulting numerical method cannot be considered “spectral” in any reasonable sense of the word. Hence the more general term *Picard integral* deferred corrections (PIDC) will be used in the remainder of this work.

This paper presents a detailed analysis of the ramifications of the choice of the quadrature nodes for PIDC methods of both implicit and semi-implicit type. Comparisons of methods based on Gauss-Legendre, Gauss-Lobatto, Gauss-Radau, and uniform quadrature nodes are presented, with particular attention paid to semi-implicit examples. After a review of PIDC methods in Sect. 2, the stability characteristics of implicit and semi-implicit versions of PIDC methods are examined through standard stability diagrams in Sect. 3. The advantage of using “right-hand” quadrature rules for stiff problems, i.e. quadrature rules that do not include the function values at the left-hand endpoint in the integration rule, is illustrated. In particular, $L(\alpha)$ -stability for implicit and semi-implicit methods with right-hand quadrature rules are established. In Sect. 4, accuracy diagrams of the PIDC schemes are considered. The absence of any significant loss of accuracy for methods with moderately high order (e.g., four through ten) using uniformly spaced nodes is demonstrated. The efficiency in terms of accuracy per implicit function evaluation of methods with different orders is also discussed. The numerical results for a variety of choices of nodes presented in Sect. 5 further illustrate the relative accuracy of different choices.

Perhaps the most surprising results in this paper concern the extent of order reduction for stiff problems. In [24], numerical results demonstrating order reduction for semi-implicit SDC methods is demonstrated for both stiff linear systems and the stiff van der Pol equation. The latter study was inspired by the results in [19] which demonstrate order reduction for (semi-implicit) Additive Runge-Kutta methods (ARK). In both of these papers the extent of order reduction for the van der Pol equation is not clear, and results here suggest that the ambiguity is due to the use of non-equilibrium initial conditions in the numerical tests. In Sect. 5.2, numerical results for a simple scalar example and the van der Pol equation illustrate the extent of order reduction for PIDC and

ARK methods. It is shown that the extent and character of order reduction depends critically on the choice of quadrature nodes for PIDC methods. This dependence is supported by an explicit derivation of the truncation error for a simple stiff equation.

2 Picard Integral Deferred Corrections Methods

In this section, a short review of PIDC methods for ODEs is presented for completeness; for more details see [10, 23, 5]. For a more direct comparison with classical defect and deferred correction schemes see [25].

The initial value ODE takes the form

$$(2.1) \quad \begin{aligned} \phi'(t) &= F(t, \phi(t)) \\ \phi(t_0) &= \phi_0. \end{aligned}$$

Here, the solution $\phi(t)$ and initial value ϕ_0 are in \mathbb{C}^N and $F : \mathbb{R} \times \mathbb{C}^N \rightarrow \mathbb{C}^N$. It is assumed that F is smooth so that the discussion of higher order methods is appropriate. In practice, one is interested in approximating the solution on a finite time interval which is further subdivided into timesteps $[t_n, t_{n+1}]$.

In general terms, deferred or defect correction methods proceed as follows. First, a lower order provisional solution $\phi^0(t) \approx \phi(t)$ with error $\delta(t) = \phi(t) - \phi^0(t)$ is computed at a number of intermediate points in the integration interval with a standard numerical method. Next, an equation for the error $\delta(t)$ or the updated solution $\phi^0(t) + \delta(t)$ is constructed using the provisional solution. Finally, a numerical approximation to this equation is computed at the intermediate points and used to improve the provisional solution. This procedure can be repeated to form increasingly more accurate approximations $\phi^k(t)$.

Deferred correction methods for initial value ODEs are first introduced by Daniel, Pereyra, and Schumaker in [9]. These methods use finite differences approximations of derivatives of $\phi^0(t)$ to approximate the higher-order derivatives of $\phi(t)$ appearing in the truncation error of the method. These approximations are used to construct a modified ODE which is then solved numerically to yield an updated solution. (See also [26, 16, 15, 20].) Deferred correction methods are also closely related to iterated defect correction methods introduced earlier by Zadunaisky [28]. Iterated defect correction methods use the derivative of a polynomial interpolant of $\phi^0(t)$ to form an ODE to which the same numerical method used to compute $\phi^0(t)$ is applied. The solution to this second ODE yields an approximation to the correction $\delta(t)$. (See also [14, 4].)

Although similar in spirit to both deferred and defect correction methods, PIDC methods do not rely on numerical approximations to the derivative of the provisional solution. This is accomplished by utilizing the Picard integral equation solution to Eq. (2.1) given by

$$(2.2) \quad \phi(t) = \phi_0 + \int_{t_0}^t F(\tau, \phi(\tau)) d\tau$$

to construct a correction equation. The provisional solution $\phi^0(t)$ does not satisfy such an equation, however the quantity

$$(2.3) \quad E(t, \phi^0) = \phi_0 + \int_{t_0}^t F(\tau, \phi^0(\tau)) d\tau - \phi^0(t)$$

provides a measure of the error, and can be computed accurately and efficiently by numerical quadrature. Since $\phi(t) = \phi^0(t) + \delta(t)$, direct substitution yields

$$(2.4) \quad \delta(t) = \int_{t_0}^t F(\tau, \phi^0(\tau) + \delta(\tau)) - F(\tau, \phi^0(\tau)) d\tau + E(t, \phi^0).$$

Note that unlike classical defect or deferred corrections methods, this equation is not in the form of an ODE.

The specific strategy of PIDC methods is to first sub-divide the interval $[t_n, t_{n+1}]$ by choosing p points $t_m \in [t_n, t_{n+1}]$, so that $t_n = t_0 < t_1 \dots < t_p \leq t_{n+1}$. These points define p subintervals $[t_m, t_{m+1}]$ for $m = 0 \dots p - 1$. Note that t_p may or may not correspond to the right endpoint t_{n+1} . Next, a standard numerical method is used to compute a provisional solution $\phi^0(t_m)$. Finally, a simple procedure is used to approximate Eq. (2.4) and iteratively improve the provisional solution such that the solution at the $(k + 1)$ st iteration is given by $\phi^{k+1}(t_m) = \phi^k(t_m) + \delta^k(t_m)$. For example, the implicit methods in [10] employ the standard backward Euler method for computing $\phi^0(t_m)$, and a similar first-order procedure for approximating $\delta^k(t_m)$. Specifically in the latter case, letting subscripts correspond to time (i.e. $\phi_m^k = \phi^k(t_m)$), the procedure is given by

$$(2.5) \quad \begin{aligned} \delta_{m+1}^k &= \delta_m^k + \Delta t_m [F(t_{m+1}, \phi_{m+1}^k + \delta_{m+1}^k) - F(t_{m+1}, \phi_{m+1}^k)] \\ &\quad + E_{m+1}(\phi^k) - E_m(\phi^k), \end{aligned}$$

where $\Delta t_m = t_{m+1} - t_m$. When F is Lipschitz continuous, each deferred correction iteration using Eq. (2.5) increases the order of accuracy of the solution by one, so long as the numerical quadrature terms in $E_m(\phi^k)$ are sufficiently accurate. (See [25] for further discussion.) Hence for each of the examples in this paper, a K th-order PIDC method implies Eq. (2.5) is solved $K - 1$ times.

Consider now an ODE which can be separated into stiff and non-stiff parts

$$(2.6) \quad \begin{aligned} \phi'(t) &= F(t, \phi(t)) = F_E(t, \phi(t)) + F_I(t, \phi(t)) \\ \phi(t_0) &= \phi_0, \end{aligned}$$

where the subscripts E and I correspond to the desire to treat the first term explicitly and the second implicitly. The PIDC strategy can be extended to this case by using a forward-backward Euler method for the provisional solution and correction equations [23]. For the provisional solution, this is

$$(2.7) \quad \phi_{m+1}^0 = \phi_m^0 + \Delta t_m [F_E(t_m, \phi_m^0) - F_I(t_{m+1}, \phi_{m+1}^0)],$$

while for the correction equation, this is

$$(2.8) \quad \delta_{m+1}^k = \delta_m^k + \Delta t_m [F_E(t_m, \phi_m^k + \delta_m^k) - F_E(t_m, \phi_m^k)]$$

$$\begin{aligned}
& + F_I(t_{m+1}, \phi_{m+1}^k + \delta_{m+1}^k) - F_I(t_{m+1}, \phi_{m+1}^k)] \\
& + E_{m+1}(\phi^k) - E_m(\phi^k).
\end{aligned}$$

Multi-implicit methods for equations with two or more different stiff terms are similarly constructed by using an operator splitting approach for both provisional solution and update equation. (See [5] for more details.)

Since the quantity $E(t, \phi^k)$ must be estimated from the provisional solution at the points t_m , the choice of t_m defines the quadrature nodes for this integration. When $E(t_{n+1}, \phi^k)$ is being computed, i.e. the integral over the entire interval $[t_n, t_{n+1}]$, choosing Gauss-Legendre quadrature nodes provides the approximation with the highest possible order of accuracy. Specifically, if t_m for $m = 1 \dots p$, are the Gauss-Legendre nodes, then $E(t_{n+1}, \phi^k)$ can be computed with order $2p + 1$ accuracy. However, using these same nodes to compute $E(t_m, \phi^k)$ (i.e. the integral over the subinterval $[t_n, t_m]$) yields only order $p + 1$ accuracy for $m \in [1, p]$. It is for this reason that using a different choice of nodes does not change the overall order of accuracy of the PIDC method.

This fact can be further illustrated by rearranging terms in the Eq. (2.5) to yield a direct equation for $\phi^{k+1} = \phi^k + \delta^k$. Note that

$$(2.9) \quad E(t_{m+1}, \phi^k) - E(t_m, \phi^k) = \int_{t_m}^{t_{m+1}} F(\tau, \phi^k(\tau)) d\tau - \phi_{m+1}^k + \phi_m^k.$$

Let $I_m^{m+1}(\phi^k)$ denote the numerical quadrature approximation of

$$(2.10) \quad \int_{t_m}^{t_{m+1}} F(\tau, \phi^k(\tau)) d\tau.$$

Then, Eq. (2.5) can be rewritten

$$(2.11) \quad \phi_{m+1}^{k+1} = \phi_m^{k+1} + \Delta t_m [F(t_{m+1}, \phi_{m+1}^{k+1}) - F(t_{m+1}, \phi_{m+1}^k)] + I_m^{m+1}(\phi^k).$$

Note that as $\delta^k(t_m)$ goes to zero, the difference term in this equation also goes to zero (assuming F is Lipschitz), so that the update for ϕ^{k+1} is increasingly determined by $I_m^{m+1}(\phi^k)$.

Since the quadrature must be done for each subinterval $[t_m, t_{m+1}]$ for $m = 0 \dots p - 1$ there are actually p quadrature rules

$$(2.12) \quad I_m^{m+1}(\phi^k) = \Delta t_m \sum_{l=0}^p q_m^l F(t_l, \phi_l^k).$$

Hence it is the accuracy of the quadrature on the subintervals $[t_m, t_{m+1}]$, rather than that of $[t_n, t_{n+1}]$, which determines the overall order of accuracy of the method. Note that the coefficients q_m^l , can be precomputed, and the quadrature is simply a matrix-vector multiplication.

As mentioned before, the points t_m are chosen in [10] to be the standard Gaussian quadrature nodes, and the solution is only computed at these nodes on the interior of the interval. This choice has several consequences. First,

the matrix-vector multiply required for the quadrature rule can be accelerated using the fast Fourier transform. For methods with very high-order, this is a substantial savings in computational cost. However, for versions with moderate order of accuracy, or when the ODE method is being used to integrate a PDE, this cost saving is often in negligible relative terms. Second, since the solution is never computed at the right hand endpoint t_{n+1} , this value is extrapolated from interior points. Lastly, the solution at the left endpoint of the interval t_n is not used in the quadrature rule in Eq. (2.12) as well.

Since using Gauss-Legendre quadrature nodes is only one of many possible choices, this paper presents an analysis of implications of using different quadrature nodes. Four different choices of quadrature nodes will be considered: Gauss-Legendre, Gauss-Lobatto, right-hand Gauss-Radau, and uniform nodes. Furthermore, given one of these choices of nodes, one could formulate two different quadrature rules depending on whether or not the left endpoint of the timestep is used in the approximation. For instance one could add the left-most endpoint to the Gauss-Legendre or Gauss-Radau nodes to formulate a new quadrature rule, or likewise one could omit the left-most endpoint from the Gauss-Lobatto or uniform nodes. The term *right-hand* will refer to any quadrature rule that does not depend on the left-most endpoint in the interval. This implies that $q_m^l = 0$ for all m in Eq. (2.12). The Gauss-Legendre and right-hand Gauss-Radau rules are both right-hand rules by this convention. The right-hand rule for Gauss-Lobatto or uniform nodes is obtained by simply integrating the polynomial which interpolates the function at all but the left-most nodes. It will be shown in the next section that right-hand rules lead to $L(\alpha)$ -stable implicit methods for each choice of nodes. A discussion of further advantages of right-hand rules is included in Sect. 6.

For semi-implicit methods, it is also possible to use a right-hand quadrature rule for the implicit term but include the left-most value in the quadrature rule for the explicit term. In this case, the quadrature rules in Eq. (2.12) become

$$(2.13) \quad I_m^{m+1}(\phi^k) = \Delta t_m \sum_{l=0}^p q_m^l F_E(t_l, \phi_l^k) + \tilde{q}_m^l F_I(t_l, \phi_l^k).$$

For Gauss-Legendre and right-hand Radau nodes, this would amount to adding the left endpoint as an additional node in the integration rules for the explicit piece. The rationale for such a choice is that it preserves the $L(\alpha)$ -stability properties of right-hand methods while increasing the order of accuracy of the quadrature of the explicit terms. Such methods are also considered in the following sections.

For notational simplicity, quadrature rules which use the left endpoint for both the explicit and implicit piece will be denoted *LL* rules. Likewise, those using the left endpoint only for the explicit piece will be referred to as *LR* rules and those which do not use the left endpoint for either piece as *RR* rules. Note that for each example presented here, the order of the quadrature rule used in the method is the same as the total number of solution iterations and hence the overall order of accuracy of the method. Therefore, methods using LR and RR

quadrature rules require an additional node for the implicit quadrature rule (and hence an additional substep) to maintain the same order of accuracy as for LL rules. Therefore methods using LL rules have a lower computational cost than those with LR and RR rules for a given order. However, the numerical tests comparing each of these three possibilities demonstrate that this reduction in cost is offset by a reduction in accuracy.

3 Linear Stability Analysis

Since the seminal paper of Dahlquist [8], the theory of the stability for numerical methods for ODEs has substantially developed. In this section, some stability properties of implicit and semi-implicit PIDC methods will be considered analytically and numerically. In particular, $L(\alpha)$ -stability for right-hand implicit PIDC methods will be established. The possible analogs of $L(\alpha)$ -stability for semi-implicit methods will also be discussed.

Consider the model problem

$$(3.1) \quad y' = \lambda y$$

$$(3.2) \quad y(0) = 1,$$

where $\lambda \in \mathbb{C}$. The exact solution is simply $y(t) = e^{\lambda t}$. Let $y_n \approx y(t_n)$ denote the numerical solution to the model problem computed with a PIDC method. Suppose y_{n+1} is obtained by advancing y_n by one time step. Then

$$(3.3) \quad y_{n+1} = \rho(\lambda \Delta t_n) y_n,$$

where $\Delta t_n = t_{n+1} - t_n$, and $\rho(\lambda \Delta t_n)$ is called the amplification factor.

For PIDC methods that are based on one-step methods such as the Euler method (or any other single step method), the stability region of the method can be defined as the set $\lambda \Delta t \in \mathbb{C}$ for which $|\rho(\lambda \Delta t)| \leq 1$. This is equivalent to defining the region in \mathbb{C} for which $|y_1(\lambda)| \leq 1$ where $y_1(\lambda)$ denotes the numerical solution after one time step with $\Delta t = 1$. For notational simplicity then, the transformation $\lambda = \lambda \Delta t$ is employed so that the amplification factor is simply $\rho(\lambda)$.

A method is defined to be A -stable if the stability region contains the entire left half of \mathbb{C} [8]. A method is defined to be $A(\alpha)$ -stable if the stability region contains the region defined by $e^{i\theta}$ with $\theta \in [\pi - \alpha, \pi + \alpha]$ [27]. Therefore A -stability is equivalent to $A(\alpha)$ -stability with $\alpha = \pi/2$.

Another important concept in stability for stiff equations is that of L -stability [12]. A method is defined to be L -stable if it is A -stable and

$$(3.4) \quad \lim_{\Re(\lambda) \rightarrow -\infty} \rho(\lambda) = 0.$$

Likewise, a method is $L(\alpha)$ -stable if it is $A(\alpha)$ -stable and the above property holds. It is often more illuminating to report values of α in terms of degrees, especially when α is very close to $\pi/2$, and this practice will be followed here.

For implicit PIDC methods based on backward Euler, the following theorem holds.¹ In this and the following theorems, it is assumed that the PIDC method (and hence the quadratures rules therein) are $\mathcal{O}(\Delta t^k)$ accurate for some $k > 1$.

THEOREM 3.1. *Let $\rho(\lambda)$ be the amplification factor determined by an implicit PIDC method based on backward Euler for which the quadrature rule does not include the left-hand endpoint t_n . Then*

$$(3.5) \quad \lim_{|\lambda| \rightarrow \infty} \rho(\lambda) = 0.$$

The proof is based on mathematical induction. Note that the provisional solution $\phi^0(t_m)$ for $m > 1$ is given by

$$(3.6) \quad \phi_m^0 = \prod_{l=1}^m \frac{1}{1 - \lambda \Delta t_l}$$

where Δt_l denotes the size of the sub-steps with $\Delta t = 1$, and hence Δt_l is $\mathcal{O}(1)$. Therefore each ϕ_m^0 is $\mathcal{O}(1/\lambda)$ as $|\lambda| \rightarrow \infty$. Now assume that $\phi^k(t_m)$ is $\mathcal{O}(1/\lambda)$ for $m > 1$. Using the direct update equation (2.11), the integration term is

$$(3.7) \quad I_m^{m+1}(\phi^k) = \Delta t_m \sum_{l=1}^p q_m^l \lambda \phi_l^k,$$

which is $\mathcal{O}(1)$ as $|\lambda| \rightarrow \infty$. For the model problem, Eq. (2.11) becomes

$$(3.8) \quad \phi_{m+1}^{k+1} = \frac{\phi_m^{k+1} - \lambda \Delta t_m \phi_{m+1}^k + I_m^{m+1}(\phi^k)}{1 - \lambda \Delta t_m}.$$

Therefore if ϕ^k is $\mathcal{O}(1/\lambda)$, then the terms in the denominator are $\mathcal{O}(1)$, and ϕ^{k+1} is $\mathcal{O}(1/\lambda)$. Note that if the left-hand value $\lambda \phi_0^k = \lambda$ is used in the quadrature in Eq. (3.7), then the integration terms become $\mathcal{O}(\lambda)$ and the proof fails.

The immediate consequence of this theorem is that any implicit PIDC of this type which is $A(\alpha)$ -stable is also $L(\alpha)$ -stable for the same α . Together with Thm. 3.1, stability regions for implicit PIDC methods computed in Sect. 3.2 establish the $L(\alpha)$ -stability of right-hand implicit methods for α very close to $\pi i/2$.

Note that the limit in Eq. (3.5) in this theorem is somewhat more general than the limit in Eq. (3.4). For implicit PIDC methods (and the majority of single-step numerical methods for ODEs), $\rho(\lambda)$ is a rational function of λ , and hence the limit in Eq. (3.4) does not depend on how λ approaches infinity. For semi-implicit methods considered below, the manner in which λ approaches infinity is important.

¹The authors thank Prof. J. Huang for pointing out this fact. Essentially the same proof appears in [18].

3.1 Semi-implicit linear stability

To extend the standard linear stability analysis above to semi-implicit methods, one must decide how to generalize the model problem in a semi-implicit fashion. The most general possibility is to let

$$(3.9) \quad y' = \lambda_E y + \lambda_I y,$$

where $\lambda_E, \lambda_I \in \mathbb{C}$ define the explicit and implicit functions respectively. With this choice, the stability region can be defined as the region in \mathbb{C}^2 for which the amplification factor $|\rho(\lambda_E, \lambda_I)| \leq 1$. This has the disadvantage that it is inconvenient to graphically describe such regions, and it is not clear how to extend concepts such as $L(\alpha)$ -stability. Furthermore, it seems logical to consider the fact that in general λ_E is assumed to be small in magnitude (i.e. non-stiff), while λ_I is allowed to have large negative real part (i.e. stiff).

In response to these difficulties, the authors in [13] investigate stability regions based only on λ_I by defining the stability region as the set of λ_I for which the scheme is stable whenever λ_E lies in the stability region of the explicit scheme (i.e. the scheme that results when $\lambda_I = 0$). If S_E denotes the stability region of the explicit scheme, then the amplification factor is

$$(3.10) \quad \rho(\lambda_I) = \max_{\lambda_E \in S_E} |\rho(\lambda_E, \lambda_I)|,$$

and $A(\alpha)$ - and $L(\alpha)$ -stability are defined in terms of $\rho(\lambda_I)$ as above.

The following semi-implicit generalization of Thm. 3.1 proves that any PIDC method based on forward-backward Euler methods, which is $A(\alpha)$ -stable by the above definition, is also $L(\alpha)$ -stable.

THEOREM 3.2. *Let $\rho(\lambda_E, \lambda_I)$ be the amplification factor corresponding to Eq. (3.9) determined by a semi-implicit PIDC method based on forward-backward Euler for which the quadrature rule for the implicit piece does not include the left-hand endpoint t_n (i.e. the LR and RR rules for which $\tilde{q}_m^0 \equiv 0$ in Eq. (2.13)). Then for any fixed λ_E ,*

$$(3.11) \quad \lim_{|\lambda_I| \rightarrow \infty} \rho(\lambda_E, \lambda_I) = 0.$$

The proof is essentially the same as that for Thm. 3.1 and is hence omitted.

Another possible way in which to define semi-implicit linear stability is to consider

$$(3.12) \quad y' = \lambda_E y + \lambda_I y,$$

where $\lambda_E + \lambda_I = \lambda$. With this choice the stability region is again $|\rho(\lambda)| \leq 1$, but clearly different procedures for choosing λ_E and λ_I will produce different types of stability regions. It should be noted that with this choice, $\rho(\lambda)$ for PIDC methods is not a rational function of λ , but rather a ratio of polynomials in λ_E and λ_I . Therefore the limit in Eq. (3.4) will in general depend on how λ approaches infinity. Nevertheless, a similar result to that in Thm. 3.2 holds.

THEOREM 3.3. *Let $\rho(\lambda)$ be the amplification factor based on Eq. (3.12) determined by a semi-implicit PIDC method based on forward-backward Euler for which the quadrature rule for the implicit piece does not include the left-hand endpoint t_n (i.e. $\tilde{q}_m^0 \equiv 0$ in Eq. (2.13)). Then for any fixed λ_E ,*

$$(3.13) \quad \lim_{|\lambda| \rightarrow \infty} \rho(\lambda) = 0.$$

Again the proof is essentially the same as for Thm. 3.1 and hence omitted.

One useful choice for Eq. (3.12) is

$$(3.14) \quad y' = \mathbb{I}(\lambda)y + \mathbb{R}(\lambda)y,$$

with $\lambda_E = \mathbb{I}(\lambda)$ and $\lambda_I = \mathbb{R}(\lambda)$. This is, among other things, the relevant decomposition for analyzing the stability of an approximation to an advection-diffusion equation based on finite differences and the method of lines, and is considered in [25, 2, 1, 6]. With this definition, no semi-implicit method can be A -stable, since for pure imaginary λ , such methods are explicit. However A -stability can certainly be achieved.

In Sect. 3.3, stability diagrams for semi-implicit methods based on both of the above definitions are shown. The diagrams indicate that it is possible to construct semi-implicit PIDC methods which are $A(\alpha)$ -stable for either definition. In addition, those methods which are formed from LR or RR quadrature rules are also $L(\alpha)$ -stable by Thms. 3.2 and 3.3.

3.2 Stability diagrams for implicit methods

In this Section, stability diagrams for fully implicit PIDC methods are presented to provide a numerical confirmation of the $L(\alpha)$ -stability of methods using right-hand quadrature rules and to enable a comparison of the stability regions for methods using different nodes.

First, the effect of omitting the left-hand point in the quadrature rule is demonstrated. In Fig. 3.1, the stability regions for 6th-order implicit PIDC methods based on backward Euler with uniform points are presented. The contours correspond to $\rho(\lambda) = 1, 0.5, 0.1, 0.01, 0.001$. hence the stability region is the region outside the curve $\rho(\lambda) = 1$. Note that the axes are scaled by a cubic transformation. The left-hand endpoint was used in the quadrature rule in panel A, but not in panel B. Both methods appear to be $A(\alpha)$ -stable, but note that only in panel B does the amplification factor approaches zero with large λ as Thm. 3.1 demands, and hence only the method using right-hand quadrature rule is $L(\alpha)$ -stable. This difference is present for methods with other orders of accuracy and with those using Gaussian quadrature nodes as well.

To give a better understanding of the $L(\alpha)$ -stability of different choices of order and nodes, Fig. 3.2 shows a magnified view of the stability regions near the imaginary axis for the implicit PIDC methods with different choices of nodes. Results are shown for 6th-, 7th-, and 10th-order methods, each using right-hand quadrature rules. These results suggest that these methods are all $A(\alpha)$ -stable for $\alpha > 89.9^\circ$. To give two specific examples, the value of α for the 10th-order

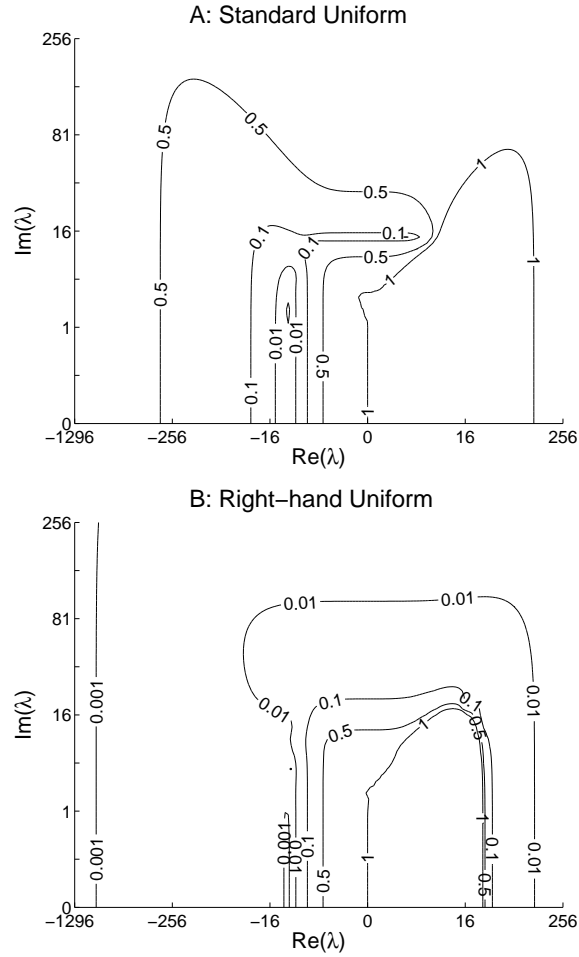


Figure 3.1: Level curves corresponding to $\rho(\lambda) = 1, 0.5, 0.1, 0.01, 0.001, 0.0001$, for 6th-order implicit PIDC methods with uniform nodes. The panels differ in the use of the left-hand endpoint in the quadrature rule.

method based on Gauss-Lobatto points is approximately 89.982° , while the value for the 6th-order method based on uniform points is greater than 89.999° (but less than 90°).

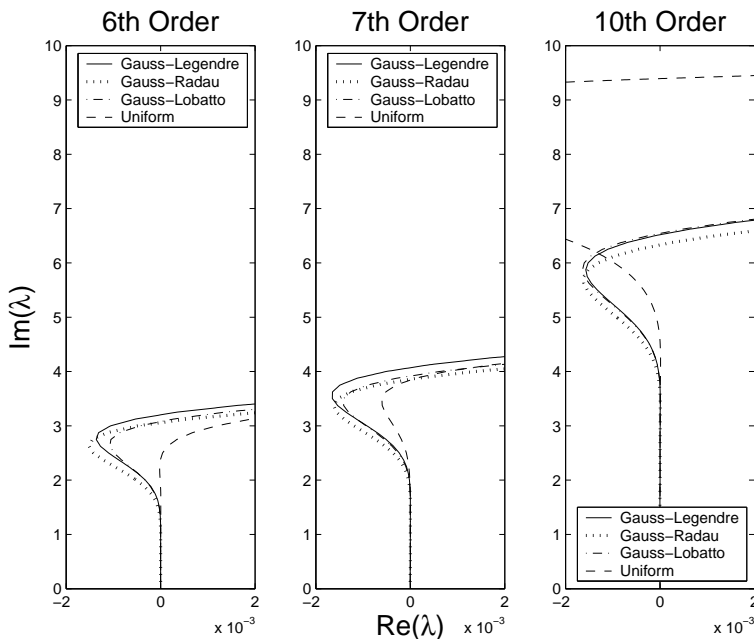


Figure 3.2: Level curves corresponding to $\rho(\lambda) = 1$, for sixth-, seventh, and tenth-order implicit PIDC methods with different choices of quadrature nodes. Each method uses a right-hand quadrature rule, and the stability regions are to the right of the curves.

3.3 Stability diagrams for Semi-implicit methods

The stability regions for semi-implicit PIDC (SIPIDC) methods based on the forward-backward Euler method are now considered. First the diagrams corresponding to the definition in Eq. (3.10) are shown in Figs. 3.3A, 3.3B, and 3.3C for the sixth-order SIPIDC method using Gauss-Labotto points and the LR, LL, and RR rules, respectively. The curves in these figures are generated numerically by computing for a uniform set of points λ_I the maximum of $|\rho(\lambda_E, \lambda_I)|$ taken over a uniform set of points λ_E in the stability region of the explicit method. The number of points λ_E considered differs from figure to figure due to shape of the stability region (327, 272, and 257 for panels A,B, and C respectively). Hence these figures should be viewed only as approximations as opposed to the figures in [13] which were computed analytically.

The stability regions shown suggest that, as in Fig. 3.1, omitting the the left-hand point in the quadrature rule for the explicit piece allows the methods to have $L(\alpha)$ -stability. In panel B, however, there is also an unstable region near the origin. For this choice of points, there are values of λ_E near the boundary of the explicit stability region for which the semi-implicit method is not stable near the origin. This can occur as well for LR rules; as one example, the seventh-order method using Lobatto points displays similar behavior. However, if one restricts

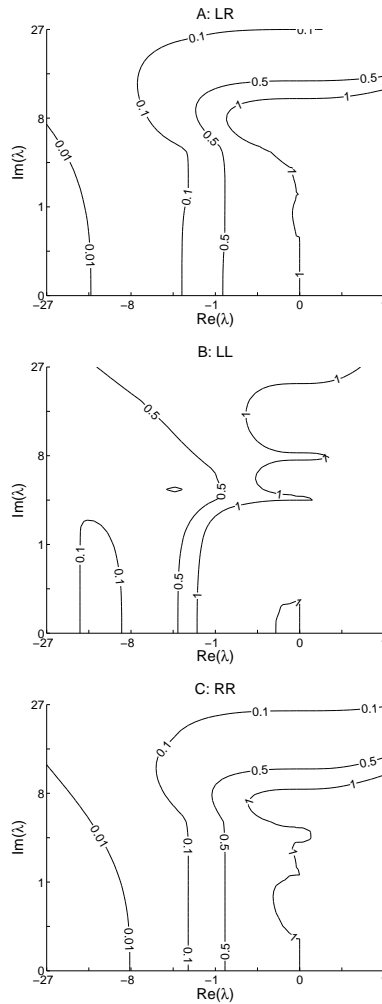


Figure 3.3: Stability regions for 6th-order semi-implicit PIDC methods using Gauss-Lobatto nodes. The contours correspond to $\rho(\lambda_I) = 1$ based on the definition in Eq. (3.10). The panels differ in the use of the left-hand endpoint in the quadrature rule.

somewhat the allowable values of λ_E , these regions of instability vanish. As a specific example, the region of instability in panel B disappears if one restricts the values of λ_E considered to those for which the explicit amplification factor is smaller than 0.9. In the context of numerical methods for advection-equation type PDEs, this is analogous to reducing the time-step by a small amount below the maximum value allowed by the CFL restriction.

In Fig.3.4, the semi-implicit stability regions for sixth-order methods using different choices of nodes are displayed. In this figure, each of the methods uses

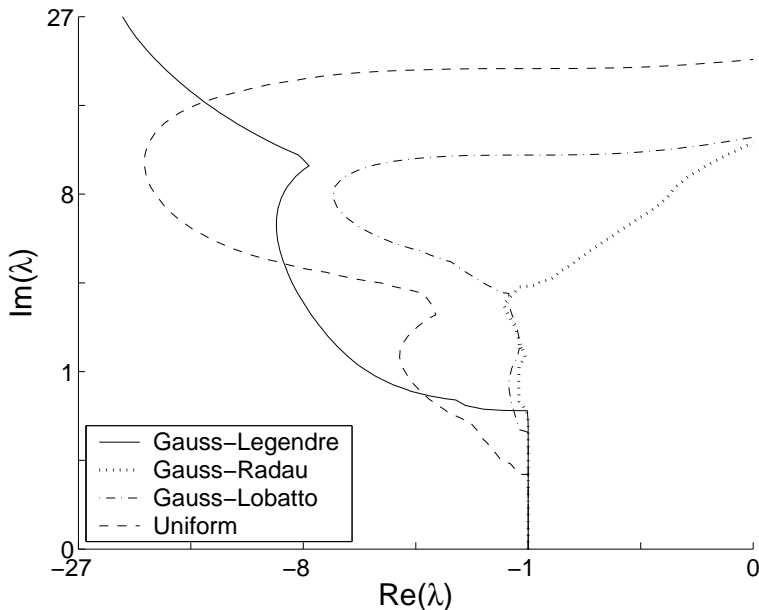


Figure 3.4: Stability diagrams for 6th-order semi-implicit PIDC methods with various choices of quadrature nodes. A LR quadrature rule is used for each, and the curves correspond to $\rho(\lambda) = 1$ based on the definition in Eq. (3.10).

a LR quadrature rule, and each choice of quadrature nodes results in an $L(\alpha)$ -stable method with $\alpha > 45^\circ$. The value for α for the method using Gauss-Radau points is actually much closer to 90° . However, this behavior does not appear to be generic for other orders of accuracy.

Finally, stability diagrams based on the definition in Eq. (3.14) are included for comparison. Fig. 3.5 shows the stability regions $\rho(\lambda) = 1$ for 6th-order SIPIDC methods obtained for LR rules using Gauss-Legendre, Gauss-Radau, Gauss-Lobatto, and uniform nodes. These stability diagrams are somewhat less informative and show no significant effect of the choice of nodes on the stability diagrams.

4 Accuracy Diagrams

Although instructive, the stability diagrams presented in the last section give limited information regarding the comparative accuracy of different PIDC methods. One simple way in which to compare the accuracy of methods is to consider the accuracy region for the linear model problem defined in Eq. (3.14). For a given tolerance ε , the *accuracy region* of a method is defined as the region in the complex plane for which $|y_1(\lambda) - e^\lambda| < \varepsilon$, where again $y_1(\lambda)$ denotes the numerical solution after one time step with $\Delta t = 1$.

Accuracy regions for semi- and multi-implicit SDC methods are presented in

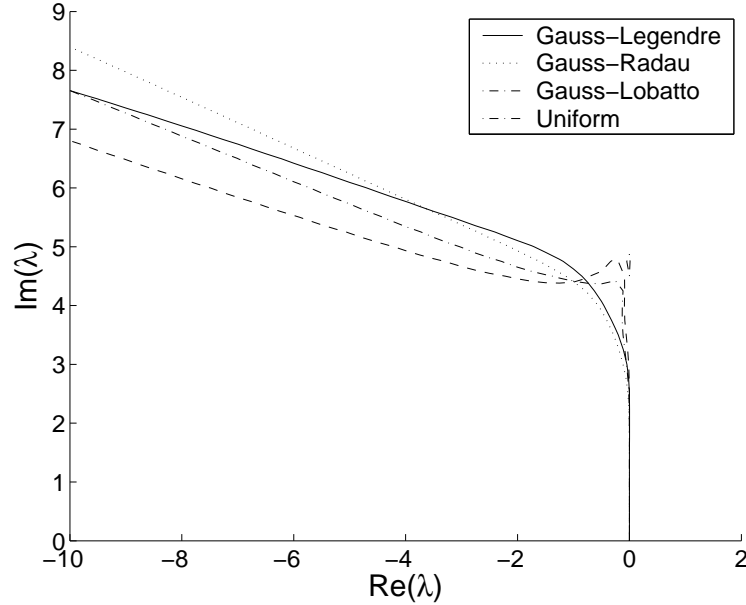


Figure 3.5: Stability diagrams for 6th-order semi-implicit PIDC methods with various choices of quadrature nodes. A LR quadrature rule is used for each, and the curves correspond $\rho(\lambda_I) = 1$ based on the definition in Eq. (3.14).

[25, 5, 24] along with some comparisons with accuracy regions from semi-implicit Runge-Kutta and defect correction methods. These results show that the size of the accuracy regions for SDC methods compares favorably with competing methods, and that even when scaled by the number of function evaluations required, higher-order SDC methods have larger accuracy regions than their lower order counterparts for small enough ϵ . In this section, the accuracy regions for SIPIDC methods are compared for different choices of quadrature nodes.

In the following examples, LR quadrature rules are used. Similar results are obtained using LL or RR rules. Fig. 4.1 displays accuracy regions for SIPIDC methods with orders 4, 7, and 10, for an error tolerance of $\epsilon = 10^{-4}$. Which set of quadrature nodes yield the highest degree of accuracy depends on the order of the overall method and on λ . For sufficiently small $|\Re(\lambda)|$, Gauss-Legendre nodes are the most accurate for the 4th-order method, whereas, somewhat surprisingly, 7th- and 10th-order methods using uniform nodes have the highest accuracy. For sufficiently large $|\Re(\lambda)|$, methods using Gauss-Lobatto nodes are the most accurate. In Fig. 4.2, the corresponding accuracy regions for $\epsilon = 10^{-8}$ are displayed. As in Fig. 4.1, the accuracy regions are larger for higher-order methods, and the increase in size is greater than the increase in computational cost. It is significant to note that even with tenth-order methods, the accuracy regions for PIDC methods with uniform nodes is larger than those using Gaus-

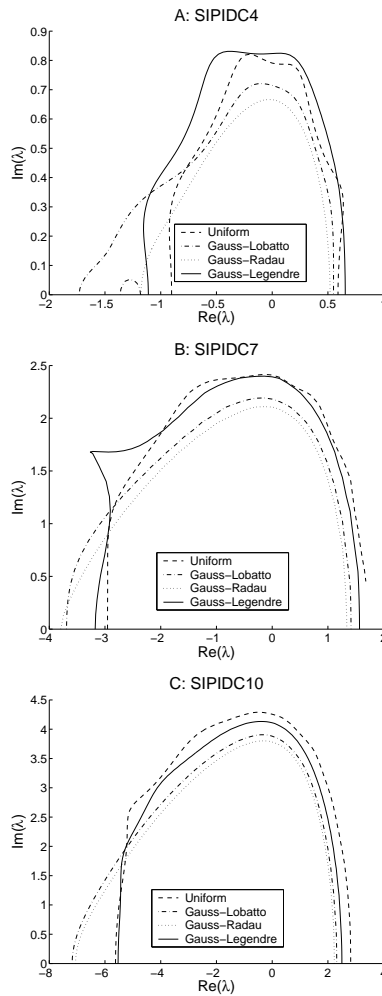


Figure 4.1: Accuracy diagrams for 4th-, 7th-, and 10th-order SIPIDC methods with tolerance $\varepsilon = 10^{-4}$. A LR quadrature rule is used in each case.

sian nodes. The implication is that, for this simple example, the use of Gaussian quadrature nodes does not provide a significant increase in accuracy compared to using uniform nodes for moderately high order.

It is worth commenting that the accuracy diagrams corresponding to uniform and Gaussian nodes are more similar in size when RR quadrature rules are used (diagrams not shown). This is not surprising since the Gauss-Legendre nodes do not normally include the left endpoint. However, even in this case, there is no evidence that using Gauss-Legendre nodes provides significantly better accuracy than uniform nodes for the orders considered. One must draw the conclusion from these tests that the component of the overall error which comes from the

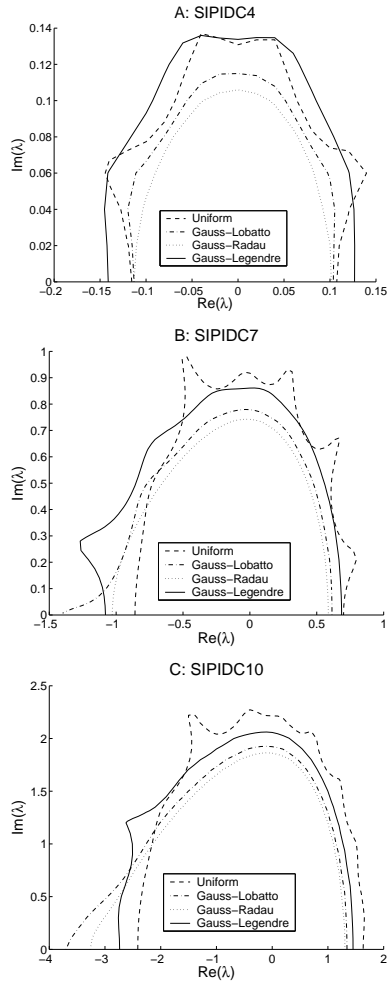


Figure 4.2: Accuracy diagrams for 4th-, 7th-, and 10th-order SIPIDC methods with tolerance $\epsilon = 10^{-8}$. A LR quadrature rule is used in each case.

particular quadrature rule use in Eq. (2.13) is not the dominant source of error in the numerical method. These conclusions are further supported by the results using less simplistic problems in the next section.

5 Semi-implicit Numerical Tests

In this section, the accuracy of various SIPIDC methods is compared for both stiff and non-stiff problems. Two different test equations are used in the comparisons. The first test equation is the scalar ODE with exact solution

$y(t) = \cos(2\pi t)$ given by

$$(5.1) \quad y(t)' = -2\pi \sin(2\pi t) - \frac{1}{\epsilon}(y - \cos(2\pi t))$$

$$(5.2) \quad y(0) = 1.$$

As $\epsilon \rightarrow 0$ this equation becomes increasingly stiff. For the semi-implicit methods considered here, the term $-2\pi \sin(2\pi t)$ in Eq. (5.1) is treated explicitly and the term $-(y - \cos(2\pi t))/\epsilon$ implicitly. This example will be referred to simply as the *cosine test*.

The second test uses the van der Pol equation, which is a popular nonlinear test problem for methods for stiff ODEs. The equation prescribes the motion of a particle $x(t)$ by

$$(5.3) \quad x''(t) + \mu(1 - x(t)^2)x'(t) + x(t) = 0.$$

Making the transformation $y_1(t) = x(t)$, $y_2(t) = \mu x'(t)$ and $t = t/\mu$, one obtains the system

$$(5.4) \quad y_1'(t) = y_2(t)$$

$$(5.5) \quad y_2'(t) = \frac{1}{\epsilon} \left(-y_1(t) + (1 - y_1(t)^2)y_2(t) \right),$$

where $\epsilon = 1/\mu^2$. As ϵ approaches zero, these equations become increasingly stiff. The semi-implicit methods considered here treat the first equation explicitly and the second implicitly, as in earlier tests in [25, 19]. All calculations reported below were performed in double precision using MATLAB programs.

5.1 Non-stiff examples

The first set of numerical examples is designed to compare the efficiency and accuracy of SIPIDC methods of differing orders and quadrature nodes for a non-stiff example, hence the cosine test with non-stiff values of ϵ is used. Three examples are included, the first comparing the efficiency for different orders of accuracy, and two additional tests comparing different choices of nodes and quadrature rules.

For each test below, the equations are computed for $t \in [0, 10]$, and the reported error is the discrete L_2 norm of the error in time of the computed solution $y(t_n)$ at each time step.

Fig. 5.1 shows log-log plots of the error versus the number of implicit function evaluations for approximations computed by the k th-order SIPIDC methods, for $k = 3, \dots, 8$. For this example, uniform quadrature points are used, and the left-hand endpoint is only used in the quadrature rule for the explicit piece (i.e., LR rules). Although for this problem the solution procedure for the implicit piece is in fact explicit, for most practical problems the number of implicit function evaluations is the relevant measure of computational cost.

Each SIPIDC method yields numerical solutions with the expected k th-order accuracy. As the results below indicate, qualitatively similar results are also obtained using Gaussian quadrature nodes or using LL and LR rules. For a given

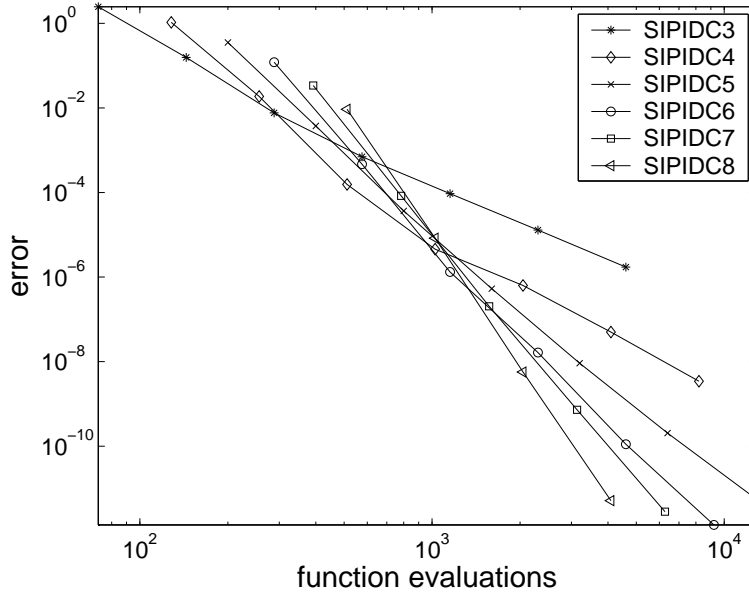


Figure 5.1: Efficiency comparison for k th-order SIPIDC methods (denoted SIPIDC k), $k = 3, \dots, 8$, using the cosine test case with non-stiff parameter $\epsilon = 0.5$. For each run, uniform nodes are used with a LR quadrature rule.

Δt , the computational costs associated with higher-order SIPIDC methods are higher, owing to the larger number of substeps per time step and the number of PIDC iterations. Nonetheless, for a sufficiently high accuracy requirement, higher-order methods are more efficient, in the sense that a given accuracy requirement is met with a lower computational cost.

The second non-stiff example compares the accuracy of differing choices of quadrature nodes. Fig. 5.2 shows efficiency results for fifth-order SIPIDC methods for Gauss-Legendre, Gauss-Radau, Gauss-Lobatto, and uniform points. In panel A, LR quadrature rules are used, while in panel B, RR quadrature rules are used. Fifth-order convergence is exhibited for each choice of nodes.

Note however that the choice of nodes with the smallest error depends on whether a LR or RR quadrature rule is being used. For LR rules, the accuracy of the solution computed using Gauss-Legendre nodes is substantially (in this case, approximately one order of magnitude) less accurate than for other nodes, while Gauss-Radau, Gauss-Lobatto, and uniform points all generate approximations with similar accuracy. This holds true for LL quadrature rules as well (data not shown). On the other hand, for RR rules, Gauss-Legendre nodes yield accuracy similar to Gauss-Lobatto nodes, whereas uniform nodes are slightly less accurate for sufficiently small Δt , and Gauss-Radau nodes yield solutions that are approximately one order of magnitude more accurate than other nodes. Similar results were also obtained for SIPIDC methods of other orders of accu-

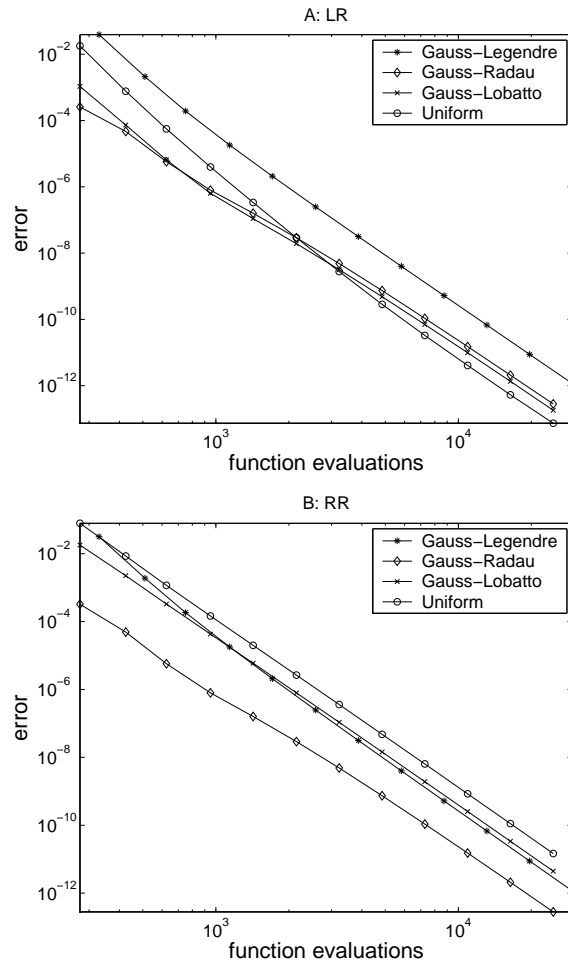


Figure 5.2: Efficiency comparison for 5th-order SIPIDC methods using the cosine test case with non-stiff parameter $\epsilon = 0.5$. A LR quadrature rule is used in panel A, while in panel B, a RR quadrature rule is used.

racy. These results suggest that the accuracy of SIPIDC methods using differing quadrature nodes depends on the quadrature rule used in the methods.

Motivated by the results of the last example, a comparison of the four choices of nodes using the quadrature rule that performed the best on the previous example is presented. Specifically LR quadrature rule is used with uniform and Gauss-Lobatto nodes, and a RR quadrature rule is used with Gauss-Legendre and Gauss-Radau nodes. The cosine problem is repeated using fifth-order SIPIDC, this time for $\epsilon = 1/10$, and a graph of error versus function evaluations is presented in in Fig. 5.3. For each choice of quadrature rule, the solution is converging with fifth order, and the error for each choice is roughly the same.

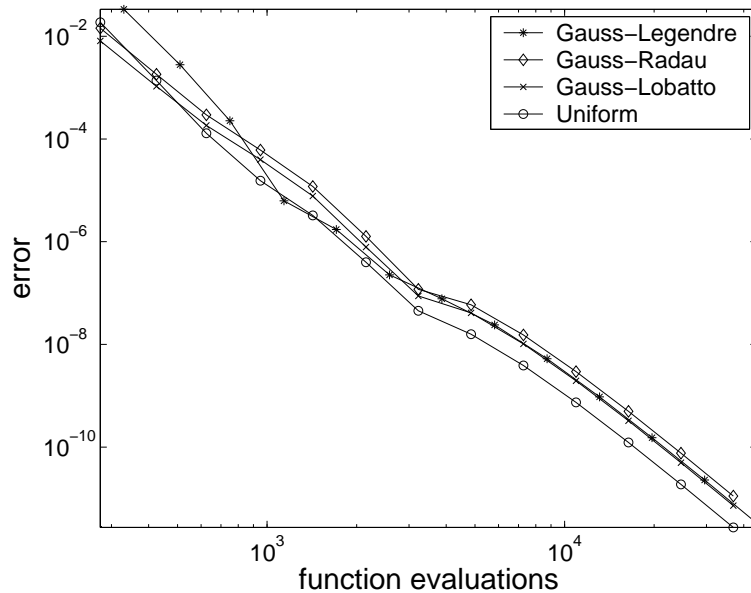


Figure 5.3: Efficiency comparison for 5th-order SIPIDC methods using the cosine test case with non-stiff parameter $\epsilon = 0.1$. A LL quadrature rule is used for uniform and Gauss-Lobatto nodes, while a RR quadrature rule is used for Gauss-Legendre and Gauss-Radau nodes.

The most relevant point demonstrated by these tests is that methods that use uniform quadrature nodes produce results of comparable accuracy to methods using Gaussian quadrature rules for these non-stiff examples. Even for methods as high as order ten, which choice of nodes produces the most accurate solution appears to be problem dependent. On the other hand, the examples in the next section show that methods using uniform nodes have a very different behavior than those using Gaussian nodes for stiff problems.

5.2 Stiff examples and order reduction

Since SIPIDC methods are meant to be used to approximate the solution to equations with both stiff and non-stiff terms, numerical results for stiff equations are presented in this section. In [25], convergence tests are conducted to compare the performance of semi-implicit SDC methods (i.e., PIDC methods based on Gauss-Lobatto nodes) as well as the semi-implicit Additive Runge-Kutta (ARK) methods from [19]. Numerical results using the van der Pol equation in both [25, 19] demonstrate order reduction when ϵ is sufficiently small, although the irregular appearance of the convergence plots in both papers makes it difficult to determine the extent of order reduction. The following example demonstrates that this difficulty is due to the choice of initial conditions in these examples rather than the methods themselves.

In [19], the numerical examples using the van der Pol equation use an integration interval of $t \in [0, 0.5]$ and initial conditions $y_1(0) = 2$ and $y_2(0) = -0.6666654321121172$. In order to enable a direct comparison, these parameters are also used in [25]. Unfortunately, these initial conditions are near the equilibrium solution only for ϵ approximately equal to 10^{-5} . For other values of ϵ , solutions starting from these initial conditions will exhibit sharp (albeit small) initial transients. Since uniform time steps are used for these tests, the error in the first time step will pollute convergence results when these transients exist.

Table 5.1: Initial conditions for the van der Pol equation example.

ϵ	$y_1(0)$	$y_2(0)$
10^{-1}	2	-0.65
10^{-3}	2	-0.66654321
10^{-4}	2	-0.666654321
10^{-5}	2	-0.6666654321
10^{-6}	2	-0.66666654321
10^{-7}	2	-0.666666654321

To illustrate this, consider the results shown in Fig. 5.4. For this example, errors are computed for the van der Pol equation for values of $\epsilon = 10^{-k}$, $k = 1, 3, 4, 5, 6, 7$. The starting values (which were determined numerically) are given in Table 5.1. This figure shows the error versus function evaluation for the 4th-order SIPIDC method using Gauss-Lobatto nodes. This method is the same as the fourth-order method used in [25], hence the results for $\epsilon = 10^{-3}$ in Fig. 5.4 can be compared to those in Figs. 5.4 and 5.5 in [25]. Fig. 5.4 shows that the order of accuracy of the solution drops to first order for a range of Δt for both components of the solution. Furthermore, the first-order error in the solution scales linearly with ϵ . These results are consistent with the results from a stiff linear system example in [25].

For comparison, the test above is repeated using the ARK4(3)6L[2]SA scheme from [19]. The results are shown in Fig. 5.5 and should be compared to Figs. 8 and 9 in [19]. For the first component of the solution, the errors convergence throughout the range of values of ϵ at near fourth-order except for a slight deviation with $\epsilon = 10^{-3}$. On the other hand, the convergence rate is reduced to first order in the second component, and the first order error again scales linearly with ϵ as with the SIPIDC example above.

The errors reported in both [25, 19] are simply an approximation to the error at the final time $T = 0.5$ computed using a reference solution. Here the errors are computed as in the cosine test except that instead of using the exact solution, a reference solution computed using a seventh-order implicit PIDC method and very small time step (exactly how small was determined experimentally and is dependent on ϵ). The fact that solutions computed using both PIDC and ARK methods convergence to the reference solution to machine precision attests to the accuracy of the solution. The use of error norm in time yields slightly smoother

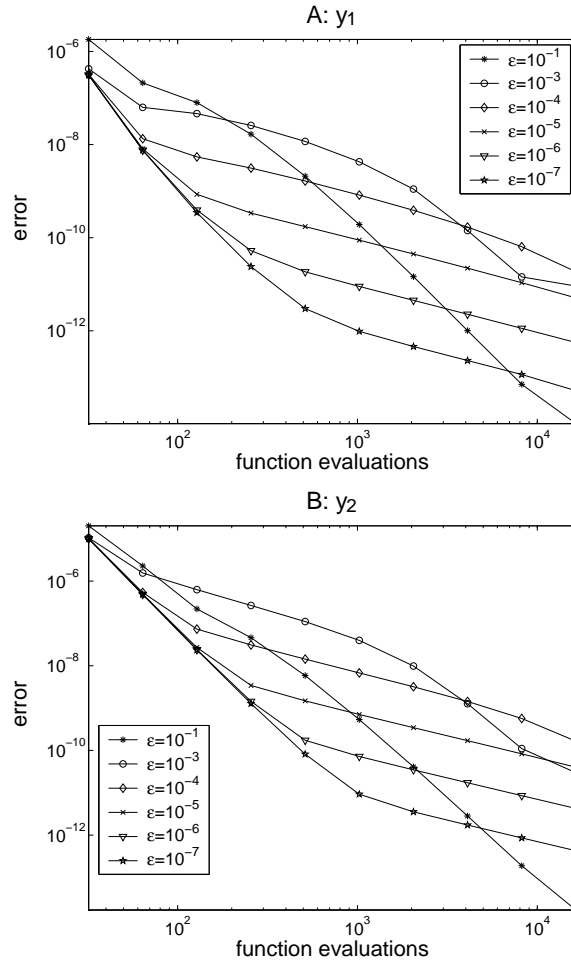


Figure 5.4: Efficiency comparison for 4th-order SIPIDC methods using Gauss-Lobatto points for the van der Pol problem for a range of ϵ values.

convergence plots, compared to measuring the error only for the final solution at T . Nonetheless, the extent of the order reduction does not depend on which form of the error is examined.

To investigate the dependence of order reduction on the choice of nodes, the above test was repeated for Gauss-Legendre, Gauss-Radau, Gauss-Lobatto, and uniform nodes, all using LR quadrature rules. Fig. 5.6 shows the y_2 error versus the number of function evaluations for approximations computed by the 7th-order SIPIDC method, for $\epsilon = 10^{-k}$, $k = 1, 3, 4, 5, 6$. Results for y_1 are similar. For sufficiently stiff parameters ($\epsilon < 10^{-4}$), the convergence rate drops to first order for a range of Δt when Gauss-Legendre, Gauss-Radau, and Gauss-Lobatto nodes are used. In contrast, when uniform nodes are used, a region of

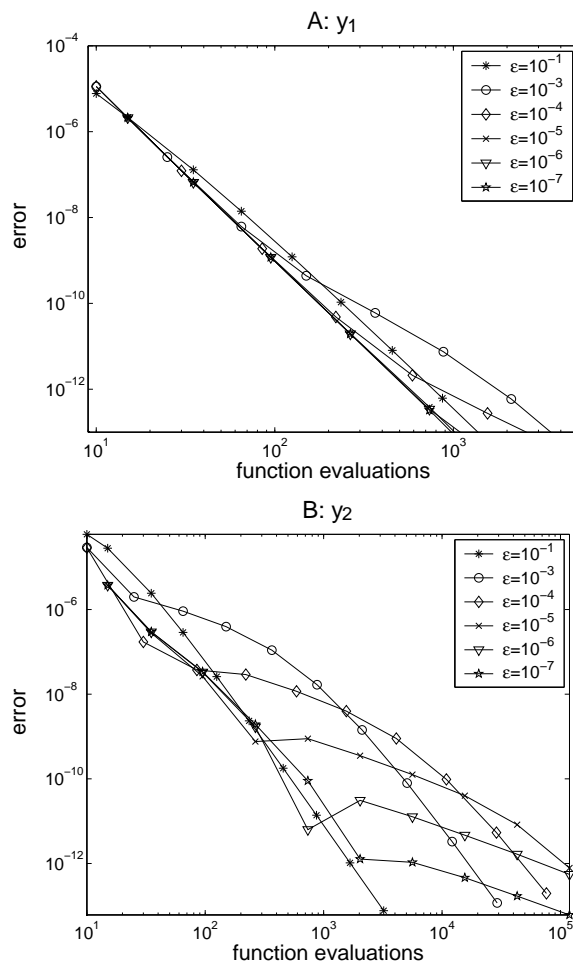


Figure 5.5: Convergence plots for the additive Runge-Kutta method ARK4(3)6L[2]SA for the van der Pol problem for a range of ϵ values.

non-convergence appears (i.e., 0th order) for sufficiently stiff parameters. Also, the magnitude of the error in these regions of non-convergence appears to scale like ϵ^2 .

It is noteworthy that these results are not consistent with the suggestion in [25, 19] that the error for semi-implicit SDC and ARK methods scales like

$$(5.6) \quad e \sim c_1 \Delta t^\alpha + c_2 \epsilon \Delta t^\beta,$$

for $\epsilon < C\Delta t$, where c_1, c_2, C, α , and β are constants. This ansatz is based on the results in [11, 17] for SDIRK methods. The results in Fig. 5.6 suggest that the error for PIDC methods scales like

$$(5.7) \quad e \sim c_0 \Delta t^{r_0} + c_1 \epsilon \Delta t^{r_1} + c_2 \epsilon^2 \Delta t^{r_2}.$$

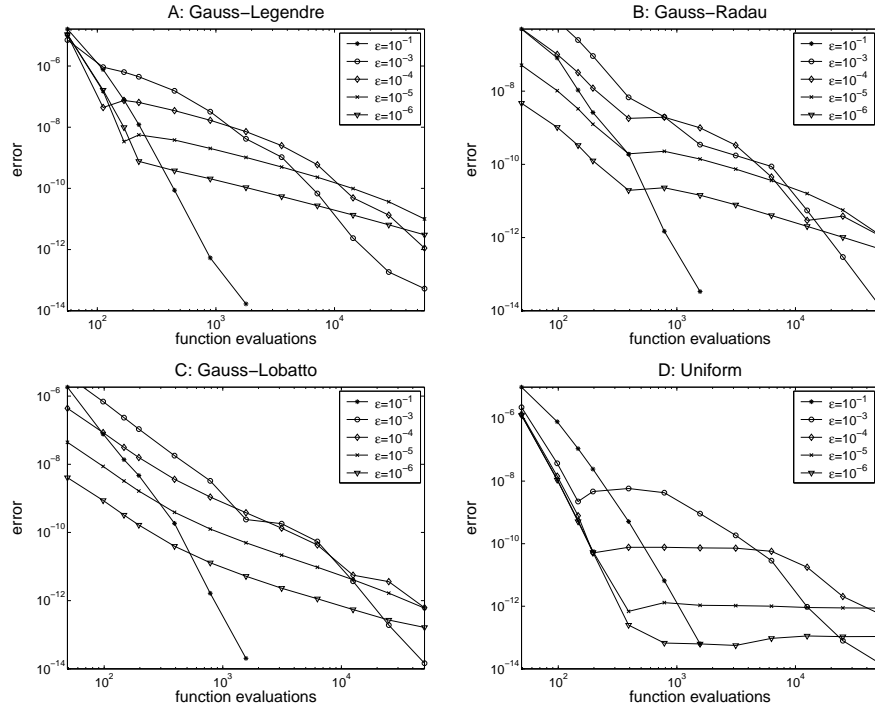


Figure 5.6: Convergence plots for 7th-order SIPIDC methods for the van der Pol problem for a range of ϵ values. A LR quadrature rule is used for each case, and the reported error is for the second component y_2 .

Here r_0 denotes the non-stiff order and the coefficients c_1 and c_2 vanish for sufficiently small Δt . Furthermore the exponents r_1 and r_2 representing the reduction in order appear to depend on the choice of quadrature nodes. This ansatz is further investigated in the following section.

Finally, the effect on order reduction of using or omitting the left endpoint in the quadrature rule is now studied. Fig. 5.7 shows convergence results obtained for y_2 using 7th-order SIPIDC methods, with $\epsilon = 10^{-k}$, $k = 1, 3, 4, 5, 6$, using LL and RR uniform nodes. Results for y_1 are qualitatively similar (not shown). When RR uniform nodes are used (Fig. 5.7B), the convergence results are similar to those obtained with LR uniform nodes (Fig. 5.6D). In contrast, when LL uniform nodes are used (Fig. 5.7A), the convergence results resemble those obtained with non-uniform nodes (Figs. 5.6A, B, and C). Analogous results are also obtained for SIPIDC methods of other orders (not shown). These results suggest indicate the coefficients in the error in Eq. (5.7) also depend on whether the left-hand endpoint is used in the quadrature rule. These curiosities are further examined in the next Section.

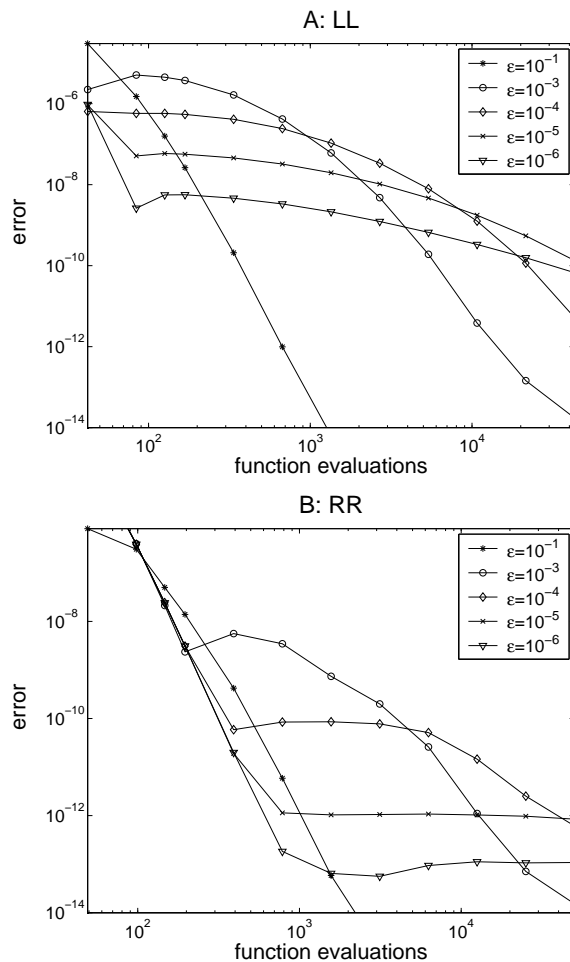


Figure 5.7: Convergence results for 7th-order SIPIDC methods using uniform nodes for the van der Pol problem for a range of ϵ values. The panels differ in the choice of quadrature rule. The reported error is for the second component y_2 .

5.3 Order reduction on the cosine test

The numerical results regarding order reduction in the previous section are at first difficult to explain. Fortunately, very similar results are obtained using the much simpler cosine test problem. This equation is simple enough to allow an explicit examination of the dominant error terms for PIDC methods, which helps to justify the form of the error suggested in Eq. (5.7).

To demonstrate the similar behavior between the two problems, the cosine test is repeated for increasingly stiff values of ϵ . Fig. 5.8 shows the error versus the number of implicit function evaluations for approximations computed using

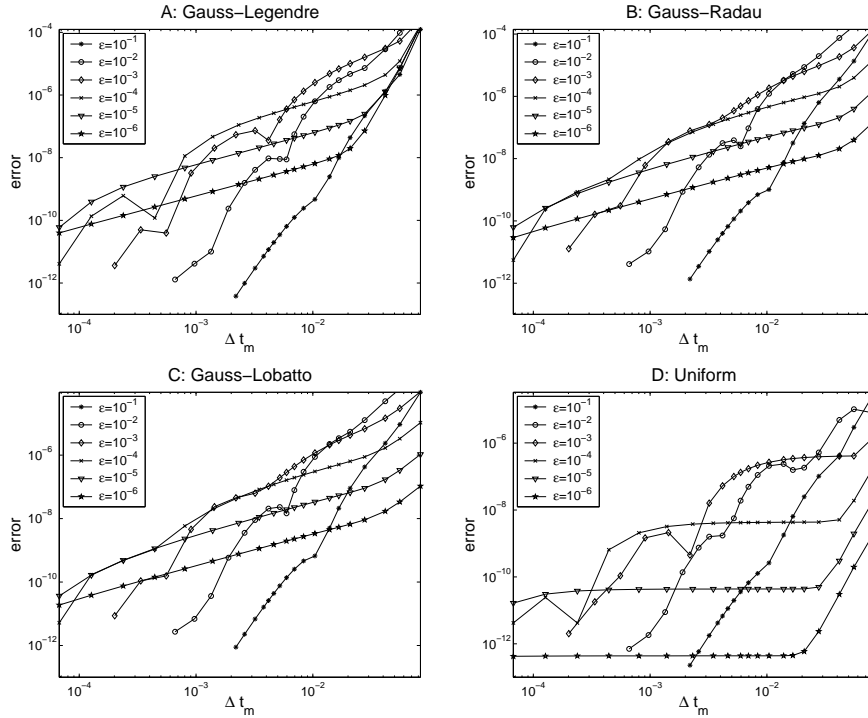


Figure 5.8: Error versus average substep size for 6th-order SIPIDC methods using the cosine test case for a range of ϵ values, from 10^{-1} to 10^{-6} . A LR quadrature rule is used in each case.

the 6th-order SIPIDC methods with $\epsilon = 10^{-k}$, $k = 1, 2, \dots, 6$. Results obtained for each choice of node with the LR quadrature rule are shown separately in the four panels. The results are consistent with those obtained for the van der Pol equation. For non-stiff parameters ($\epsilon = 0.1$ and 0.01), 6th-order approximations are obtained for all three quadrature nodes as expected. For sufficiently stiff parameters ($\epsilon < 0.001$), a region of first-order convergence is observed for methods using Gauss-Legendre, Gauss-Radau, and Gauss-Lobatto nodes (wherein the error scales like ϵ), while a 0th-order region appears for uniform nodes (wherein the error scales like ϵ^2).

That uniform nodes yield convergence behavior different from non-uniform nodes when applied to a stiff problem can be explained by the error formula (A.24) derived in the Appendix. In that formula, the low-order terms have the form $c_1 \epsilon \Delta t + c_2 \epsilon^2$, where for uniform nodes and a right-hand rule, c_1 vanishes. Although this error formula is valid only for a simple equation of the same form as the cosine test, it is consistent with the numerical results (Fig. 5.8), which show that

- for a sufficiently stiff problem, uniform nodes yield zeroth-order approx-

imations with errors that scale as ϵ^2 , whereas non-uniform nodes yield first-order approximations with errors that scale as ϵ ;

- for a given stiffness parameter, order reduction begins for uniform nodes at smaller Δt 's compared to non-uniform nodes; and
- for $\epsilon > \Delta t_m$ the error formula (5.7) is not valid and one should observe the convergence with full order for sufficiently small Δt .

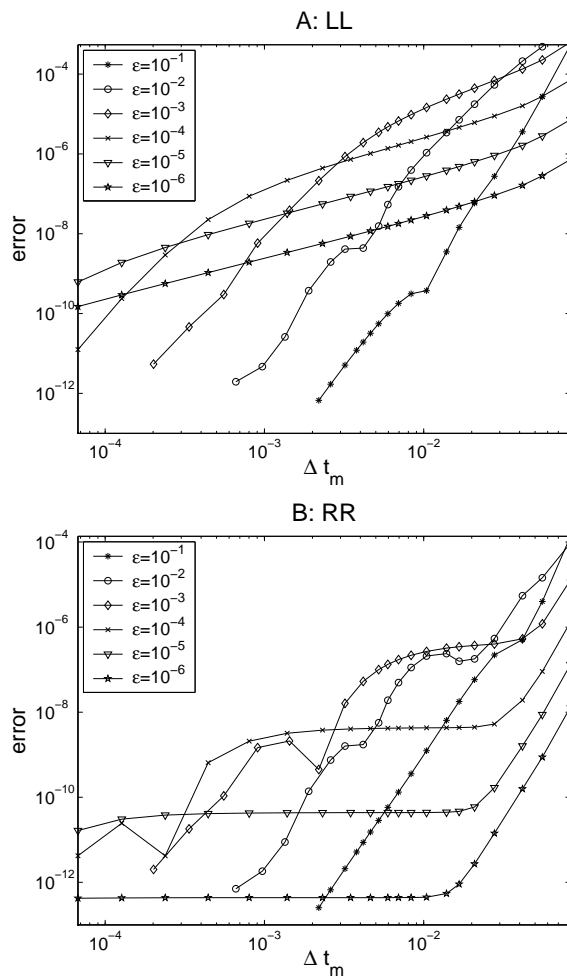


Figure 5.9: Errors versus average substep size for the cosine test case using 6th-order SIPIDC methods with uniform nodes. The panels differ in the choice of quadrature rule.

The cancellation of the $\epsilon\Delta$ term in the error formula (A.24) occurs only for uniform points and a right-hand quadrature rule. To confirm this the convergence

behavior for the alternative quadrature rules LL and RR is now investigated. Fig. 5.9 shows convergence results obtained for 6th-order SIPIDC methods using LL and RR uniform nodes. Similar to what was observed for the stiff van der Pol problem, results for RR uniform nodes (Fig. 5.9B) resemble those obtained with LR uniform nodes (Fig. 5.8D), whereas results for LL uniform nodes (Fig. 5.9A) resemble those obtained with non-uniform nodes (Figs. 5.8A, B, and C). These results confirm that the cancellation occurs for uniform nodes with LR and RR quadrature rules, but not for LL nodes.

6 Discussion

An analysis is presented in this paper of the impact of the choice of quadrature nodes and rules on the accuracy and stability of implicit and semi-implicit PIDC methods which are based on first-order Euler methods. Several pertinent conclusions are drawn from examples and analysis:

1. Implicit PIDC methods that are based on backward Euler and that use a right-hand quadrature rule (i.e. those that do not use the value of the left-hand end-point in Eq. (2.12)) are $L(\alpha)$ -stable, with α very close to $\pi/2$.
2. For SIPIDC methods applied to nonstiff or mildly-stiff problems, similar accuracy is obtained using Gauss-Legendre, Gauss-Radau, Gauss-Lobatto, and uniform nodes for moderately high order (e.g. seven or less). Which nodes give the best accuracy per function evaluation appears to be problem dependent. Thus, an argument may be made for choosing uniform points instead of the non-uniform Gauss points used in previous studies [10, 23, 5] when it is more convenient.
3. The ambiguous results concerning order reduction on the van der Pol problem appearing in both [25, 19] are due to the use of non-equilibrium initial conditions. Results presented in this study show that fourth-order ARK scheme in [19] displays an $\mathcal{O}(\epsilon\Delta t)$ error in the second component of the solution for small values of the stiffness parameter ϵ , and likewise the semi-implicit SDC method used in [25] displays a reduction to first order in both components of the solution when equilibrium initial conditions are used.
4. The particular form of order reduction for SIPIDC methods applied to the van der Pol test case depends on the choice of quadrature nodes and quadrature rules. In particular, numerical results using uniform points with a right-hand quadrature rule display $\mathcal{O}(\epsilon^2)$ error terms as opposed to the $\mathcal{O}(\epsilon\Delta t)$ error displayed by methods using nonuniform nodes or the left-hand rule.
5. The SIPIDC methods display similar order reduction on the cosine test as on the van der Pol test. Furthermore, the simple form of the cosine test allows a direct confirmation of the leading terms in the order reduction.

As mentioned earlier, the motivation for studying SIPIDC methods for ODEs is to enable the construction of higher-order methods for PDEs with multiple disparate time scales. The conclusions above have relevant consequences in this regard. To begin, there is an additional benefit to using right-hand quadrature rules for the temporal integration of certain PDEs in that the value of the implicit term needs never be computed explicitly, even at the beginning of the computation. As a particular example, consider the numerical methods for viscous incompressible flow described in [23, 24] wherein the implicit term stems from the diffusion term in a PDE. In order to achieve higher-order spatial accuracy, the resulting implicit equations are solved in spectral space for uniform grids or with integral equation solvers [7] for adaptive meshes. In either case, additional work is required if one wishes to simply evaluate the viscous term explicitly at a given time. In the latter case, using for example finite-difference approximations near coarse-fine grid interfaces requires unwieldy stencils and results in a less accurate value compared to that computed implicitly through the integral equation method.

The second conclusion above also has implications for the construction of numerical methods for PDEs. There are examples where using uniform nodes in time simplify the overall algorithm considerably. One example stems from the use of block rectangular adaptive mesh refinement in both time and space (see e.g. [3]), in which larger time steps are used on coarser regions of the mesh. When the refinement ratio is an integer, it is necessary to use uniform substeps to enable the substep solutions from meshes with different grid spacing to be synchronized in time. Uniform time-levels also simplify the use of multi-step methods based on a BDF approach (see e.g. [2]) in the prediction step of PIDC methods. This strategy was investigated in [25] as a possible remedy for order reduction in very stiff problems, and a paper which further investigates this potential is in preparation [22].

The phenomenon of order reduction for semi-implicit methods applied to stiff problems has seen only limited investigation to date. As mentioned in the third point above, both PIDC methods and the ARK schemes from [19] display order reduction. Although a qualitatively similar order reduction for PIDC methods is observed for both the van der Pol test problem and the simpler cosine test, it is not clear whether this behavior is completely generic, especially in regard to the differences in behavior between uniform and Gauss points. As just mentioned, the benefits of using semi-implicit BDF type predictors with PIDC methods to remedy order reduction is being more closely investigated [22].

Finally, it is worth mentioning that there are an unlimited number of possibilities regarding the choice of quadrature nodes for PIDC methods which have not been considered here. One could imagine choosing the nodes in order to achieve some desirable property in the overall numerical scheme based on a physical understanding of the true solution. The analysis in this work suggests that such a hypothetical choice would likely not substantially affect the stability or accuracy of the method for mildly stiff problems but could perhaps provide a significant benefit for very stiff problems.

A Appendix

In this Appendix an analytical formulation is developed for the truncation error for a SIPIDC methods applied to a simple stiff equation.

Given a smooth function $p(t)$, consider the ODE with exact solution $y(t) = p(t)$ given by

$$(A.1) \quad \begin{aligned} y' &= p'(t) - \frac{1}{\epsilon}(y - p(t)) \\ y(0) &= p(0). \end{aligned}$$

Here ϵ controls the stiffness with equation becoming more stiff as ϵ goes to zero. The first term in (A.1) will be handled explicitly and the second implicitly. The following analysis applies to the stiff case when $\epsilon \ll \Delta t$.

First consider the computation of a provisional solution $\phi^0(t_m)$ computed using the forward-backward Euler method described by Eq. (2.7). Let $p_m \equiv p(t_m)$ and $\phi_m^0 \equiv \phi^0(t_m)$. Given a previously computed value ϕ_m^0 with error $e_m^0 = p_m - \phi_m^0$, one step of forward-backward Euler applied to Eq. (A.1) is

$$(A.2) \quad \phi_{m+1}^0 = \frac{\phi_m^0 + \Delta t_m(p'_m + \frac{1}{\epsilon}p_{m+1})}{1 + \frac{\Delta t_m}{\epsilon}}.$$

When $\epsilon < \Delta t_m$, the quantity $1/(1 + \frac{\Delta t_m}{\epsilon})$ can be expanded into the series

$$(A.3) \quad \frac{1}{1 + \frac{\Delta t_m}{\epsilon}} = \frac{\epsilon}{\Delta t_m} \left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m} \right)^2 \dots \right).$$

Hence Eq. (A.2) is

$$(A.4) \quad \phi_{m+1}^0 = p_{m+1} + \left(p'_m + \frac{\phi_m^0 - p_{m+1}}{\Delta t_m} \right) \left(\epsilon - \frac{\epsilon^2}{\Delta t_m} + \frac{\epsilon^3}{\Delta t_m^2} \dots \right).$$

Using the fact that $\phi_m^0 + e_m^0 = p_m$ and the Taylor series for $p(t)$, the term in the first parentheses can be expanded to

$$(A.5) \quad \begin{aligned} p'_m + \frac{\phi_m^0 - p_{m+1}}{\Delta t_m} &= p'_m + \frac{-e_m^0 - \Delta t_m p'_m - \frac{\Delta t_m^2}{2} p''_m - \frac{\Delta t_m^3}{6} p'''_m \dots}{\Delta t_m} \\ &= -\left(\frac{e_m^0}{\Delta t_m} + \frac{\Delta t_m p''_m}{2} + \frac{\Delta t_m^2 p'''_m}{6} \right) + \mathcal{O}(\Delta t^3). \end{aligned}$$

Substituting this expression into Eq. (A.4) and recalling $e_{m+1}^0 = \phi_{m+1}^0 - p_{m+1}$ yields

$$\begin{aligned} e_{m+1}^0 &= e_m^0 \left(\frac{\epsilon}{\Delta t_m} - \frac{\epsilon^2}{\Delta t_m^2} + \frac{\epsilon^3}{\Delta t_m^3} \dots \right) \\ &\quad + \left(\frac{p''_m}{2} + \frac{\Delta t_m p'''_m}{6} \right) \left(\epsilon \Delta t_m - \epsilon^2 + \frac{\epsilon^3}{\Delta t_m} + \dots \right) \end{aligned}$$

$$(A.6) \quad +\mathcal{O}(\epsilon\Delta t^2) + \mathcal{O}(\epsilon^2\Delta t) + \mathcal{O}(\epsilon^3) \dots$$

Now consider the correction equation given a provisional solution $\phi^0(t_m)$. Note that $f(\phi^0(t_m), t_m) = p'_m - \frac{1}{\epsilon}e^0(t_m)$. The direct form of the correction equation (2.11) for Eq. (A.1) is

$$(A.7) \quad \begin{aligned} \phi_{m+1}^1 &= \phi_m^1 + \Delta t_m \left(p'_m - p'_m - \frac{1}{\epsilon}(\phi_{m+1}^1 - p_{m+1} - \phi_{m+1}^0 + p_{m+1}) \right) \\ &\quad + I_m^{m+1}(\phi^0) \\ &= \phi_m^1 + \Delta t_m \left(-\frac{1}{\epsilon}(\phi_{m+1}^1 - \phi_{m+1}^0) \right) + I_m^{m+1}(\phi^0). \end{aligned}$$

Solving for ϕ_{m+1}^1 yields

$$(A.8) \quad \phi_{m+1}^1 = \frac{\phi_m^1 + \frac{\Delta t_m}{\epsilon}\phi_{m+1}^0 + I_m^{m+1}(p'(t) - \frac{1}{\epsilon}e^0(t))}{1 + \frac{\Delta t_m}{\epsilon}}.$$

Now consider the quadrature term in the numerator. The integration rule given by Eq. (2.12) defines

$$(A.9) \quad I_m^{m+1} \left(p'(t) - \frac{1}{\epsilon}e^0(t) \right) = \Delta t_m \sum_{l=0}^p q_m^l \left(p'_l - \frac{1}{\epsilon}e_l^0 \right).$$

Since the integration rule is assumed to be $\mathcal{O}(\Delta t^q)$, the first term can be integrated yielding

$$(A.10) \quad I_m^{m+1} \left(p'(t) - \frac{1}{\epsilon}e^0(t) \right) = p_{m+1} - p_m + \mathcal{O}(\Delta t^q) - \frac{\Delta t_m}{\epsilon} \sum_{l=0}^p q_m^l e_l^0.$$

Substituting this expression into Eq. (A.8) gives

$$(A.11) \quad \phi_{m+1}^1 = \frac{\phi_m^1 + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon} (\phi_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0) + \mathcal{O}(\Delta t^q)}{1 + \frac{\Delta t_m}{\epsilon}}.$$

Using the expansion in Eq. (A.3) yields

$$(A.12) \quad \begin{aligned} \phi_{m+1}^1 &= \frac{\epsilon}{\Delta t_m} \left(\phi_m^1 + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon} \left(\phi_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0 \right) + \mathcal{O}(\Delta t^q) \right) \\ &\quad - \left(\frac{\epsilon}{\Delta t_m} \right)^2 \left(\phi_m^1 + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon} \left(\phi_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0 \right) + \mathcal{O}(\Delta t^q) \right) \\ &\quad + \left(\frac{\epsilon}{\Delta t_m} \right)^3 \left(\phi_m^1 + p_{m+1} - p_m + \frac{\Delta t_m}{\epsilon} \left(\phi_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0 \right) + \mathcal{O}(\Delta t^q) \right) \dots \end{aligned}$$

hence

$$\begin{aligned}
\phi_{m+1}^1 &= \phi_{m+1}^0 \left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m} \right)^2 \dots \right) \\
&+ (\phi_m^1 - p_m + p_{m+1}) \left(\frac{\epsilon}{\Delta t_m} - \left(\frac{\epsilon}{\Delta t_m} \right)^2 + \left(\frac{\epsilon}{\Delta t_m} \right)^3 \dots \right) \\
&- \sum_{l=0}^p q_m^l e_l^0 \left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m} \right)^2 \dots \right) \\
\text{(A.13)} \quad &+ \mathcal{O}(\epsilon \Delta t^{q-1}) + \mathcal{O}(\epsilon^2 \Delta t^{q-2}) + \mathcal{O}(\epsilon^3 \Delta t^{q-3}) \dots
\end{aligned}$$

Finally, define the error in the updated solution $e_m^1 = \phi_m^1 - p_m$. Then subtracting p_{m+1} from both sides of the equation yields

$$\begin{aligned}
e_{m+1}^1 &= e_m^1 \left(\frac{\epsilon}{\Delta t_m} - \left(\frac{\epsilon}{\Delta t_m} \right)^2 + \left(\frac{\epsilon}{\Delta t_m} \right)^3 \dots \right) \\
&+ \left(e_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0 \right) \left(1 - \frac{\epsilon}{\Delta t_m} + \left(\frac{\epsilon}{\Delta t_m} \right)^2 \dots \right) \\
\text{(A.14)} \quad &+ \mathcal{O}(\epsilon \Delta t^{q-1}) + \mathcal{O}(\epsilon^2 \Delta t^{q-2}) + \mathcal{O}(\epsilon^3 \Delta t^{q-3}) \dots
\end{aligned}$$

Consider now a full time step of a SIPIDC method for Eq. (A.1). Assume the error at the beginning of the time step is given by e_0^0 . The dominant terms in the provisional solution error (A.6) are

$$\text{(A.15)} \quad e_{m+1}^0 = \frac{p_m''}{2} (\epsilon \Delta t_m - \epsilon^2) + z_m,$$

where

$$\text{(A.16)} \quad z_m = \begin{cases} e_0^0 \frac{\epsilon}{\Delta t_1}, & m = 1, \\ \epsilon^2 \frac{\Delta t_{m-1}}{\Delta t_m} \frac{p_{m-1}''}{2}, & m > 1. \end{cases}$$

Likewise, the dominant piece of the correction equation error in Eq. (A.14) will come from the term

$$\text{(A.17)} \quad e_{m+1}^1 = e_{m+1}^0 - \sum_{l=0}^p q_m^l e_l^0.$$

Substituting the dominant provisional error given in Eq. (A.15) into the dominant correction error formula (A.17) gives

$$\text{(A.18)} \quad e_{m+1}^1 = \frac{p_{m+1}''}{2} (\epsilon \Delta t_{m+1} - \epsilon^2) + z_m - \sum_{l=1}^p q_m^l \left(\frac{p_l''}{2} (\epsilon \Delta t_l - \epsilon^2) + z_l \right).$$

The summation term can be rewritten as

$$\text{(A.19)} \quad \sum_{l=1}^p q_m^l \left(\frac{p_l''}{2} (\epsilon \Delta t_l - \epsilon^2) + z_l \right) = \sum_{l=1}^p q_m^l \epsilon \frac{p_l''}{2} \Delta t_l - \frac{\epsilon^2}{2} \sum_{l=1}^p q_m^l p_l'' + \sum_{l=1}^p q_m^l z_l.$$

The first of the three summations can be rearranged to yield.

$$(A.20) \quad \sum_{l=1}^p q_m^l \epsilon \frac{p_l''}{2} \Delta t_l = \sum_{l=1}^p q_m^l \epsilon \frac{p_{m+1}'' + \mathcal{O}(\Delta t)}{2} \Delta t_l = \frac{p_{m+1}''}{2} \epsilon \sum_{l=1}^p q_m^l \Delta t_l + \mathcal{O}(\epsilon \Delta t^2).$$

The second term gives

$$(A.21) \quad -\frac{\epsilon^2}{2} \sum_{l=1}^p q_m^l p_l'' = -\frac{\epsilon^2}{2 \Delta t_m} (p_{m+1}' - p_m' + \mathcal{O}(\Delta t^q)) = -\frac{\epsilon^2}{2} p_{m+1}'' + \mathcal{O}(\epsilon^2 \Delta t).$$

Therefore substituting these expression into Eq. (A.18) and simplifying gives

$$(A.22) \quad e_{m+1}^1 = \frac{\epsilon p_{m+1}''}{2} \left(\Delta t_{m+1} - \sum_{l=1}^p q_m^l \Delta t_l \right) + \sum_{l=1}^p q_m^l z_l + \mathcal{O}(\Delta t^q) + \mathcal{O}(\epsilon \Delta t^2) + \mathcal{O}(\epsilon^2 \Delta t)$$

Rearranging slightly the error takes the form

$$(A.23) \quad e_{m+1}^1 = \epsilon \Delta t_m \frac{p_{m+1}''}{2} \left(1 - \sum_{l=1}^p q_m^l \frac{\Delta t_l}{\Delta t_m} \right) + z_m + \sum_{l=1}^p q_m^l z_l + \mathcal{O}(\Delta t^q) + \mathcal{O}(\epsilon \Delta t^2) + \mathcal{O}(\epsilon^2 \Delta t).$$

Note for uniform nodes where all the sub-steps are equal,

$$(A.24) \quad \sum_{l=1}^p q_m^l \frac{\Delta t_l}{\Delta t_m} = \sum_{l=1}^p q_m^l \frac{\Delta t_m}{\Delta t_m} = \sum_{l=1}^p q_m^l = 1$$

and (A.23) simplifies to

$$(A.25) \quad e_{m+1}^1 = z_m + \sum_{l=1}^p q_m^l z_l + \mathcal{O}(\Delta t^q) + \mathcal{O}(\epsilon \Delta t^2) + \mathcal{O}(\epsilon^2 \Delta t).$$

For quadrature rules based on non-uniform quadrature nodes, the $\mathcal{O}(\epsilon \Delta t)$ error term remains.

REFERENCES

1. Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.*, 25:151–167, 1997.
2. Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton. Implicit-explicit methods for time-dependent PDE's. *SIAM J. Numer. Anal.*, 32:797–823, 1995.
3. M. J. Berger and J. Olinger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

4. K. Böhmer, P. Hemker, and H. J. Stetter. The defect correction approach. In K. Böhmer and H. J. Stetter, editors, *Defect Correction Methods. Theory and Applications*, pages 1–32. Springer-Verlag, 1984.
5. A. Bourlioux, A. T. Layton, and M. L. Minion. Higher-order multi-implicit spectral deferred correction methods for problems of reacting flow. *J. Comput. Phys.*, 189:351–376, 2003.
6. M. P. Calvo, J. de Frutos, and J. Novo. Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations. *Appl. Numer. Math.*, 37:535–549, 2001.
7. H.W. Cheng, J. Huang, and T. Leiterman. An adaptive fast solver for the modified helmholtz equation in two dimensions. *J. Comput. Phys.*, 2004. submitted.
8. G. Dahlquist. A special stability problem for linear multistep methods. *BIT*, 3:27–43, 1963.
9. J. W. Daniel, V. Pereyra, and Larry L. Schumaker. Iterated deferred corrections for initial value problems. *Acta Cient. Venezolana*, 19:128–135, 1968.
10. Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT*, 40(2):241–266, 2000.
11. M. Roche E. Hairer, Ch. Lubich. Error of Runge-Kutta methods for stiff problems studied via differential algebraic equations. *BIT*, 28(3):678–700, 1988.
12. B. L. Ehle. On pade approximations to the exponential function and A-stable methods for the numerical solution of initial value problems. Dept. Applied Analysis and Computer Sci., Research Rpt. CSR 2010, University of Waterloo, 1969.
13. J. Frank, W. H. Hundsdorfer, and J. G. Verwer. Stability of implicit-explicit linear multistep methods. *Appl. Numer. Math.*, 25:193–205, 1997.
14. R. Frank and C. W. Ueberhuber. Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. *BIT*, 17:146–159, 1977.
15. Bertil Gustafsson and Lin Hemmingsson-Fränden. Deferred correction in space and time. *J. Sci. Comput.*, 17(1–4):541–550, 2002.
16. Bertil Gustafsson and Wendy Kress. Deferred correction methods for initial value problems. *BIT*, 41:986–995, 2001.
17. Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, 1991.
18. Jenni Hudson. Spectral deferred correction methods for ode initial value problems. Master’s thesis, University of North Carolina, 2004.
19. Christopher A. Kennedy and Mark H. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Appl. Numer. Math.*, 44:139–181, 2003.

20. Wendy Kress and Bertil Gustafsson. Deferred correction methods for initial boundary value problems. *J. Sci. Comput.*, 17(1-4):241–251, 2002.
21. A. T. Layton and M. L. Minion. Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. *J. Comput. Phys.*, 194(2):697–714, 2004.
22. A. T. Layton and M. L. Minion. Implications of the choice of predictors and correctors for semi-implicit Picard integral deferred corrections methods. page in preparation, 2004.
23. M. L. Minion. Higher-order semi-implicit projection methods. In M. Hafez, editor, *Numerical Simulations of Incompressible Flows. Papers from the workshop held in Half Moon Bay, CA, June 19-21, 2001*, pages 126–140, River Edge, NJ, January 2003. World Scientific Publishing.
24. M. L. Minion. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. *Appl. Numer. Math.*, 48(3-4):369–387, 2004.
25. Michael L. Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Comm. Math. Sci.*, 1:471–500, 2003.
26. R. D. Skeel. A theoretical framework for proving accuracy results for deferred corrections. *SIAM J. Numer. Anal.*, 19(1):171–196, 1981.
27. O. B. Widlund. A note on unconditionally stable linear multistep methods. *BIT*, 7:65–70, 1967.
28. P.E. Zadunaisky. A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations. In G. Contopoulos, editor, *The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings of International Astronomical Union, Symposium 25*, 1964.