INTEGRAL DEFERRED CORRECTION METHODS FOR SCIENTIFIC COMPUTING

By

Maureen Marilla Morton

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Applied Mathematics

2010

ABSTRACT

INTEGRAL DEFERRED CORRECTION METHODS FOR SCIENTIFIC COMPUTING

By

Maureen Marilla Morton

Since high order numerical methods frequently can attain accurate solutions more efficiently than low order methods, we develop and analyze new high order numerical integrators for the time discretization of ordinary and partial differential equations. Our novel methods address some of the issues surrounding high order numerical time integration, such as the difficulty of many popular methods' construction and handling the effects of disparate behaviors produce by different terms in the equations to be solved.

We are motivated by the simplicity of how Deferred Correction (DC) methods achieve high order accuracy [72, 27]. DC methods are numerical time integrators that, rather than calculating tedious coefficients for order conditions, instead construct high order accurate solutions by iteratively improving a low order preliminary numerical solution. With each iteration, an error equation is solved, the error decreases, and the order of accuracy increases. Later, DC methods were adjusted to include an integral formulation of the residual, which stabilizes the method. These Spectral Deferred Correction (SDC) methods [25] motivated Integral Deferred Corrections (IDC) methods. Typically, SDC methods are limited to increasing the order of accuracy by one with each iteration due to smoothness properties imposed by the gridspacing. However, under mild assumptions, explicit IDC methods allow for any explicit rth order Runge-Kutta (RK) method to be used within each iteration, and then an order of accuracy increase of r is attained after each iteration [18]. We extend these results to the construction of implicit IDC methods that use implicit RK methods, and we prove analogous results for order of convergence.

One means of solving equations with disparate parts is by semi-implicit integrators, handling a "fast" part implicitly and a "slow" part explicitly. We incorporate additive RK (ARK) integrators into the iterations of IDC methods in order to construct new arbitrary order semi-implicit methods, which we denote IDC-ARK methods. Under mild assumptions, we rigorously establish the order of accuracy, finding that using any rth order ARK method within each iteration gives an order of accuracy increase of r after each iteration [15]. We apply IDC-ARK methods to several numerical examples and present preliminary results for adaptive timestepping with IDC-ARK methods.

Another means of solving equations with disparate parts is by operator splitting methods. We construct high order splitting methods by employing low order splitting methods within each IDC iteration. We analyze the efficiency of our split IDC methods as compared to high order split methods in [77] and also note that our construction is less tedious. Conservation of mass is proved for split IDC methods with semi-Lagrangian WENO reconstruction applied to the Vlasov–Poisson system. We include numerical results for the application of split IDC methods to constant advection, rotating, and classic plasma physics problems.

This is a preliminary, yet significant, step in the development of simple, high order numerical integrators that are designed for solving differential equations that display disparate behaviors. Our results could extend naturally to an asymptotic preserving setting or to other operator splittings.

DEDICATION

To my father, in loving memory.

ACKNOWLEDGMENT

If I were to mention everyone who contributed to my success, the list would be longer than my dissertation. Within the limitations of space, I wish to acknowledge a few of these amazing people.

The first thanks go to my advisor, Dr. Andrew Christlieb, who has infinite faith in the capabilities of his students, and whose support of me is not only academically excellent, but also extraordinarily benevolent and personal. I also am indebted to the members of our research team and our collaborators, in particular to Drs. Benjamin Ong and Jing–Mei Qiu, who have provided valuable insight to my research and indispensable friendship. I wish to thank my comprehensive and defense committee members, Drs. G. Bao, D. Liu, K. Promislow, J. Qian, Y. Wang, and Z. Zhou, for their wisdom, help, and general good humor. Everyone knows that the secretaries and the computer technicians are the most important members of any department, so I cannot fail to thank them as well for making all the details go smoothly. My fellow graduate students were tremendously instrumental in sustaining me through the entire process, and I am particularly grateful for my Chinese friends, who allowed me the privilege of sharing in their *yǒuyì*.

Most of all, I thank my family: my mother, for teaching me to work hard, to be independent, and how to understand the academic culture; my father, for teaching me that relationships are more important than achievements and for inspiring me with his own unrealized dream of going to college (to become a math teacher); and my brother and sisters, for their delightful encouragement despite their not-so-secret belief that I am insane for studying so much. I am ever grateful to Jesus, my God, for sustaining me here at MSU, just as he promised, and without whose help I would have failed, "For the Lord gives wisdom, and from his mouth come knowledge and understanding" (Proverbs 2:6, NIV).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

Throughout scientific history, experiments, physical and mathematical models, and, more recently, scientific computing have been partners in generating new knowledge. Although all are vital components of discovery, we focus on the role that scientific computing plays. In many cases, experiments may not be feasible due to expense or physical constraints, and models may be too complex to find analytic solutions. In these situations, further insight may be found through computationally approximating solutions to models, such as those models given by ordinary differential equations (ODEs), partial differential equations (PDEs), differential algebraic equations (DAEs), or others. The daily changes in technology and scientific discoveries means that the need for innovative numerical methods always exists. In particular, it is desirable to obtain accurate solutions at a low computational cost and in real time. Frequently, higher order numerical methods can attain such accuracy more efficiently than lower order methods, although care must be taken to maintain certain properties; e.g., for some physical problems, one must be careful that a numerical method conserves mass when the physical theory asserts that mass is preserved. In this dissertation, we focus on higher order numerical integrators for the time discretization of ODEs and PDEs.

Some of the issues surrounding high order numerical time integration include: difficulty of the method's construction (for example, the order conditions for constructing higher order Runge-Kutta methods and the number of coefficients one

must find for some general high order splitting methods increase exponentially with the order of the method [37, 77]); the equations exhibit multi-scale behaviors and/or different operators and terms in the equations produce disparate but coupled (frequently nonlinear) behaviors, leading to severe timestep restrictions or difficulties in numerical implementation; and treatment of boundary conditions at high order. In this dissertation, we contribute to the work that tackles some of these issues by further developing Integral Deferred Correction (IDC) methods, described below.

One way to overcome the first issue of a complicated construction of high order methods was introduced and later expanded upon in articles such as [78, 65, 28, 27, 72]. This class of methods, called Deferred Correction (DC) methods (or Defect Correction), involves taking a prediction step to form an approximate solution using a numerical method of choice, then iteratively forming an error equation, solving for the error using any numerical method of choice, and updating the approximate solution. The main principle is that, at each iteration, the order of accuracy of the DC method increases by the order of the numerical method used at each iteration. The prediction step can be considered one iteration, and each step that solves the error equation (also called the correction step) can be considered one additional iteration. The beauty of DC methods is their simplicity in constructing higher order methods. One only needs to use low order integrators within the prediction and correction steps, and the iterative process attains arbitrary high order without needing to consider complicated order conditions. A key component of DC methods is the treatment of the residual within the error equation. For simplicity, suppose we are solving an initial value problem (IVP)

$$y'(t) = f(t, y), \quad y(0) = y_0. \tag{1}$$

The "differential form" of the residual, r, is

$$r(t) = \eta'(t) - f(t, \eta(t)),$$

where η is a provisional numerical solution to (1). Approximating the derivative, $\eta'(t)$, within the DC correction step is numerically unstable (see, e.g., [32]). More recently, Greengard and Rokhlin introduced a variant of DC methods, called Spectral Deferred Correction (SDC) methods, by incorporating the integral formulation of the residual,

$$\int_0^t r(\tau)d\tau = \eta(t) - \eta(0) - \int_0^t f(\tau, \eta(\tau))d\tau,$$

to stabilize the correction step [25]. This development sparked renewed interest in DC methods. Several methods influenced by SDC methods include Krylov Deferred Correction methods, applied to ODEs and DAEs [41, 42]; semi-implicit and multi-implicit SDC methods (SISDC and MISDC, respectively) applied to ODEs whose right hand sides include stiff terms (i.e., terms with disparate sizes) [60, 47, 9]; and Integral Deferred Correction (IDC) methods, applied to ODEs and PDEs, with a special choice of nodes and integrators used at each step [18, 17], including adjustments to allow parallel in time implementation of IDC methods [16].

Typically, SDC methods are limited to increasing the order of accuracy by one after each correction step due to certain smoothness properties that are imposed by the gridspacing. Some works that adjusted SDC methods to incorporate integrators other than Euler methods in the prediction and correction loops include [48, 47, 18, 17]. In particular, [18, 17] present IDC methods that allow for any explicit RK methods of arbitrary order to be used within the prediction and correction loops, and that allow for an order of accuracy increase greater than one after each correction

loop. They rigorously established under certain assumptions on the gridspacing and the smoothness of the exact solution that, when an rth order explicit RK method is used to predict and correct the numerical solution, the order of accuracy of the IDC method increases by r for each prediction and correction loop. Thus fewer correction loops are required to attain a certain order of accuracy, when compared to SDC methods. They also found that such high order IDC methods' stability and efficiency compared favorably to RK methods alone. In this dissertation, we extend the construction and theory of IDC methods using explicit RK integrators to the construction of arbitrary order IDC methods that use implicit RK integrators and a rigorous presentation of analogous results to the order of convergence theory.

The second issue of high order numerical integrators, that of handling equations with distinct parts, multi-scale, and/or nonlinear behaviors, is quite broad. We focus on methods that handle different terms in the equations via separate means. First consider semi-implicit, or implicit-explicit, integrators. Typically, a semi-implicit integrator is intended to solve an equation of the form

$$y'(t) = F(t, y) + G(t, y), \tag{2}$$

with appropriate initial or boundary conditions. For simplicity of presentation, we suppress any non-time dependence that may be in F and G, which could be functions or operators. Suppose F contains only nonstiff (not particularly large) terms and G contains any stiff (could be large) terms. Semi-implicit integrators can be used to solve the stiff part implicitly and the nonstiff part explicitly, thereby gaining the benefit of the implicit method for the stiffness but potentially saving computational effort by handling some terms explicitly. Popular integrators include additive Runge-Kutta (ARK) methods and integrators constructed from linear multi-step methods, such as Adams or BDF methods. [60, 47] introduce semi-implicit SDC methods that

may attain orders of accuracy higher than ARK or BDF methods, but in [60], the increase in accuracy at each correction loop is limited to 1st order, while [47] only presents specific examples of a 2nd order ARK and a 2nd order BDF method that may be successfully used in the correction loops. In this dissertation, we expand the results in [47, 18], and our IDC with implicit RK methods to construct a general formulation that clearly presents the ability to incorporate any order ARK scheme into the IDC framework. We denote the new construction as IDC-ARK methods. We rigorously establish under certain assumptions that, when an rth order ARK method is used to predict and correct the numerical solution, the order of accuracy of the IDC method increases by r for each prediction and correction loop. We also include numerical examples of IDC-ARK methods applied to IVPs that substantiate the theory and to an advection-diffusion equation, verifying that the semi-implicit structure allows for an improved CFL condition. To further handle multi-scale situations, especially equations whose solutions contain initial, inner, or boundary layers, we also present preliminary results for adaptive implementation of IDC-ARK methods. Since IDC methods form an approximation to the error at each time step, they fit naturally into an adaptive setting, where the size of the time step is adjusted as necessary according to some estimation of the size of the error at that step.

Equations of the form (2) may also be solved via operator splitting methods. Splitting methods may be useful when, for example, y depends on more than one spatial variable, say on (x_1, x_2), in addition to time, t, but F depends only on x_1 and G depends only on x_2. In this case, we may approximate the solution by successively solving the following two subproblems,

$$\partial_t y = F, \quad \partial_t y = G,$$

which now are simple one-dimensional problems rather than a more complicated

two-dimensional problem. Sometimes also an operator splitting may be chosen such that the subproblems are linear although the original problem (2) is nonlinear. Many high order splitting methods are costly both to construct and to implement [77], although some reasonable methods have been developed in [49, 50], including high order splitting methods constructed via DC methods and applied to Vlasov-Maxwell systems. We wished to seek an improved high order splitting method by using the IDC framework instead. We present these novel split IDC methods in this dissertation. They incorporate simple first order splitting methods and second order Strang splitting methods within the prediction and correction loops to obtain arbitrary order splitting methods. The construction is specified for application to the Vlasov-Poisson equations, which are frequently used to describe the behavior of particles in some plasma physics settings. We show that a new formulation of the error equation in the correction step is required for proper application to the Vlasov-Poisson system. As required by the physical theory, we also prove that split IDC methods for Vlasov equations conserve mass when conservative semi-Lagrangian WENO reconstruction is employed in conjunction with the IDC splitting method.

The third issue that causes difficulties in numerical time integration, treating boundary conditions (BCs) in a way that maintains the high order of the numerical method, will not be discussed in this dissertation. However, this area is essential if the discussed methods are to be applied to realistic problems. In particular, this field is mainly unexplored for deferred correction methods. The application of split DC methods to the Vlasov-Maxwell system in [49, 50] only utilizes periodic BCs. Essentially, in this dissertation, we also only apply IDC methods to differential equations with periodic BCs. Motivated by the high order treatment of BCs presented by LeVeque in [52, 53], we expect that equivalent or alternative means of handling BCs could be designed to take advantage of IDC's unique framework of extending low order methods to high order methods. This design may circumvent the extensive

approximations of higher order derivatives that are necessary in LeVeque's work.

Chapter 1 briefly introduces several equations that describe certain physical processes that motivate some of the numerical issues we wish to address via IDC methods. Chapter 2 presents a history and literature review of DC methods and more recent developments. It also includes a description of our new arbitrary order IDC methods that incorporate implicit RK methods in their construction and a rigorous proof of the theoretical order results for such IDC methods. Chapter 3 lays out the construction of our new semi-implicit IDC-ARK methods, order of accuracy theoretical results, how the semi-implicit proof differs from the implicit proof in Chapter 2, and application to numerical examples such as Van der Pol's oscillator, an initial layer problem, and an advection-diffusion equation. Chapter 4 is a short study on the implementation of IDC methods in an adaptive setting, with some preliminary results for adaptive IDC-ARK methods. Chapter 5 shows the construction of new high order split IDC methods, an analysis of efficiency as compared to high order splitting methods in [77], a proof of conservation of mass, and numerical results for its application to constant advection, rotating, and Vlasov-Poisson equations. Chapter 6 suggests directions for future research, such as extending asymptotic preserving methods to the IDC framework.

Chapter 1

Physical Motivation

Many physical problems involve both fast and slow events, or can be modeled by equations that have operators whose numerical solutions can be simplified by operator splitting. Modeling these situations may result in equations that contain widely separated time scales or equations whose parts may be computed separately. These equations include but are not limited to: chemical rate equations, convection-diffusion equations, Vlasov-Poisson equations, and hyperbolic conservation laws with stiff relaxation. We present some of these problems as motivation for studying semi–implicit numerical integrators that handle two different time scales and splitting methods that allow for simpler numerical computations, although we have not necessarily tested all the problems described below.

1.1 Chemical Rate Equations

Chemical reactions are often modeled by rate equations that are formed by using the mass action law. An example of a system of rate equations is given by the ROBER

problem, which describes an autocatalytic reaction [59]. The system is

$$\frac{d}{dt}y_1 = -0.04y_1 + 10^4 y_2 y_3, \tag{1.1}$$

$$\frac{d}{dt}y_2 = 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2, \tag{1.2}$$

$$\frac{d}{dt}y_3 = 3 \cdot 10^7 y_2^2, \tag{1.3}$$

$$y_1(0) = 1, \quad y_2(0) = 0, \quad y_3(0) = 0, \tag{1.4}$$

where the y_i represent the concentrations of the chemical species involved, and the coefficients on the right hand sides are the rate constants, which describe the rate of certain component reactions. The widely disparate rate constants contribute to the stiff nature of this problem, and the solution to this system changes quickly initially (although it varies smoothly as time progresses). Typically, there are far more species involved in a chemical reaction, thus rendering the system of rate equations much larger than the ROBER problem; e.g., the chemical Akzo Nobel problem or the pollution problem, both also in [59]. However, the ROBER problem sufficiently demonstrates that widely separated time scales may arise in chemical rate equations.

1.2 Convection-Diffusion-Reaction Equations

Multiscale effects also play a role in the solution of convection-diffusion-reaction equations. Typically, the diffusion term dominates the other terms due to the discretization of the spatial derivatives, and the reaction term is significantly smaller

than the other terms. An example of such an equation is given in [45], in the form

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho u \\ \rho e_0 \\ \rho Y_i \end{bmatrix} = -\frac{\partial}{\partial x}\begin{bmatrix} \rho u \\ \rho u^2 + p \\ (\rho e_0 + p)u \\ \rho u Y_i \end{bmatrix} \tag{1.5}$$

$$+ \begin{bmatrix} 0 \\ \frac{4\mu}{3}\frac{\partial^2 u}{\partial x^2} \\ \lambda\frac{\partial^2 T}{\partial x^2} + \frac{4\mu}{3}\left(\frac{\partial u}{\partial x}\right)^2 + \frac{\partial}{\partial x}\left(\sum_{i=1}^{ncs} \rho D_i h_i \frac{\partial Y_i}{\partial x}\right) \\ \rho D_i \frac{\partial^2 Y_i}{\partial x^2} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\omega}_i \end{bmatrix}. \tag{1.6}$$

This system is a one-dimensional version of the simplified, gas-phase, multicomponent, compressible NavierStokes equations with chemical reaction. $(\mu, \lambda, \rho D_i)$ are the transport coefficients, where μ is molecular viscosity, λ is thermal conductivity, ρ is fluid density, and D_i is the effective Fickian diffusion coefficient. The species is designated by $i = 1, \ldots, ncs$, where ncs is the number of chemical species, the fluid velocity is represented by u, T is temperature, Y_i are the species mass fraction, p is pressure, e_0 is total specific internal energy, h is the partial specific enthalpy of species i, ω_i is the reaction rate of species i. The first term on the right hand side is the convection term, the second term is the diffusion term, and the last term is the reaction term. In addition to the differences in sizes of the terms that could arise based on the spatial derivatives, there are also potential separations of time scales based on the sizes of the transport coefficients and the reaction rates.

1.3 Vlasov–Poisson Equations

Many natural and industrial processes, such as solar weather and integrated circuits manufacturing, involve plasmas. A plasma is sometimes called the fourth state of matter, since adding sufficient energy to a solid produces a liquid, adding energy to a liquid leads to a gas, and increasing the energy yet again results in a plasma. A plasma can also be defined as an electrically neutral collection of charged particles. The sizes, speeds of, and forces acting on or by these charged particles may vary by several orders of magnitude. Hence it is of interest to consider computational issues in models of plasmas. As a special case, the electron motion in a one-dimensional cold plasma with a stationary ion background may be modeled by the following Vlasov-Poisson equations, where the physical constants have been normalized to one.

$$\partial_t f + v\partial_x f - E\partial_v f = 0, \tag{1.7}$$

$$E = -\partial_x \phi, \tag{1.8}$$

$$-\partial_x^2 \phi = \rho, \tag{1.9}$$

where $f(x, v, t)$ is the electron probability density function, x is space, v is the velocity, t is time, E is the electric field, ϕ is the potential, and ρ is the charge density [54]. If we choose to rewrite (1.7) in a Lagrangian framework, from the point of view of an electron within the flow (as opposed to an Eulerian framework, which has a fixed spatial grid), and discretize the problem in (x, v)-phase space, we obtain N ODE systems with varying stiffnesses:

$$x_i'(t) = v_i(t), \tag{1.10}$$

$$v_i'(t) = E(t, x_i), \quad i = 1, \ldots, N, \tag{1.11}$$

where the electric field on the right hand side of (1.11) contains contributions from the electrons, ions, and free space [14]. The velocities and forces may vary significantly in magnitude between different values of i and among the terms on the right hand side of (1.11) for each i. Alternatively, one may use a semi-Lagrangian framework to solve (1.7) for the distribution, in which case splitting methods can be applied. Further details are presented in Chapter 5.

1.4 Hyperbolic Conservation Laws

The Vlasov equation is one type of a class of problems known as hyperbolic conservation laws. Much of the following information about such problems is compiled from [66, 51, 70]. For simplicity of explanation, we explain hyperbolic conservation laws via the general scalar one-dimensional case, solving

$$u_t + f(u)_x = 0, \tag{1.12}$$

where $f(u)$ is the flux. Corresponding to the physics involved, such an equation satisfies conservation of mass. Consider the case where $u(x,t) = \rho(x,t)$, the density, with flux $f(\rho(x,t))$, over the interval $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$, where the flow is to the right:

$$\begin{array}{ccccc} & f(\rho(x,t)) & \rho(x,t) & f(\rho(x,t)) & \\ & \rightarrow & & \rightarrow & \\ \hline | & & & | & \\ x_{i-\frac{1}{2}} & & \smile & x_{i+\frac{1}{2}} & \\ & & I_i & & \end{array} \tag{1.13}$$

The integral form of conservation of mass over the interval I_i is:

$$\text{mass at } t^{n+1} = \int_{I_i} \rho(x, t^{n+1}) dx \tag{1.14}$$

$$= \int_{I_i} \rho(x, t^n) dx + \int_{t^n}^{t^{n+1}} f(\rho(x_{i-\frac{1}{2}}, t)) - f(\rho(x_{i+\frac{1}{2}}, t)) dt \tag{1.15}$$

$$= \text{mass at } t_n + \text{flow in} - \text{flow out.} \tag{1.16}$$

We sketch how the integral form (1.14) leads to the differential form of the conservation law. Looking at a small piece in space δx and in time δt, we have

$$\left(\rho(x, t^{n+1}) - \rho(x, t^n)\right) \delta x = -\left(f(\rho(x_{i+\frac{1}{2}}, t)) - f(\rho(x_{i-\frac{1}{2}}, t))\right) \delta t, \tag{1.17}$$

which, upon rearranging, gives

$$\frac{\rho(x, t^{n+1}) - \rho(x, t^n)}{\delta t} = -\frac{1}{\delta x}\left(f(\rho(x_{i+\frac{1}{2}}, t)) - f(\rho(x_{i-\frac{1}{2}}, t))\right). \tag{1.18}$$

From this equation, we deduce the differential form of the conservation law

$$\rho_t + f(\rho)_x = 0, \tag{1.19}$$

which in a more general case is given by (1.12). In a physical setting, the integral form (1.14) is more natural and meaningful.

An understanding of some basic analytic properties is needed for considering a numerical solution. We describe the solutions of two types of equations in terms of characteristics, the lines along which solutions are constant. First we consider a linear problem, such as the linear advection-diffusion equation,

$$u_t + u_x = 0, \quad u(x, 0) = u_0(x). \tag{1.20}$$

The solution is $u(x,t) = u_0(x-t)$. This solution is constant along characteristics described by $\frac{dx}{dt} = 1$, since

$$\frac{du}{dt} = (\frac{\partial}{\partial t} + \frac{dx}{dt}\frac{\partial}{\partial x})u = (\frac{\partial}{\partial t} + \frac{\partial}{\partial x})u = 0. \tag{1.21}$$

In this situation, the characteristics will not cross. For a nonlinear problem, the situation is much more complicated, and characteristics may cross after a certain time, resulting in a discontinuity of the solution. As an example, we consider Burger's equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \quad u(x,0) = u_0(x). \tag{1.22}$$

The solution is constant ($u = u_0$) along characteristics described by $\frac{dx}{dt} = u$, but since the slope of the characteristics is determined by $\frac{1}{u}$, then in many cases, the characteristics will cross after a certain time; e.g., when $u_0(x) = \sin x$. Where the characteristics come together, a shock forms. This discontinuity means the differential form of the conservation law does not work, so the integral form is needed. Only a classical strong solution works for the differential equation, but the integral equation can be solved by a weak solution. A weak solution is not unique, but a weak solution satisfying a property called entropy is called the entropy solution, which is unique. For example, in the Riemann problem, the initial conditions are given by

$$u_0(x) = \begin{cases} u_l, & x < x^* \\ u_r, & x > x^* \end{cases}, \tag{1.23}$$

where u_l and u_r are constant functions. If the entropy condition is satisfied, then $u_l > u_r$ means the solution will form a shock, and $u_l < u_r$ means the solution will form a rarefaction.

The numerical solution of hyperbolic conservation laws is described in Section 5.3, with an emphasis on a numerical method called WENO, which is designed to solve problems with piecewise smooth solutions that contain discontinuities.

Sometimes hyperbolic conservation laws have a stiff relaxation term $\mathcal{R}(u)$, such as diffusion, on the right hand side:

$$\partial_t u + \partial_x f(u) = \mathcal{R}(u), \tag{1.24}$$

where $\mathcal{R}(u)$ is quite large, thus contributing to the multiscale nature of the problem. This type of term shows up in models such as shallow water, granular gas, and traffic flow equations [64].

Chapter 2

Deferred Correction Methods

2.1 Defect Correction Methods

Deferred Correction (DC) methods were developed in the 1960s and 1970s. They can be considered methods to solve a system of ODEs (that may arise within an algorithm solving certain PDEs), or in the more general sense, to solve an equation of the form $Fy = 0$. Our current interest lies in the ODE and PDE application.

2.1.1 DC Algorithm for an IVP

In particular, suppose we are solving the IVP

$$\begin{cases} y'(t) &= f(t,y), \quad t \in [0,T], \\ y(0) &= y_0. \end{cases} \tag{2.1}$$

DC (sometimes called Iterated Defect Correction) involves taking a prediction step to form an approximate solution via an arbitrary numerical method (e.g. RK), then iteratively forming an error equation, solving for the error, and updating the approximate solution. It is of interest to note that the approximate solution approaches

the collocation solution to the IVP as the error is improved at each step [28].

We solve the IVP (2.1) on a grid

$$0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{2.2}$$

where $H = t_{n+1} - t_n$. Each subinterval $[t_n, t_{n+1}]$ is discretized again into M subintervals

$$t_n = t_{n,0} < t_{n,0} < \cdots < t_{n,m} < \cdots < t_{n,M} = t_{n+1}, \tag{2.3}$$

M is fixed. We consider the DC method on one subinterval $[t_n, t_{n+1}]$ and drop the subscript n. Without loss of generality, we look at the first subinterval, $[0, H]$.

The DC algorithm is as follows [28]:

1. **Prediction step:** compute an approximate solution to (2.1) over the grid points (2.3),

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}), \tag{2.4}$$

which is an approximation to the exact solution

$$\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M). \tag{2.5}$$

For example, applying a first order Backward Euler method to (2.1) gives

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h\, f(t_{m+1}, \eta_{m+1}^{[0]}). \tag{2.6}$$

2. **Correction step:** Then interpolate $\eta^{[0]}$ by the Mth degree polynomial satisfying

$$\eta^{(0)}(t_m) = \eta_{\mathbf{m}}{}^{[0]}, \quad m = 0, 1, \ldots, M. \tag{2.7}$$

Form the residual (originally called the "defect" in this context).

$$d^{(0)}(t) = (\eta^{(0)})'(t) - f(t, \eta^{(0)}(t)) \quad t \in [0, H]. \tag{2.8}$$

Add the residual $d^{(0)}(t)$ to the RHS of the original IVP (2.1) to obtain a new IVP whose exact solution is $\eta^{(0)}(t)$.

$$\begin{cases} y'(t) &= f(t,y) + d^{(0)}(t), \quad t \in [0,T], \\ &= f(t,y) + (\eta^{(0)})'(t) - f(t, \eta^{(0)}(t)) \\ y(0) &= y_0. \end{cases} \tag{2.9}$$

Solving the new IVP (2.9) by the same method as for (2.1), we obtain the numerical solution $\pi^{[0]} = \pi_0^{[0]}, \ldots, \pi_M^{[0]}$. Then we use the known exact discretization errors, $\pi_m^{[0]} - \eta^{(0)}(t_m)$, of (2.9) as an estimate for the unknown errors $\eta_m^{[0]} - y(t_m)$ of (2.1).

Then the identity

$$y(t_m) = \eta_m^{[0]} - (\eta_m^{[0]} - y(t_m)) \quad m = 0, 1, \ldots, M \tag{2.10}$$

is approximated by

$$\eta_m^{[1]} \doteq \eta_m^{[0]} - (\pi_m^{[0]} - \eta^{(0)}(t_m)) \quad m = 0, 1, \ldots, M. \tag{2.11}$$

The process is then iterated so that we have

$$\eta_m^{[k+1]} \doteq \eta_m^{[0]} - (\pi_m^{[k]} - \eta^{(k)}(t_m)) \quad m = 0, 1, \ldots, M; \quad k = 1, 2, 3, \ldots, \tag{2.12}$$

where $\eta^{(k)}$ interpolates $\eta^{[k]} = (\eta_0^{[k]}, \eta_1^{[k]}, \ldots, \eta_m^{[k]}, \ldots, \eta_M^{[k]})$.

Note that the error estimation method was suggested by Zadunaisky in [78], where he also references earlier papers where he tried this method, but the iterative procedure was due to others, e.g. Pereyra, Frank and Ueberhuber, Stetter [65, 28, 27, 72].

2.1.2 DC Algorithm for a General Setting

In a more general setting ([27], [72]), suppose we are solving

$$Fy = 0 \tag{2.13}$$

for the solution y, and F is an operator described below. Then the common structural principle of DC methods is error estimatation of a discretization method plus iterative improvement of the error estimate, the advantage being that DC proceeds on the original grid from the first discretization. Stetter first describes a DC method on general normed linear spaces then proceeds to project the problem onto discrete spaces and other spaces, e.g. of polynomial interpolants, so that we see the above case of an ODE on a discrete grid is exactly an example of the general case of DC methods.

For the basic principle of DC methods, one considers a nonlinear bijective operator F such that 0 belongs to the range of F. The goal is to find an approximation to the unique solution y^* of (2.13):

1. Use the approximate inverse $\widetilde{G}$ (or the exact inverse of or the solution operator of the approximate problem $\widetilde{F}y = 0$, $\widetilde{F} \approx F$) to form an initial approximation $y_0 \doteq \widetilde{G}0$.

2. Form the residual/defect $d_0 \doteq Fy_0$.

3. Compute an approximate solution of $Fy = d_0$ (the "neighboring problem"),

which is $\bar{y}_0 \doteq \widetilde{G} d_0$.

The exact solution to the neighbor problem is known to be y_0, so the exact error of the neighbor problem is known: $\bar{y}_0 - y_0$. However, the exact error of the original problem is unknown: $y_0 - y^*$. The error of the neighbor problem is used to estimate the error of the original problem, and thus $y^* \approx y_0 - (\bar{y}_0 - y_0)$ gives an improved approximation y_1 to y^*:

$$y_1 = y_0 - (\bar{y}_0 - y_0). \tag{2.14}$$

The process may be repeated, for $i = 0, 1, 2, \ldots$:

$$d_i \doteq F y_i, \tag{2.15}$$

$$\bar{y}_i \doteq \widetilde{G} F y_i = \widetilde{G} d_i \tag{2.16}$$

$$y_{i+1} = y_i - (\bar{y}_{i+1} - y_0) = (I - \widetilde{G}F) y_i + y_0 = y_i - \widetilde{G} d_i + y_0. \tag{2.17}$$

Note that y^* is the solution of (2.13) and a fixed point of (2.17). Thus convergence occurs if $I - \widetilde{G}F$ is a contraction.

When limited to a numerical method on a discrete grid, this general form of a DC method changes somewhat. A DC method on the discrete space results in an iterate ξ_i that approaches a fixed point ξ^*. In turn, the fixed point ξ^* should be reasonably close to the fixed point y^* of (2.17) on the original space (of course, appropriate conditions are required). Typically ξ^* is some collocation solution which in fact should be close to the projection of y^* onto the discrete space.

2.1.3 Differential Formulation of the DC Algorithm

The differential formulation of a DC method is a rewriting of the special case of application to a system of ODEs [25, 18], or of the time portion of certain PDEs. As above, consider the IVP (2.1) on the interval $[0, H]$, subdivided as in (2.3).

1. **Prediction step:** compute an approximate solution to (2.1) over the grid points (2.3),

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}), \tag{2.18}$$

which is an r_0^{th} order approximation to the exact solution

$$\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M). \tag{2.19}$$

For example, applying a first order Backward Euler method to (2.1) gives

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h\, f(t_{m+1}, \eta_{m+1}^{[0]}). \tag{2.20}$$

2. **Correction step:** use the error function to improve the accuracy of the approximate solution at each iteration.

   For $k = 1$ to K_{loop}, where K_{loop} is the number of correction steps:

   (a) Denote the exact *error function* from the previous step as

$$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{2.21}$$

   where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Notice that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

   (b) Compute the *numerical error vector*, $\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$, using an $(r_k)^{th}$ order numerical method to compute the solution of the error equation,

$$(e^{(k-1)})'(t) = f(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - (\eta^{(k-1)})'(t) e^{(k-1)}(0) = 0, \tag{2.22}$$

where $\eta^{(k-1)}(t)$ and $(\eta^{(k-1)})'(t)$ are approximated by the Mth degree polynomial interpolant and its derivative, respectively. $\delta^{[k]}$ is an $(r_k)^{th}$ order approximation to

$$e^{[k-1]} = (e_0^{[k-1]}, \ldots, e_m^{[k-1]}, \ldots, e_M^{[k-1]}), \tag{2.23}$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function at t_m.

(c) Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

2.1.4 Order Theorems for DC Methods

Theorem 2.1.1. *Consider a DC method as in Section 2.1.1 with a global [28] or local [5] connection strategy, equidistant grids $h_m = h = \frac{H}{M}$ for all m, fixed degree M of the interpolating polynomials, and $H \to 0$. If an arbitrary (explicit or implicit) RK scheme of order p ($p \leq M$) is used, and if f satisfies suitable differentiability conditions, then the approximate solution after the kth correction satisfies*

$$\eta_m^{[k]} - y(t_m) = \mathcal{O}(h^{min(p(k+1),M)}) \text{ for } h \to 0, \tag{2.24}$$

where y is the exact solution.

[27] has a similar version of Theorem 2.1.1, with variant assumptions:

Theorem 2.1.2. *If the discretization method is of order q, if the approximation $\eta^{[k-1]}$ from the previous DC step(s) gives an $\mathcal{O}(h^r)$ error, and if certain smoothness requirements are met, then the result $\eta^{[k]}$ of a DC step satisfies*

$$\eta^{[k]} - \Pi_h y = \mathcal{O}(h^{min(r+q),J)}) \text{ for } h \to 0, \tag{2.25}$$

where $\Pi_h y$ is the projection of the exact solution y onto the grid, provided that the maximum attainable order J is larger than p and q. Moreover, further DC steps are possible if $J > r + q$.

In essence, Theorem 2.1.2 tells us that Theorem 2.1.1 holds when p is different at each DC iteration, i.e. when the discretization method has a different order at each iteration. Note the importance of equidistant gridspacing. The same results do not generally hold for nonequidistant grids.

2.1.5 DC Methods and Collocation

As aforementioned, the result of the DC solution approaches a collocation solution, or rather the fixed point of the DC method is a collocation solution to the problem (2.13). This collocation solution is determined by the choice of interpolating polynomial, for example, and also determines the maximum attainable order of accuracy of the DC method. The following theorem from [28], where the discretization method used at each DC step is backward Euler, illustrates one such claim.

Theorem 2.1.3. *On $[0, H]$, $\eta^* = (y_0, \eta_1^*, \ldots, \eta_M^*)$ is a fixed point of the DC method with base scheme of backward Euler if and only if*

$$d^*(t_m) = 0, \quad m = 0, 1, \ldots, M \tag{2.26}$$

where

$$d^*(t) \doteq (\eta^{(*)})'(t) - f(t, \eta^{(*)}(t)), \tag{2.27}$$

and $\eta^{(*)}(t)$ interpolates η^* (i.e. $\eta^{(*)}(t)$ is the solution of the corresponding collocation scheme). This theorem holds for all nodes (2.3), not necessarily equispaced; however, equidistant nodes seem to converge to η^* more readily.

2.1.6 Continuing Developments in DC Methods

In [6], Stetter is quoted: "The scheme [DC] is simple; the essential point is the definition of the defect [residual]." The treatment of the residual does in fact appear to be the major factor in successful adaptations of DC methods. In the years immediately following DC's conception, some difficulties became apparent. One of these troubles was a lower than expected order of accuracy with nonequidistant nodes (recall that Theorems 2.1.1 and 2.1.2 required equispacing). However, a benefit of some nonequidistant nodes is a higher order collocation solution with fewer gridpoints (e.g. Gaussian quadrature). In 2004, Auzinger et al [5] rewrote the residual in several different ways to obtain similar order of accuracy results for nonequispaced nodes. For one modified DC method, the residual and the embedded discretization scheme are both approximated at the unequally spaced gridpoints. For another method, the residual was interpolated at the unequally spaced nodes while the discretization method at each iteration was carried out at equidistant gridpoints. Auzinger et al also provided some numerical examples where their modified DC methods were successfully (and unsuccessfully) applied to stiff IVPs [6].

Another difficulty of early DC methods is the possible instability of the method even when the lower order base scheme is stable, which may be due to the discretization used for forming the residual [40, 35]. The next section suggests how the discretization of the residual could affect the stability and discusses a more stable formulation introduced in 2000.

2.2 Spectral Deferred Correction Methods

Recall in Section 2.1.3, the formation of the residual involved the derivative of a polynomial interpolant. This derivative is typically approximated by what is known

as spectral differentiation.

A ***spectral differentiation matrix*** is a linear map

$$\mathcal{D}_M : \begin{bmatrix} \vdots \\ f(x_m) \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} \vdots \\ f'(x_m) \\ \vdots \end{bmatrix}, \tag{2.28}$$

where $\mathcal{D}_M$ can be found by representing f by a polynomial interpolant of degree M (or any truncated series expansion), and differentiating the interpolant, so that the ijth entry of $\mathcal{D}_M$ is:

$$\mathcal{D}_{ij} = \frac{d}{dt} c_j(t)\Big|_{t=t_i}, \tag{2.29}$$

where

$$c_j(t) = \prod_{k=0,\, k\neq j}^{M} \frac{t - t_k}{t_j - t_k}. \tag{2.30}$$

The problem is that $\mathcal{D}_M$ is increasingly ill-conditioned as M increases [32]. Spectral integration, on the other hand, appears to be inherently more stable.

A ***spectral integration matrix*** is a linear map

$$\mathcal{I}_M : \begin{bmatrix} \vdots \\ f(x_m) \\ \vdots \end{bmatrix} \rightarrow \begin{bmatrix} \vdots \\ \int^{x_m} f(x)dx \\ \vdots \end{bmatrix}, \tag{2.31}$$

where $\mathcal{I}_M$ also can be found by representing f by a truncated series expansion. For example, representing f by its Lagrange polynomial interpolant of degree M, the

ijth entry of $\mathcal{I}_M$ is:

$$\mathcal{I}_{ij} = \int_{t_0}^{t_i} c_j(t)dt, \tag{2.32}$$

where $c_j(t)$ is as defined in (2.30) above.

Perceiving spectral integration's benefit, Greengard introduced spectral integration to replace spectral differentiation within spectral methods [32]. He proposed recasting the differential equation as an integral equation, where the solution can be stably recovered by integration. For example, he found that for the differentiation matrix $\mathcal{D}_M$, the process of differentiation via Chebyshev series can amplify errors by a factor proportional to M^2, whereas for the integration matrix $\mathcal{I}_M$, the process of integration amplifies errors by a factor of less than 2.4.

In 2000 [25], spectral integration rather than differentiation was applied to the residual in DC, and the new method was designated as Spectral Deferred Correction (SDC). Apparently SDC does not have the instability issues which plague classical DC [60], [40], [35], [32], and it may even cause improved stability in some cases where explicit RK methods are used as the base discretization schemes [18], [17].

2.2.1 Formulation of SDC Methods

SDC methods are deferred correction methods which use spectral integration to approximate the residual and (in the original formulation) Euler timestepping to update the prediction and correction steps [25]. The details of the algorithm may be seen by applying an SDC method to the scalar IVP (2.1). The time interval, $[0, T]$, is discretized into subintervals

$$t_n = t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{2.33}$$

where $H = t_{n+1} - t_n$. Each subinterval $[t_n, t_{n+1}]$ is discretized again into M subintervals

$$t_n = t_{n,0} < t_{n,0} < \cdots < t_{n,m} < \cdots < t_{n,M} = t_{n+1}, \tag{2.34}$$

M is fixed. We consider the SDC method on one subinterval $[t_n, t_{n+1}]$ and drop the subscript n.

1. **Prediction step:** compute an approximate solution to (2.1) over the grid points (2.34),

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}), \tag{2.35}$$

which is an r_0^{th} order approximation to the exact solution

$$\mathbf{y} = (y_0, y_1, \ldots, ym, \ldots, y_M). \tag{2.36}$$

For example, applying a first order backward Euler method to (2.1) gives

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h\, f(t_{m+1}, \eta_{m+1}^{[0]}). \tag{2.37}$$

2. **Correction step:** use the error function to improve the accuracy of the approximate solution at each iteration.

For $k = 1$ to K_{loop}, where K_{loop} is the number of correction steps:

(a) Denote the exact *error function* from the previous step as

$$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{2.38}$$

where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Notice that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

(b) Compute the *residual function*,

$$\epsilon^{(k-1)}(t) = (\eta^{(k-1)})'(t) - f(t, \eta^{(k-1)}(t)). \tag{2.39}$$

(c) Compute the *numerical error vector*, $\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$, using an $(r_k)^{th}$ order numerical method to discretize the integral form of the error equation,

$$(e^{(k-1)})'(t) = f(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f(t, \eta^{(k-1)}(t)) - \epsilon^{(k-1)}(t) \tag{2.40}$$

$$\doteq F(t, e^{(k-1)}(t)) - \epsilon^{(k-1)}(t), \tag{2.41}$$

where $F(t, e(t)) = f(t, \eta(t) + e(t)) - f(t, \eta(t))$. Denote $\delta^{[k]}$ as an $(r_k)^{th}$ order approximation to

$$e^{[k-1]} = (e_0^{[k-1]}, \ldots, e_m^{[k-1]}, \ldots, e_M^{[k-1]}), \tag{2.42}$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function at t_m. For example, applying a first order backward Euler method to (2.41)

gives

$$\delta_{m+1}^{[k]} = \delta_m^{[k]} + h(f(t_{m+1}, \eta_{m+1}^{(k-1)} + \delta_{m+1}^{(k)}(t)) - f(t_{m+1}, \eta_{m+1}^{(k-1)})) \tag{2.43}$$

$$- \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt. \tag{2.44}$$

Note the integral $\int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt$ is evaluated with $(M+1)$th order (or higher if Gaussian quadrature nodes are used) accuracy via spectral integration as in (2.32).

(d) Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

2.2.2 Order Theorems for SDC With Euler Base Schemes

In the original SDC paper [25], the gridpoints (2.34) could be arbitrary and the following theorem is satisfied for SDC with either forward or backward Euler as the base discretization scheme:

Theorem 2.2.1. *[25] For any sufficiently smooth function $f : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ and any natural numbers M, K_{loop}, SDC with either Euler method as the base discretization scheme with $M+1$ submeshpoints (on the interval $[0, H]$) and K_{loop} correction steps converges to the exact solution $\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M)$ with order of accuracy $\mathcal{O}(h^{min(M+1, K_{loop}+1)})$.*

Xia, Xu, and Shu [76] also prove a similar theorem. The use of Gauss-Lobatto quadrature nodes in the submeshpoints results in an improved order of accuracy when SDC is combined with one of the Euler methods. With $M+1$ Gauss-Lobatto nodes, the order is $\mathcal{O}(h^{min(2M, K_{loop}+1)})$ [41].

2.2.3 Stability for SDC Methods With Euler Base Schemes

Stability regions for SDC methods can be calculated in the traditional sense.

Definition 2.2.2. *Region of absolute stability. When solving the Dahlquist problem*

$$y' = \lambda y \text{ for } t \in [0,1] \text{ and } \lambda \in \mathbb{C} \tag{2.45}$$

$$y(0) = 1, \tag{2.46}$$

the amplification factor $Am(\lambda)$ *(or the stability function* $R(z)$ *for* $z = \lambda h \in \mathbb{C}$, h *is one timestep) is defined by*

$$Am(\lambda) = R(z) \doteq \eta_1, \tag{2.47}$$

where η_1 *is the numerical solution at time* $t = 1$ *(or* $t = h$*). The region of absolute stability* S *is*

$$S(\lambda h) = S(z) = \{\lambda \in \mathbb{C} \text{ s.t. } |Am(\lambda)| \leq 1\} = \{z \in \mathbb{C} \text{ s.t. } |R(z)| \leq 1\}. \tag{2.48}$$

SDC with a base scheme of forward Euler has an interesting property that the size of the stability region increases as the order of the method increases [25]. All of the SDC methods with a base scheme of backward Euler are $A(\alpha)$-stable. They are not L-stable, but the limit $lim_{|\lambda|\to\infty} Am(\lambda)$ is less than $\frac{1}{2}$. Unfortunately, some of the imaginary axis appears to be lost as the order increases [25].

2.2.4 Further Developments for SDC Methods

Several variants of SDC methods have been developed recently. Huang, Jia, and Minion developed an accelerated version of SDC methods for ODE initial value problems and for differential algebraic equation systems using Newton-Krylov methods

[41, 42]. In these so-called Krylov Deferred Correction methods, the original SDC method acts as a preconditioner for the system. An iteration of SDC preconditions the equation, a solution for the error is computed by a Newton-Krylov method, and then the process is repeated, thus accelerating SDC's convergence to the collocation solution. Also, Minion developed semi-implicit SDC methods which treat the stiff and nonstiff parts of a stiff ODE separately via an implicit and explicit method, respectively [60]. Christlieb, Qiu, and Ong experimented with higher order explicit RK methods as the base scheme in Integral Deferred Correction methods (IDC), which are motivated by SDC methods [18, 17] (see also Section 2.3.1). Also, higher order splitting methods for PDEs were constructed by using ADI methods within the differential DC framework, including applications to Vlasov-Maxwell systems with periodic boundary conditions [46, 49, 50].

2.2.5 Semi-implicit SDC Methods Constructed with Euler Methods

When a stiff ODE can be split into a nonstiff part and a stiff part, it is advantageous to treat the nonstiff term explicitly and the stiff term implicitly in order to save computational cost but still handle the stiffness effectively. See the introduction to Chapter 3 for more details on semi-implicit methods. Consider the initial value problem

$$\begin{cases} y'(t) &= f(t,y) = f_S(t,y) + f_N(t,y), \quad t \in [0,T], \\ y(0) &= y_0. \end{cases} \tag{2.49}$$

The stiff terms are contained in $f_S(t,y)$, and $f_N(t,y)$ contains only nonstiff terms.

Minion [60] developed semi-implicit SDC (SISDC) methods to handle an IVP of the form (2.49). In each predicition and correction step of an SDC method, backward Euler was applied to the stiff term while forward Euler was applied to the

nonstiff term. I.e.,

1. **Prediction step:** compute an approximate solution to (2.49) over the grid points (2.86),

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}), \tag{2.50}$$

which is a first order approximation to the exact solution

$$\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M). \tag{2.51}$$

when we apply forward and backward Euler as follows:

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h\, f_S(t_{m+1}, \eta_{m+1}^{[0]}) + h\, f_N(t_m, \eta_m^{[0]}). \tag{2.52}$$

2. **Correction step:** use the error function to improve the accuracy of the approximate solution at each iteration.

For $k = 1$ to K_{loop}, where K_{loop} is the number of correction steps:

(a) Denote the exact *error function* from the previous step as

$$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{2.53}$$

where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Notice that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

(b) Compute the *residual function*,

$$\begin{aligned} \epsilon^{(k-1)}(t) &\doteq (\eta^{(k-1)})'(t) - f(t, \eta^{(k-1)}(t)) \\ &= (\eta^{(k-1)})'(t) - f_S(t, \eta^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)). \end{aligned}$$

(c) Compute the *numerical error vector*, $\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$, discretizing the integral form of the error equation,

$$(e^{(k-1)})'(t) = f_S(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \quad (2.54)$$

$$+ f_N(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)) \quad (2.55)$$

$$- \epsilon^{(k-1)}(t) \quad (2.56)$$

$$\doteq F_S(t, e^{(k-1)}(t)) + F_N(t, e^{(k-1)}(t)) - \epsilon^{(k-1)}(t), \quad (2.57)$$

where $F_S(t, e(t)) = f_S(t, \eta(t) + e(t)) - f_S(t, \eta(t))$ and $F_N(t, e(t)) = f_S(t, \eta(t) + e(t)) - f_N(t, \eta(t))$. Then $\delta^{[k]}$ is a first order approximation to

$$e^{[k-1]} = (e_0^{[k-1]}, \ldots, e_m^{[k-1]}, \ldots, e_M^{[k-1]}), \quad (2.58)$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function at t_m, when we apply a first order backward Euler method to (2.57) gives:

$$\delta_{m+1}^{[k]} = \delta_m^{[k]} + h(f_S(t_{m+1}, \eta_{m+1}^{(k-1)} + \delta_{m+1}^{(k)}(t)) - f_S(t_{m+1}, \eta_{m+1}^{(k-1)}) \quad (2.59)$$

$$+ f_N(t_m, \eta_m^{(k-1)} + \delta_m^{(k)}(t)) - f_N(t_m, \eta_m^{(k-1)})) \quad (2.60)$$

$$- \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt, \quad (2.61)$$

where the integral of the residual is approximated via some quadrature method, e.g., using (2.31).

(d) Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

Order of Accuracy of SISDC With Euler Base Schemes

SISDC with forward and backward Euler has the same order of accuracy rules as for SDC with forward Euler alone or with backward Euler alone.

Theorem 2.2.3. *[25] For any sufficiently smooth function* $f = f_S + f_N : \mathbb{R} \times \mathbb{C} \to \mathbb{C}$ *and any natural numbers* M, K_{loop}, *SISDC with backward and forward Euler methods applied to* f_S *and* f_N *respectively with* $M+1$ *submeshpoints (on the interval* $[0, H]$*) and* K_{loop} *correction steps converges to the exact solution* $\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M)$ *with order of accuracy* $\mathcal{O}(h^{min(M+1, K_{loop}+1)})$.

Uniform stepsizes may not be necessary, for example, Gauss-Lobatto nodes appear to work as well [60]. Numerical work suggests that when neither f_S nor f_N are stiff, then SISDC with forward and backward Euler over Gauss-Lobatto nodes is $min(2M, K_{loop}+1)$ order accurate. However, when applied to Van der Pol's equation as given in Section 5 of [60], SISDC shows order reduction for both uniform and Gauss-Lobatto nodes when f_S is stiff.

Stability of SISDC With Euler Base Schemes

Semi-implicit methods designed to solve equations of the form (2.49) require a non-standard definition of stability. One definition of stability used in [60] considers, rather than the Dahlquist equation (2.45), a test problem of the form

$$\begin{cases} y'(t) &= \beta i y + \alpha y \\ y(0) &= 1 \end{cases} \tag{2.62}$$

where α and β are real and correspond to the stiff and nonstiff situations, respectively. I.e., $\beta i y$ is treated explicitly and αy is treated implicitly when calculating the stability region for a semi-implicit method. This definition seems lacking, e.g.,

when (2.62) is employed as a test equation, the case of stiff imaginary values is excluded, but is reasonable for considering stability of the method when applied to advection-diffusion equations. The results of using (2.62) indicate that SISDC with forward and backward Euler have increasing stability as the order increases. Although stability regions scaled by the number of function evaluations suggest a different story, accuracy may in fact confirm the indication of the unscaled regions. See Section 4 of [60] for further details.

2.3 Integral Deferred Correction Methods

2.3.1 Explicit IDC-RK Methods

In [18], higher order explicit Runge-Kutta (RK) integrators as the base discretization schemes in SDC methods was investigated. It was shown that the order of the SDC method improves by the order of the RK method for each correction step of SDC whenever the gridpoints (2.3) are equally spaced, but numerical results show that the same order increase does not occur with a nonequidistant grid, e.g. Gauss-Lobatto nodes. Since equidistant gridpoints are used for these methods constructed with RK integrators, and since incorporating RK methods requires a rewriting of the integral formulation given in the construction of SDC methods (see Section 2.3.2 for details on the new formulation), these new constructions are desinated Integral Deferred Correction (IDC) methods. In particular, IDC methods constructed with RK integrators are designated IDC-RK methods.

The following order result was rigorously derived for explicit IDC-RK methods.

Theorem 2.3.1. *Let $y(t)$ be the solution to the IVP (2.1) and $y(t)$ has at least S ($S \geq M + 2$) degrees of smoothness in the continuous sense. Consider one time interval of an SDC method with $t \in [0, H]$ and uniformly distributed quadrature*

points (2.3). *Suppose an* r_0*th order explicit RK method is used in the prediction step and* $(r_1, r_2, \ldots, r_{K_{loop}})$*th order explicit RK methods are used in* K_{loop} *correction steps. Let* $s_k = \sum_{j=0}^{k} r_j$. *If* $s_{K_{loop}} \leq M + 1$, *then the local truncation error is of order* $\mathcal{O}(h^{s_{K_{loop}}+1})$.

Stability regions for fourth, 6th, 8th, and 12th order explicit IDC-RK methods are plotted in [18]. For 4th order methods, stability regions of SDC with four steps of forward Euler, of IDC-RK with two steps of 2nd order RK, and of 4th order RK alone are plotted. Similar combinations are plotted for the 6th, 8th, and 12th order IDC-RK methods. Rth order IDC-RK stability regions are all larger than the region for the Rth order explicit RK method. Also, as the order of the base RK scheme increases, the stability region increases [18, 17]. SDC and IDC methods appear to have stability-preserving or -enhancing properties, for certain orders, which may be investigated further in [30, 31, 57].

Accuracy regions were also calculated for IDC-RK methods. A comparison between IDC-RK and RK methods to show the attainable accuracy for a fixed number of function evaluations is also given [18, 17].

Definition 2.3.2. *An accuracy region for a numerical method is found first by solving (2.45) with a fixed number of function evaluations. Then the accuracy region is given by a contour plot of the error at* $T = 1$ *for that fixed number of function evaluations over* $\lambda \in \mathbb{C}$.

As with the stability regions, the size of the accuracy region increases as the order of the base scheme increases. However, the accuracy region of an Rth order IDC-RK method is not always larger than the accuracy region of an Rth order RK method. Fourth and sixth order IDC-RK methods have smaller accuracy regions than the same order RK methods, but an 8th order IDC-RK method (with base

scheme of RK4) and an 8th order RK method have similar accuracy regions. Also, 12th order IDC-RK methods appear to have accuracy regions comparable to that of RK12 (and possibly larger than, in some cases).

Efficiency, which quantifies how much computational work is necessary to obtain a desired error tolerance, is another way to compare various numerical methods. Since efficiency (as well as stability and accuracy plots) is easier to specify for an RK method, first consider that IDC methods with Euler or explicit RK methods as base schemes can be reformulated as a single higher stage RK method [17].

Proposition 2.3.3. *[17] An IDC method, constructed using $(M+1)$ quadrature nodes and $(K_{loop}+1)$ prediction plus correction iterations of an explicit s-stage RK method, can be reformulated as a $((K_{loop}+1)\cdot s\cdot M)$-stage RK method.*

Using the RK reformulation of such IDC methods, we can consider their efficiency. The local truncation error (LTE) of a pth order RK method used to solve (2.1) can be written as

$$LTE = \sum_{i=p+1}^{\infty} h^i \left(\sum_{j=1}^{\lambda_i} \alpha_{ij} D_{ij}\right), \tag{2.63}$$

where $h = \frac{H}{M}$, α_{ij} are the truncation error coefficients, D_{ij} are the elementary differentials, and λ_i is the number of elementary differentials of order $\mathcal{O}(h^i)$. Then, if h is sufficiently small, we can bound the LTE by:

$$LTE \leq h^{p+1}\cdot\lambda_{p+1}\cdot\|\alpha_{p+1,j}\|\infty\cdot\|D_{p+1,j}\|\infty = ch^{p+1}\cdot(\lambda_{p+1}\|D_{p+1,j}\|\infty). \tag{2.64}$$

For two pth order RK methods, the LTE are

$$LTE_1 = c_1 h^{p+1} \cdot (\lambda_{p+1} \| D_{p+1,j} \| \infty) \text{ and } LTE_2 = c_2 h^{p+1} \cdot (\lambda_{p+1} \| D_{p+1,j} \| \infty). \tag{2.65}$$

If the same tolerance ϵ bounds both LTE_1 and LTE_2, then the largest stepsizes that will allow this tolerance for both methods are

$$h_1 = (\frac{\epsilon}{\beta c_1}) \text{ and } h_2 = (\frac{\epsilon}{\beta c_2}), \tag{2.66}$$

respectively, where $\beta = \lambda_{p+1} \| D_{p+1,j} \| \infty$. Since the cost per iteration is s_i, where the first RK method is computed in s_1 stages and the other in s_2 stages, and $\frac{1}{h_i}$ is the number of iterations required, then the total amount of work done by each method is $\frac{s_i}{h_i}$. Taking a ratio of the total amount of work done gives the (relative) efficiency of method 2 as compared with the first method.

$$\text{efficiency} = \frac{s_2/h_2}{s_1/h_1}. \tag{2.67}$$

An efficiency close to 1 means that methods 1 and 2 require similar work to achieve the same error tolerance. An efficiency of 1.5, e.g., means that method 2 requires 50% more work than method 1 to reach the same error tolerance. The results of IDC-RK efficiency calculations corroborate the results of the accuracy plots for the 4th, 6th, and 8th order IDC-RK compared with the same order RK methods (machine precision restrictions prevented the calculation of the efficiency of the 12th order methods) [17].

2.3.2 Implicit IDC-RK Methods

In this section, we present a description of implicit RK methods and their formulation within the framework of IDC methods. We provide a general formulation that allows the construction of an arbitrary order implicit IDC method based upon the number of correction loops or the order of the (arbitrary) implicit RK subscheme. The order of accuracy results for implicit IDC-RK methods are analogous to the results for explicit IDC-RK methods, with some changes in the proof; therefore we provide a rigorous analysis of the local truncation error of such implicit IDC-RK methods.

Implicit Runge-Kutta Methods

An implicit Runge-Kutta numerical integrator can be defined as follows:

Definition 2.3.4. *[37] Let p denote the number of stages and $a_{11}, a_{12}, \ldots, a_{1p}$; $a_{21}, a_{22}, \ldots, a_{2p}; \ldots;$ $a_{p1}, a_{p2}, \ldots, a_{pp}$; $b_1, b_2, \ldots, b_p$; $c_1, c_2, \ldots, c_p$ be real coefficients. Then the method*

$$k_i = f(t_0 + c_i h, y_0 + h \sum_{j=1}^{p} a_{ij} k_j) \text{ for } i = 1, 2, \ldots, p \tag{2.68}$$

$$\eta_1 = y_0 + h \sum_{j=1}^{p} b_j k_j \tag{2.69}$$

is called a p-stage implicit RK method for solving the IVP (2.1). A RK method has order r if for a sufficiently smooth IVP (2.1),

$$\|y(t_0 + h) - \eta_1\| \leq K h^{r+1}, \tag{2.70}$$

i.e., the Taylor series for the exact solution $y(t_0 + h)$ and η_1 coincide up to and

including the term h^r.

General Formulation of IDC with implicit RK

Applying the formulation from [18] for IDC with higher order explicit Runge-Kutta, we obtain a general formulation for IDC with implicit RK methods. Consider the initial value problem (2.1). The time interval, $[0, T]$, is discretized into intervals

$$t_n = t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{2.71}$$

where $H = t_{n+1} - t_n$. Each interval $[t_n, t_{n+1}]$ is discretized again into subintervals

$$t_n = t_{n,0} < t_{n,1} < \cdots < t_{n,m} < \cdots < t_{n,M} = t_{n+1}, \tag{2.72}$$

M is fixed. Uniform subintervals $h = t_{n,m+1} - t_{n,m}$ are assumed, based on the results of [18]. The IDC method is applied on each time interval $[t_n, t_{n+1}]$. We drop the subscript n since the method is the same for each time interval. In each prediction and correction loop of IDC, we apply an implicit RK method as in Definition 2.3.4.

- **Prediction loop:** Use an r_0th order numerical method to obtain a numerical solution

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}), \tag{2.73}$$

which is an r_0th order approximation to

$$\mathbf{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M), \tag{2.74}$$

where $y_m = y(t_m)$ is the exact solution at t_m. We apply a an implicit p_0-stage r_0th order RK scheme (2.68) as follows:

$$k_i = f(t_m + c_i h, \eta_m^{[0]} + h \sum_{j=1}^{p_0} a_{ij} k_j) \text{ for } i = 1, 2, \ldots, p_0, \quad (2.75)$$

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h \sum_{j=1}^{p_0} b_j k_j. \quad (2.76)$$

- **Correction loop:** Use the error function to improve the accuracy of the numerical solution at each iteration.

  For $k = 1$ to K_{loop} (K_{loop} is the number of correction loops):

  1. Denote the *error function* from the previous step as

  $$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t),$$

  where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Note that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

  2. Compute the *residual function*,

  $$\epsilon^{(k-1)}(t) \doteq (\eta^{(k-1)})'(t) - f(t, \eta^{(k-1)}(t)).$$

  In fact, the residual is not computed directly, but an integral of the residual is computed, as described below.

  3. Compute the *numerical error vector*, $\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$, which is an r_kth order approximation to

  $$e^{[k-1]} = (e_0^{[k-1]}, \ldots, e_m^{[k-1]}, \ldots, e_M^{[k-1]}), \quad (2.77)$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function at t_m.

To do this, first use an r_kth order numerical method to discretize the integral form of the *error equation*,

$$\begin{cases} (Q^{(k-1)})'(t) = F(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) \doteq G(t, Q^{(k-1)}(t)), \\ Q^{(k-1)}(0) = 0, \end{cases} \tag{2.78}$$

where

$$Q^{(k-1)}(t) = e^{(k-1)}(t) + E^{(k-1)}(t), \tag{2.79}$$

$$F(t, e(t)) = f(t, \eta(t) + e(t)) - f(t, \eta(t)), \tag{2.80}$$

$$E^{(k-1)}(t) = \int_{t_0}^{t} \epsilon^{(k-1)}(\tau) d\tau. \tag{2.81}$$

The r_kth order numerical approximation to $Q_m^{[k-1]} = Q^{(k-1)}(t_m)$ is $\Omega_m^{[k]}$.

Then compute the *numerical error vector* as

$$\delta^{[k]} = \Omega^{[k]} - E^{[k-1]}, \tag{2.82}$$

where $E_m^{[k-1]} = E^{(k-1)}(t_m)$.

We apply a p_k-stage r_kth order implicit RK method to (2.78), giving,

for $i = 1, 2, \ldots, p_k$ and $m = 0, 1, \ldots, M-1$:

$$k_i = F(t_m + c_i h,\, e_m^{[k-1]} + h \sum_{j=1}^{p_k} a_{ij} k_j - \int_{t_0}^{t_m + c_i h} \epsilon^{(k-1)}(\tau) d\tau),$$

$$\Omega_{m+1}^{[k]} = \Omega_m^{[k]} + h \sum_{j=1}^{p_k} b_j k_j,$$

$$\delta_{m+1}^{[k]} = \Omega_{m+1}^{[k]} - E^{[k-1]}(t_{m+1}). \tag{2.83}$$

In fact, we use

$$\begin{aligned} \delta_{m+1}^{[k]} &= \Omega_{m+1}^{[k]} - \int_{t_0}^{t_{m+1}} \epsilon^{(k-1)}(\tau) d\tau \\ &\approx \Omega_{m+1}^{[k]} - \left(\eta_{m+1}^{[k-1]} - y_0 - \sum_{j=0}^{M} \alpha_j f(t_j, \eta_j^{[k-1]}) \right), \end{aligned}$$

where we have approximated the integral by interpolatory quadrature formulas, as in [25], and multiplied $\eta^{[k-1]}$ by an interpolation matrix to obtain its value at intermediate stage times $t_m + c_j h$, $c_j \neq 0, 1$ [18].

4. Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

Truncation Error for IDC-BE

An understanding of the proof of the order of accuracy of implicit IDC-RK methods is aided by walking through the case of backward Euler (BE) before the case of higher order implicit RK as the IDC base scheme. The goal is to show that, under certain conditions, IDC with a first order BE method used in the prediction and correction steps (IDC-BE) has a local truncation error of $\mathcal{O}(h^{K_{loop}+2})$. To show this, we need some definitions and propositions about the degree of smoothness and differentiation of functions and of discrete data. Some of these definitions and

properties are given below in Section 2.3.2, but also see [18] for further details. The following notation will be used: $\frac{\partial}{\partial t}$ = partial derivative and $\frac{d}{dt}$ = total derivative, e.g.

$$\frac{d}{dt}f(t,y(t)) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t} \tag{2.84}$$

Several analytical and numerical preliminaries are needed to analyze IDC methods. The smoothness of discrete data sets will be established, analog to the smoothness of functions; this idea of smoothness is used to analyze the error vectors. Let $\psi(t)$ be a function for $t \in [0,T]$, and without loss of generality, consider the first interval, $[0,H]$, of the grid (2.71). Denote the corresponding discrete data set,

$$(\vec{t},\vec{\psi}) = \{(t_0,\psi_0),\ldots,(t_M,\psi_M)\}, \tag{2.85}$$

where t_m are the uniform quadrature nodes given by (2.72).

Definition 2.3.5. *(smoothness of a function): A function $\psi(t)$, $t \in [0,T]$, possesses σ degrees of smoothness if $\|d^s\psi\|_\infty := \left\|\frac{\partial^s}{\partial t^s}\psi\right\|_\infty$ is bounded for $s = 0,1,2,\ldots,\sigma$, where $\|\psi\|_\infty := \max_{t\in[0,T]}|\psi(t)|$.*

Definition 2.3.6. *(discrete differentiation): Consider the discrete data set, $(\vec{t},\vec{\psi})$, defined in (2.85), and denote L^M as the usual the Lagrange interpolant, an Mth degree polynomial that interpolates (t,ψ). An sth degree discrete differentiation is a linear mapping that maps $\vec{\psi} = (\psi_0,\psi_1,\ldots,\psi_M)$ into $\overrightarrow{\hat{d}_s\psi}$, where $(\hat{d}_s\psi)_m = \frac{\partial^s}{\partial t^s}L^M(t,\psi)|_{t=t_m}$. This linear mapping can be represented by a matrix multiplication $\overrightarrow{\hat{d}_s\psi} = \hat{D}_s\cdot\vec{\psi}$, where $\hat{D}_s \in \mathcal{R}^{(M+1)\times(M+1)}$ and $(\hat{D}_s)_{mn} = \frac{\partial^s}{\partial t^s}c_n(t)|_{t=t_m}$, $m,n = 0,\ldots,M$.*

Given a distribution of quadrature nodes on $[0,1]$, the differentiation matrices, $\hat{D}_s^{[0,1]}$, $s = 1,\ldots,M$ have constant entries. If this distribution of quadrature nodes

is rescaled from $[0,1]$ to $[0,H]$, then the corresponding differentiation matrices are $\hat{D}_1 = \frac{1}{H}\hat{D}_1^{[0,1]}$ and $\hat{D}_s = \left(\frac{1}{H}\right)^s \hat{D}_s^{[0,1]}$.

Definition 2.3.7. *The $(\hat{S},\infty)$ Sobolev norm of the discrete data set $(\vec{t},\vec{\psi})$ is defined as*

$$\left\|\vec{\psi}\right\|_{\hat{S},\infty} := \sum_{s=0}^{\sigma} \left\|\overrightarrow{\hat{d}_s\psi}\right\|_\infty = \sum_{s=0}^{\sigma} \left\|\hat{D}_s \cdot \vec{\psi}\right\|_\infty,$$

where $\overrightarrow{\hat{d}_s\psi} = Id \cdot \vec{\psi}$ is the identity matrix operating on $\vec{\psi}$.

Definition 2.3.8. *(smoothness of a discrete date set): A discrete data set, (2.85), possesses σ ($\sigma \leq M$) degrees of smoothness if $\|\vec{\psi}\|_{\hat{S},\infty}$ is bounded as $h \to 0$.*

We emphasize that smoothness is a property of discrete data sets in the limit as $h \to 0$. We also impose $\sigma \leq M$, because $\overrightarrow{\hat{d}_\sigma\psi} \equiv \vec{0}$, for $\sigma > M$. See [18] for a detailed discussion.

Example 2.3.9. *(A discrete data set with only one degree of smoothness): Consider the discrete data set*

$$(\vec{t},\vec{\psi}) = \left\{(0,0), \left(\frac{H}{4},\frac{H}{4}\right), \left(\frac{H}{2},\frac{H}{2}\right), \left(\frac{3H}{4},\frac{H}{4}\right), (H,0)\right\}.$$

The first derivative

$$\overrightarrow{\hat{d}_1\psi} = \left(-\frac{4}{3},\frac{10}{3},0,-\frac{10}{3},\frac{4}{3}\right),$$

is bounded independent of H, while the second derivative

$$\overrightarrow{\hat{d}_2\psi} = \left(\frac{272}{3H},-\frac{16}{3H},-\frac{112}{3H},-\frac{16}{3H},\frac{272}{3H}\right),$$

is unbounded as $H \to 0$. *Therefore,* $(\vec{t}, \vec{\psi})$ *has one, and only one degree of smoothness in the discrete sense.*

For further details, definitions, and properties of smoothness in the discrete and continuous sense, we refer the reader to [18].

Now consider the local truncation error of SDC methods with a first order backward Euler method in the prediction and correction steps. Analagous to Theorem 4.1 in [18], we have the following theorem:

Theorem 2.3.10. *Let* $y(t)$ *be the solution to the IVP (2.1) and* $y(t)$ *has at least* S *(*$S \geq M + 2$*) degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with* $t \in [0, H]$ *and uniformly distributed quadrature points*

$$0 = t_0 < t_1 < \cdots < t_m < \cdots < t_M = H, \tag{2.86}$$

equispaced with $h = \frac{H}{M}$.

If a first order BE method is used in the prediction and K_{loop} *correction steps (*$K_{loop} < M+1$*), then the local truncation error of the IDC method is* $\mathcal{O}(h^{K_{loop}+2})$

To prove the theorem, two lemmas (below) are required: one for the prediction step and one for the correction step. Together the lemmas prove Theorem 2.3.10 by mathematical induction on k, the number of correction steps. Lemma 2.3.11, analogous to Lemma 4.2 in [18], is the initial case. Lemma 2.3.12, analogous to Lemma 4.3 in [18], is the induction step.

Lemma 2.3.11. *(Prediction step). Let* $y(t)$ *be the solution to the IVP (2.1) and* $y(t)$ *has at least* S *(*$S \geq M + 2$*) degrees of smoothness in the continuous sense. Consider one time interval of an IDC-BE method with* $t \in [0, H]$ *and uniformly distributed quadrature points (2.86). Let* $\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ *be the numerical solution on the quadrature points (2.86) at the prediction step. Then:*

1. *The error vector* $e^{[0]} = y - \eta^{[0]}$ *satisfies*

$$\|e^{[0]}\|_\infty \sim \mathcal{O}(h^2) \text{ as } t \to 0. \tag{2.87}$$

2. *The rescaled error vector* $\widetilde{e}^{[0]} = \frac{1}{h} e^{[0]}$ *has* M *degrees of smoothness in the discrete sense.*

Proof. For convenience, drop the superscript $^{[0]}$. Apply a backward Euler update to obtain an approximate solution $\eta_{m+1} = \eta_m + h\, f(t_{m+1}, \eta_{m+1})$. For the exact solution $y(t)$, Taylor expand $y(t_m) = y_m$ about $t = t_{m+1}$. Then

$$y_{m+1} = y_m + h\, f(t_{m+1}, y_{m+1}) + R_{m+1} \tag{2.88}$$

$$= y_{m+1} + \frac{h^2}{2!} y^{(2)}_{m+1} - \cdots + \frac{(-1)^{(S-1)} h^{(S-1)}}{(S-1)!} y^{(S-1)}_{m+1} + \mathcal{O}(h^S) + R_{m+1} \tag{2.89}$$

gives

$$R_{m+1} = \sum_{i=2}^{S-1} \frac{(-1)^{(i+1)}}{i!} h^i y^{(i)}_{m+1} + \mathcal{O}(h^S) \doteq r_{m+1} + \mathcal{O}(h^S) \tag{2.90}$$

Clearly $r_{m+1} \sim \mathcal{O}(h^2)$ (since y has S degrees of smoothness).

Now consider the error:

$$e_{m+1} = y_{m+1} - \eta_{m+1} \tag{2.91}$$

$$= y_m - \eta_m + h\,(f(t_{m+1}, y_{m+1}) - f(t_{m+1}, \eta_{m+1})) + r_{m+1} + \mathcal{O}(h^S) \tag{2.92}$$

$$\doteq e_m + h\, u_{m+1} + r_{m+1} + \mathcal{O}(h^S) \tag{2.93}$$

Taylor expanding about y_{m+1} gives

$$u_{m+1} = f(t_{m+1}, y_{m+1}) - f(t_{m+1}, \eta_{m+1}) \tag{2.94}$$

$$= (f(t_{m+1}, y_{m+1}) - f(t_{m+1}, y_{m+1} - e_{m+1})) \tag{2.95}$$

$$= \sum_{i=1}^{S-2} \frac{(-1)^{i+1}}{i!} (e_{m+1})^i \frac{\partial^i}{\partial y^i} f(t_{m+1}, y_{m+1}) + \mathcal{O}((e_{m+1})^{S-1}) \tag{2.96}$$

1. Now prove part 1. of Lemma 2.3.11 by induction on m, the index of gridpoints on $[0, H]$.

   - When $m = 0$, $e_0 = 0 \sim \mathcal{O}(h^2)$.
   - Assume $e_m \sim \mathcal{O}(h^2)$, and show $e_{m+1} \sim \mathcal{O}(h^2)$.

   Then $e_{m+1} = e_m + h\,u_{m+1} + r_{m+1} + \mathcal{O}(h^S)$, or:

$$e_{m+1} - h\,u_{m+1} = e_m + r_{m+1} + \mathcal{O}(h^S) \tag{2.97}$$

$$= \mathcal{O}(h^2) + \mathcal{O}(h^2) + \mathcal{O}(h^S) = \mathcal{O}(h^2). \tag{2.98}$$

   Note that

$$e_{m+1} - h\,u_{m+1} = e_{m+1} - h\,(\sum_{i=1}^{S-2} \frac{(-1)^{i+1}}{i!} (e_{m+1})^i \frac{\partial^i}{\partial y^i} f(t_{m+1}, y_{m+1}) + \mathcal{O}((e_{m+1})^{S-1})).$$

   Letting $e = e_{m+1}$, we have an equation of the form:

$$\mathcal{O}(h^2) = e + h\,\widetilde{a}_1\,e + h\,\widetilde{a}_2\,e^2 + \cdots + h\,\widetilde{a}_{S-2}\,e^{S-2} + \mathcal{O}(h\,e^{S-1}), \tag{2.99}$$

   where the $\widetilde{a}_j$ are bounded and do not depend on h, since y has S degrees of

smoothness. Now we will show by way of contradiction that e must be $\mathcal{O}(h^2)$. Suppose $e \sim \mathcal{O}(h^p)$ for some $p < 2$. Then

$$\mathcal{O}(h^2) = a_0\, h^p + a_1\, h^{p+1} + a_2\, h^{2p+1} + \cdots + a_{S-2}\, h^{(S-2)p+1} + \mathcal{O}(h^{(S-1)p+1}), \tag{2.100}$$

where the a_j are bounded and do not depend on h, which implies that the 0^{th} term must cancel with some later term(s). It is impossible for the 0^{th} term to cancel with more than one term since the coefficients do not depend on h. Thus we must have

$$-\frac{a_l}{a_0} = h^{p-lp-1} \text{ for some integer } l \geq 2. \tag{2.101}$$

Since $-\frac{a_l}{a_0}$ does not depend on h, the only possibility is $p = \frac{-1}{l-1}$. Noting that $v\,p + 1 = 1 - \frac{v}{l-1} < 1$ for all $v \geq 1$, we conclude that the right hand side is larger than $\mathcal{O}(h)$, a contradiction.

Hence $e_{m+1} \sim \mathcal{O}(h^2)$, and therefore (2.87) holds.

2. The proof of part 2 of Lemma 2.3.11 is nearly identical to that in Lemma 4.2 of [18]. For completeness, we include the proof here. We again use an inductive approach, but this time, with respect to s, the degree of smoothness. First, note that a divided difference approximation to the derivative of the rescaled error vector gives

$$(d_1\widetilde{e})m = \frac{\widetilde{e}_{m+1}\widetilde{e}m}{h} = \widetilde{u}_{m+1} + \frac{r_{m+1}}{h^2} + \mathcal{O}(h^{S-2}), \tag{2.102}$$

where

$$\widetilde{u}_{m+1} = \frac{u_{m+1}}{h} = \sum_{i=1}^{S-2} h^{i-1} f_{y^i}(t_{m+1}, y_{m+1})(\widetilde{e}_{m+1})^i + \mathcal{O}(h^{2S-3}),$$

since $u_{m+1} = \sum_{i=1}^{S-2} f_{y^i}(t_{m+1}, y_{m+1})(e_{m+1})^i + \mathcal{O}((e_{m+1})^{S-1})$ and $\|\widetilde{e}\|_\infty \sim \mathcal{O}(h)$. We are now ready to prove that $\widetilde{e}$ has M degrees of smoothness by induction. Again, since $\|\widetilde{e}\|_\infty \sim \mathcal{O}(h)$, $\widetilde{e}$ has at least zero degrees of smoothness in the discrete sense. Assume that $\widetilde{e}$ has $s \leq M-1$ degrees of smoothness. We will show that $(d_1\widetilde{e})$ has s degrees of smoothness, from which we can conclude that $\widetilde{e}$ has $(s+1)$ degrees of smoothness. Since f_{y^i} has $(S-i-1)$ degrees of smoothness in the continuous sense, the vector form $f_{y^i} = [f_{y^i}(t_0, y_0), \ldots, f_{y^i}(t_M, y_M)]$ has $(S-i-1)$ degrees of smoothness in the discrete sense. Consequently, $h^{i-1} f_{y^i}$ (vector) has $(S-2)$ degrees of smoothness, which implies that $\widetilde{u}$ has min $(S-2, s)$ degrees of smoothness. Similarly, $\frac{r}{h^2}$ has $(S-2)$ degrees of smoothness in the discrete sense. Hence $(d_1\widetilde{e})$ has s degrees of smoothness, which implies that $\widetilde{e}$ has $(s+1)$ degrees of smoothness. Since this argument holds for $S \geq M+2$, we can conclude that $\widetilde{e}$ has M degrees of smoothness.

□

Lemma 2.3.12. *(Correction steps). Let $y(t)$ be the solution to the IVP (2.1) and $y(t)$ has at least S ($S \geq M+2$) degrees of smoothness in the continuous sense. Consider one time interval of an IDC-BE method with $t \in [0, H]$ and uniformly distributed quadrature points (2.86). Suppose in the $(k-1)$th ($k \leq M$) correction step, $e^{[k-1]} \sim \mathcal{O}(h^{k+1})$ and the rescaled error vector $\widetilde{e}^{[k-1]} = \frac{1}{h^k} e^{[k-1]}$ (now $\mathcal{O}(h)$) has $M-k+1$ degrees of smoothness in the discrete sense. Then, after another correction step:*

1. *The updated error vector* $e^{[k]}$ *satisfies*

$$\|e^{[k]}\|_\infty \sim \mathcal{O}(h^{k+2}). \tag{2.103}$$

2. *The rescaled error vector* $\widetilde{e}^{[k-1]} = \frac{1}{h^k} e^{[k-1]}$ *has* $(M-k)$ *degrees of smoothness in the discrete sense.*

Proof. We integrate $(e^{(k-1)})'(t)$ and use that result and $\delta^{[k]}$ to obtain an equation for $e^{[k]}$. Then we prove 1. and 2.

Recall the error equation (2.78) at the end of the $(k-1)^{th}$ correction step, and integrate it over $[t_m, t_{m+1}]$.

$$\begin{aligned}
e_{m+1}^{[k-1]} &= e_m^{[k-1]} + \int_{t_m}^{t_{m+1}} F(t, e^{(k-1)}(t))dt - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt \\
&= e_m^{[k-1]} + \int^{t_{m+1}} F(t, e^{(k-1)}(t))dt - \int_{t_m} F(t, e^{(k-1)}(t))dt \\
&\quad - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt \\
&= e_m^{[k-1]} + \int^{t_{m+1}} F(t, e^{(k-1)}(t))dt \\
&\quad - \left(\int^{t_{m+1}} F(t, e^{(k-1)}(t))dt + (-1)h\, F(t_{m+1}, e_{m+1}^{[k-1]}) \right. \\
&\quad + (-1)^2 \frac{h^2}{2!} \frac{d}{dt} F(t_{m+1}, e_{m+1}^{[k-1]}) + \dots \\
&\quad \left. + (-1)^{M+1} \frac{h^{M+1}}{(M+1)!} \frac{d^M}{dt^M} F(t_{m+1}, e_{m+1}^{[k-1]}) + \mathcal{O}(h^{M+2}) \right) \\
&\quad - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt
\end{aligned}$$

$$e_{m+1}^{[k-1]} = e_m^{[k-1]} + \sum_{i=1}^{M+1} (-1)^{i+1} \frac{h^i}{i!} \frac{d^{i-1}}{dt^{i-1}} F(t_{m+1}, e_{m+1}^{[k-1]}) + \mathcal{O}(h^{M+2}) - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt \tag{2.104}$$

Using $\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$ to denote the numerical error vector obtained from the backward Euler scheme at the k^{th} correction step gives:

$$\delta_{m+1}^{[k]} = \delta_m^{[k]} + h\left(f(t_{m+1}, \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k-1]}) - f(t_{m+1}, \eta_{m+1}^{[k-1]})\right) - \left(\int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt + \mathcal{O}(h^{M+2})\right), \tag{2.105}$$

where the last term is due to the spectral integration used to numerically integrate $\epsilon^{(k-1)}(t)$ at $M+1$ gridpoints.

Now subtract equation (2.105) from equation (2.104) in order to obtain $e_{m+1}^{[k]}$.

$$\begin{aligned} e_{m+1}^{[k]} &= e_{m+1}^{[k-1]} - \delta_{m+1}^{[k]} \\ &= e_m^{[k-1]} + \sum_{i=1}^{M+1} (-1)^{i+1} \frac{h^i}{i!} \frac{d^{i-1}}{dt^{i-1}} F(t_{m+1}, e_{m+1}^{[k-1]}) + \mathcal{O}(h^{M+2}) \\ &\quad - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt - \delta_m^{[k]} \\ &\quad - h(f(t_{m+1}, \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k-1]}) - f(t_{m+1}, \eta_{m+1}^{[k-1]})) \\ &\quad + (\int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)dt + \mathcal{O}(h^{M+2})), \end{aligned}$$

$$e_{m+1}^{[k]} = e_m^{[k]} + h(f(t_{m+1}, \eta_{m+1}^{[k-1]} + e_{m+1}^{[k-1]}) - f(t_{m+1}, \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k-1]})) + \sum_{i=1}^{M} (-1)^i \frac{h^{i+1}}{(i+1)!} \frac{d^i}{dt^i} F(t_{m+1}, e_{m+1}^{[k-1]}) + \mathcal{O}(h^{M+2}), \tag{2.106}$$

$$e_{m+1}^{[k]} = e_m^{[k]} + h\,u_{m+1}^{[k]} + r_{m+1}^{[k-1]} + \mathcal{O}(h^{M+2}), \tag{2.107}$$

where

$$\begin{aligned} u_{m+1}^{[k]} &= f(t_{m+1}, y_{m+1}) - f(t_{m+1}, \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k-1]}) \\ &= f(t_{m+1}, y_{m+1}) - f(t_{m+1}, y_{m+1} - e_{m+1}^{[k]}) \\ &= \sum_{i=1}^{S-2} \frac{(-1)^{i+1}}{i!} \frac{\partial^i}{\partial y^i} f(t_{m+1}, y_{m+1})(e_{m+1}^{[k]})^i + \mathcal{O}((e_{m+1}^{[k]})^{S-1}), \end{aligned}$$

and where

$$r_{m+1}^{[k-1]} = \sum_{i=1}^{M} (-1)^i \frac{h^{i+1}}{(i+1)!} \frac{d^i}{dt^i} F(t_{m+1}, e_{m+1}^{[k-1]}).$$

Note

$$\begin{aligned} F(t, e^{(k-1)}(t)) &= f(t, y(t)) - f(t, y(t) - e^{(k-1)}(t)) \\ &= \sum_{j=1}^{S-2} \frac{(-1)^{j+1}}{j!} \frac{\partial^j}{\partial y^j} f(t, y(t))(e^{(k-1)}(t))^j + \mathcal{O}((e^{(k-1)}(t))^{S-1}). \end{aligned}$$

Since $\widetilde{e}^{[k-1]}$ has $(M-k+1)$ degrees of smoothness in the discrete sense, then $\frac{d}{dt}\widetilde{e}^{(k-1)}(t) \sim \mathcal{O}_h(1)$ by Proposition 3.19 from [18]. Then

$$\begin{aligned} \frac{d}{dt} F(t, e^{(k-1)}(t)) = \sum_{j=1}^{S-2} \frac{(-1)^{j+1}}{j!} \Big(&\frac{d}{dt} \frac{\partial^j}{\partial y^j} f(t, y(t))(\widetilde{e}^{(k-1)}(t))^j h^{kj} \\ &+ \frac{\partial^j}{\partial y^j} f(t, y(t)) \frac{d}{dt} (\widetilde{e}^{(k-1)}(t))^j h^{kj} \Big) + \mathcal{O}(\frac{d}{dt}(\widetilde{e}^{(k-1)}(t))^{S-1} h^{k(S-1)}). \end{aligned}$$

1. Now prove part 1. of Lemma 2.3.12 by induction on m, the index of gridpoints on $[0, H]$.

- When $m = 0$, $e_0^{[k]} = 0 \sim \mathcal{O}(h^{k+2})$.
- Assume $e_m^{[k]} \sim \mathcal{O}(h^{k+2})$, and show $e_{m+1}^{[k]} \sim \mathcal{O}(h^{k+2})$.

Then recall

$$u_{m+1}^{[k]} = \sum_{i=1}^{S-2} \frac{(-1)^{i+1}}{i!} \frac{\partial^i}{\partial y^i} f(t_{m+1}, y_{m+1})(e_{m+1}^{[k]})^i + \mathcal{O}((e_{m+1}^{[k]})^{S-1})$$

and

$$e_{m+1}^{[k]} = e_m^{[k]} + h\, u_{m+1}^{[k]} + r_{m+1}^{[k-1]} + \mathcal{O}(h^{M+2}).$$

Note $r_{m+1}^{[k-1]}$ only includes $e_{m+1}^{[k-1]} \sim \mathcal{O}(h^{k+1})$ but never $e_{m+1}^{[k]}$.
Also, $r_{m+1}^{[k-1]} = \mathcal{O}(h\, F) = \mathcal{O}(h \frac{d}{dt} F) = \mathcal{O}(h^{k+2})$.
Thus

$$\mathcal{O}(h^{k+2}) = e_m^{[k]} + r_{m+1}^{[k-1]} = e_{m+1}^{[k]} - h\, u_{m+1}^{[k]},$$

which, just as in the proof of part 1. of Lemma 2.3.11, can be written as

$$\begin{aligned} \mathcal{O}(h^{k+2}) = a_0\, h^p + a_1\, h^{p+1} + a_2\, h^{2p+1} + \cdots + a_{S-2}\, h^{(S-2)p+1} \\ + \mathcal{O}(h^{(S-1)p+1}), \end{aligned}$$

where $e_{m+1}^{[k]} \sim \mathcal{O}(h^p)$.

Then the proof is the same as in Lemma 2.3.11, except we assume by way of contradiction that $p < k + 2$, rather than $p < 2$.

2. The proof of the second part is nearly identical to that of Lemma 4.3 in [18], but we include it here for completeness. Again, we use an inductive argument

based on s, the degree of smoothness of $\widetilde{e}^{[k]}$. First, the rescaled error vector, $\widetilde{e}^{[k]}$, has at least 0 degrees of smoothness since $\|\widetilde{e}^{[k]}\|_\infty \sim \mathcal{O}(h)$ is bounded. Assume that $\widetilde{e}^{[k]}$ has $s < M-k$ degrees of smoothness. We will prove that $(d_1\widetilde{e}^{[k]})$ has s degrees of smoothness, from which we can conclude that $\widetilde{e}^{[k]}$ has $(s+1)$ degrees of smoothness in the discrete sense. The divided difference approximation to the derivative of the rescaled error vector can be expressed as

$$(d_1\widetilde{e}^{[k]})_m = \frac{\widetilde{e}^{[k]}_{m+1} - \widetilde{e}^{[k]}_m}{h} = \widetilde{u}^{[k]}_{m+1} + \widetilde{r}^{[k-1]}_{m+1} + \mathcal{O}(h^{M-k}),$$

where

$$\begin{aligned}\widetilde{u}^{[k]}_{m+1} &= \frac{u^{[k]}_{m+1}}{h^{k+1}} \\ &= \sum_{i=1}^{S-2} h^{(k+1)(i-1)} f_{y^i}(t_{m+1}, y_{m+1})(\widetilde{e}^{[k]}_{m+1})^i + \mathcal{O}\left(h^{(S-2)(k+2)+1}\right)\end{aligned}$$

has s degrees of smoothness, and

$$\begin{aligned}\widetilde{r}^{[k-1]}_{m+1} &= \frac{r^{[k-1]}_{m+1}}{h^{k+2}} = \sum_{i=1}^{M} \frac{h^{i-1}}{(i+1)!}\frac{d^{i-1}}{dt^{i-1}}\left(\frac{1}{h^k}\frac{d}{dt}F(t_{m+1}, e^{(k-1)}(t_{m+1}))\right) \\ &= \sum_{i=1}^{M} \frac{h^{i-1}}{(i+1)!}\frac{d^{i-1}}{dt^{i-1}} \sum_{j=1}^{S-2} \frac{(-1)^{j+1}}{j!} \\ &\quad \left(\left(h^{k(j-1)}\frac{df_{y^j}}{dt}(t_{m+1}, y_{m+1})\right)(\widetilde{e}^{(k-1)}_{m+1})^j + \frac{d(\widetilde{e}^{(k-1)}_{m+1})^j}{dt}(h^{k(j-1)} f_{y^j})\right)\end{aligned}$$

has at least $\min(S-(j+1)+k(j-1)-1, M-k) \geq s$ degrees of smoothness. Since $(d_1\widetilde{e}^{[k]})$ has s degrees of smoothness, we can conclude that $\widetilde{e}^{[k]}$ has $(M+1-k)$ degrees of smoothness.

□

Proof. (of Theorem 2.3.10). Use induction with respect to k, the index of correction steps in the SDC method.

- By Lemma 2.3.11, Theorem 2.3.10 is valid for $k = 0$ and the rescaled error vector $\widetilde{e}^{[0]}$ has M degrees of smoothness in the discrete sense.
- Assume Theorem 2.3.10 is valid for $(k-1)$ correction loops and the rescaled error vector $\widetilde{e}^{[k-1]}$ has $(M-k+1)$ degrees of smoothness. Then by Lemma 2.3.12, Theorem 2.3.10 is also valid for k correction loops and the rescaled error vector $\widetilde{e}^{[k]}$ has $(M-k)$ degrees of smoothness in the discrete sense.

□

Truncation Error for Implicit IDC-RK

Here we provide the local truncation error estimates for IDC methods which use high order implicit RK methods in the prediction and correction steps.

Theorem 2.3.13. *Let $y(t)$ be the solution to the IVP (2.1) and $y(t)$ has at least S ($S \geq M+2$) degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (2.86). Suppose an r_0th order implicit RK method is used in the prediction step and $(r_1, r_2, \ldots, r_{K_{loop}})$th order implicit RK methods are used in K_{loop} correction steps. Let $s_k = \sum_{j=0}^{k} r_j$. If $s_{K_{loop}} \leq M+1$, then the local truncation error is of order $\mathcal{O}(h^{s_{K_{loop}}+1})$.*

The proof of Theorem 2.3.13 follows from Lemmas 2.3.14 and 2.3.17, below, for the prediction and correction steps, respectively.

The following lemma discusses the local truncation error and smoothness of the error for the prediction loop of implicit IDC-RK methods.

Lemma 2.3.14. *(Prediction step): Let $y(t)$ be the solution to the IVP (2.1) and $y(t)$ has at least S ($S \geq M+2$) degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (2.86). Let $\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ be the numerical solution on the quadrature points (2.86), obtained using an r_0th order implicit RK method (see Definition 2.3.4) at the prediction step. Then:*

1. *The error vector $e^{[0]} = y - \eta^{[0]}$ satisfies*

$$\|e^{[0]}\|_\infty \sim \mathcal{O}(h^{r_0+1}) \tag{2.108}$$

and

2. *The rescaled error vector $\widetilde{e}^{[0]} = \frac{1}{h^{r_0}} e^{[0]}$ has $min(S-r_0, M)$ degrees of smoothness in the discrete sense.*

First consider some useful facts, in the exact solution space and the numerical solution space, needed to prove the lemma.

Proposition 2.3.15. *The exact solution of the IVP (2.1), with S degrees of smoothness, satisfies*

$$y_{m+1} = y_m + \sum_{j=1}^{S-1} \frac{h^j}{j!} y^{(j)}(t_m) + \mathcal{O}(h^S) \tag{2.109}$$

The term $y^{(j)}$ in (2.109) can be expressed as

$$y^{(j)}(t) \doteq E_j(t, y(t)) = \sum_{q_t^i + q_y^i \leq j-1} \alpha^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}} \prod_{i=1}^{n_i} (f_{t^{q_t^i} y^{q_y^i}})^{w_i} f^{w_f} \tag{2.110}$$

where $f_{t^{q_t^i} y^{q_y^i}} = \frac{\partial^{(q_t^i + q_y^i)} f}{\partial t^{q_t^i} \partial y^{q_y^i}}$, the $\alpha^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}}$ are constant coefficients, and w_i, w_f, q_t^i, q_t^i, and n_i are nonnegative integers.

Equation (2.110) is called 'elementary differentiation' in any standard numerical ODE book, e.g. [37], and is more easily understood by computing the first few derivatives of y:

$$y^{(1)}(t) = f; \quad y^{(2)}(t) = f_t + f_y f;$$
$$y^{(3)}(t) = f_{tt} + 2\, f_{ty} f + f_{yy} f^2 + f_y f_t f_y^2 f; \quad \ldots \tag{2.111}$$

The equation can be easily proved by induction on j.

Proposition 2.3.16. *Assume that $y(t)$ in IVP (2.1) has $S \geq r$ degrees of smoothness. Suppose that an r^{th} order RK method is used to solve (2.1). If h is the size of each sub-interval $[t_m, t_{m+1}]$, within $[0, H]$, then*

$$\eta_{m+1} = \eta_m + h\, E_1(t_m, \eta_m) + \cdots + \frac{h^r}{r!} E_r(t_m, \eta_m) + R_r(t_m, \eta_m) + \mathcal{O}(h^S), \tag{2.112}$$

where the function $E_r(t, y)$ is defined in (2.110), and the remainder term

$$R_r(t, \eta) = \sum_{j=r+1}^{S-1} h^j \tag{2.113}$$
$$\left(\sum_{q_t^i + q_y^i \leq j-1} \beta^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}} \prod_{i=1}^{n_i} (f_{t^{q_t^i} y^{q_y^i}})^{w_i}(t_m, \eta_m) f^{w_f}(t_m, \eta_m) \right),$$

has constant coefficients $\beta^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}}$ determined by the formulation of the RK method.

Proof. The right hand side of (2.112) comes from Taylor expanding the numerical solution of an r^{th} order RK method. Note that the first $r + 1$ terms coincide with the Taylor expansion of the exact solution (2.109). (2.113) can be proved by induction. □

Proof. (of Lemma 2.3.14). The proof is identical to that given in [18], but we repeat it here for completeness. The superscript $^{[0]}$ is dropped since there is no ambiguity.

1. First, we prove part 1. of the lemma. From equations (2.109) and (2.112), we see that a Taylor expansion of the error, $e_{m+1} = y_{m+1} - \eta_{m+1}$, about $t = t_m$ gives

$$e_{m+1} = e_m + u_m + r_{1,m} - r_{2,m} + \mathcal{O}(h^S),$$

where

$$\begin{aligned}
u_m &= \sum_{i=1}^{r_0} \frac{h^i}{i!}(E_i(t_m, y_m) - E_i(t_m, \eta_m)) \\
&\quad \sum_{i=1}^{r_0} \frac{h^i}{i!} \sum_{j=1}^{S-1-i} \frac{(-1)^{j-1}(e_m)^j}{j!} \frac{\partial^j E_i}{\partial y^j}(t_m, y_m) + \mathcal{O}((e_m)^{S-i}), \\
r_{1,m} &= \sum_{i=r_0+1}^{S-1} \frac{h^i}{i!} y^{(i)}(t_m), \\
r_{2,m} &= \sum_{j=r_0+1}^{S-1} h^j \sum_{q_t^i + q_y^i \leq j-1} \beta^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}} \\
&\quad \prod_{i=1}^{n_i} (f_{t^{q_t^i} y^{q_y^i}})^{w_i}(t_m, \eta_m) f^{w_f}(t_m, \eta_m).
\end{aligned}$$

Now we bound $\|e^{[0]}\|_\infty$ by induction. By definition, $e_0 = 0$, so obviously $e_0 \sim (h^{r_0+1})$. Assume that $e_m \sim (h^{r_0+1})$. Since $u_m \sim (h^{r_0+2})$, $r_{1,m} \sim (h^{r_0+1})$, and $r_{2,m} \sim (h^{r_0+1})$, then we have $e_{m+1} \sim (h^{r_0+1})$, which completes the inductive proof.

2. Now we prove part 2. of the lemma, the smoothness of the rescaled error vector. We again use an inductive approach, but with respect to s, the degree

of smoothness. Note that a divided difference approximation to the derivative of the rescaled error vector gives

$$(d_1\widetilde{e})_m = \frac{\widetilde{e}_{m+1} - \widetilde{e}_m}{h} = \widetilde{u}_m + \widetilde{r}_{1,m} + \widetilde{r}_{2,m} + \mathcal{O}(h^{S-r_0-1}),$$

where

$$\begin{aligned}
\widetilde{u}_m &= \frac{u_m}{h^{r_0+1}} \\
&= \sum_{i=1}^{r_0} \frac{1}{i!} \sum_{j=1}^{S-1-i} \frac{(-1)^{j-1} h^{i+r_0(j-1)-1}}{j!} \frac{\partial^j E_i}{\partial y^j}(t_m, y_m)(\widetilde{e}_m)^j \\
&\quad + \mathcal{O}(h^{S-r_0-1}), \\
\widetilde{r}_{1,m} &= \frac{r_{1,m}}{h^{r_0+1}} = \sum_{i=r_0+1}^{S-1} \frac{h^{i-r_0-1}}{i!} y^{(i)}(t_m), \\
\widetilde{r}_{2,m} &= \frac{r_{2,m}}{h^{r_0+1}} = \sum_{j=0}^{S-r_0-2} h^j \sum_{q_t^i + q_y^i \le j-1} \beta^{w_1 \cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}} \\
&\quad \prod_{i=1}^{n_i} (f_{t^{q_t^i} y^{q_y^i}})^{w_i}(t_m, \eta_m) f^{w_f}(t_m, \eta_m).
\end{aligned}$$

Now we prove that $\widetilde{e}$ has $\min(S - r_0, M)$ degrees of smoothness by induction. Since $\|\widetilde{e}\|_\infty \sim (h)$ is bounded, $\widetilde{e}$ has at least zero degrees of smoothness in the discrete sense. Assume that $\widetilde{e}$ has $s < \min(S - r_0, M)$ degrees of smoothness in the discrete sense. We will prove that $(d_1\widetilde{e})$ has s degrees of smoothness, from which we can conclude that $\widetilde{e}$ has $(s+1)$ degrees of smoothness in the discrete sense. From a similar argument as in the proof of Lemma 2.3.11, $\widetilde{u}$ has s degrees of smoothness in the discrete sense. Assuming that $y(t)$ has S degrees of smoothness, $\widetilde{r_1}$ has $(S - r_0 - 1)$ degrees of smoothness. Since $\widetilde{e}$ has s degrees of smoothness, e has $\min(s + r_0, M)$ degrees of smoothness,

and $\eta = y - e$ has $\min(s + r_0, M)$ degrees of smoothness. Thus, $\widetilde{r_2}$ has $\min(s + r_0, S - r_0 - 1, M)$, at least s, degrees of smoothness in the discrete sense. Therefore, $\widetilde{e}$ has $(s + 1)$ degrees of smoothness, and we can conclude that $\widetilde{e}$ has $\min(S - r_0, M)$ degrees of smoothness in the discrete sense.

□

The following lemma discusses the local truncation error and smoothness of the error for the correction loop of implicit IDC-RK methods.

Lemma 2.3.17. *(Correction step): Let $y(t)$ be the solution to the IVP (2.1) and $y(t)$ has at least S ($S \geq M + 2$) degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (2.86). Suppose $e^{[k-1]} \sim \mathcal{O}(h^{s_{k-1}+1})$ and the rescaled error vector $\widetilde{e}^{[k-1]} = \frac{1}{h^{s_{k-1}}} e^{[k-1]}$ has $M+1-s_{k-1}$ degrees of smoothness in the discrete sense. Then, after the k^{th} correction step, provided an $r_k th$ order implicit RK method is used and $k \leq K_{loop}$,*

1. *The error vector $e^{[k]}$ satisfies*

$$\|e^{[k]}\|_\infty \sim \mathcal{O}(h^{s_k+1}) \tag{2.114}$$

and

2. *The rescaled error vector $\widetilde{e}^{[k]} = \frac{1}{h^{s_k}} e^{[k]}$ has $M+1-s_k$ degrees of smoothness in the discrete sense.*

Further, recall the error equation (2.78)

$$\begin{aligned}(e^{(k-1)})'(t) &= f(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f(t, \eta^{(k-1)}(t)) - \epsilon^{(k-1)}(t) \\ &\doteq F(t, e^{(k-1)}(t)) - \epsilon^{(k-1)}(t).\end{aligned}$$

It can be written as

$$(Q^{(k-1)})'(t) = F(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) \doteq G^{(k-1)}(t, Q^{(k-1)}(t)), \quad (2.115)$$

where $Q^{(k-1)}(t) = e^{(k-1)}(t) + E^{(k-1)}(t)$ and $E^{(k-1)}(t) = \int_0^t \epsilon^{(k-1)}(\tau)d\tau$. The analysis for the correction steps in this section will rely on this form of the error equation (2.115).

Define the rescaled quantities as

$$\begin{aligned} \widetilde{e}^{(k-1)}(t) &= \frac{1}{h^s k-1} e^{(k-1)}(t), \\ \widetilde{Q}^{(k-1)}(t) &= \frac{1}{h^s k-1} Q^{(k-1)}(t), \\ \widetilde{G}^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)) &= \frac{1}{h^s k-1} G^{(k-1)}(t, h^s k-1 \widetilde{Q}^{(k-1)}(t)), \end{aligned} \quad (2.116)$$

and thus (2.78) becomes

$$(\widetilde{Q}^{(k-1)})'(t) = \widetilde{G}^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)). \quad (2.117)$$

In the correction steps of the IDC method, a p-stage, r^{th} order implicit RK method (2.68) applied to (2.117) gives

$$\widetilde{k}_i = \widetilde{G}^{(k-1)}(t_0 + c_i h, \widetilde{Q}_0^{[k-1]} + h \sum_{j=1}^{p} a_{ij} \widetilde{k}_j) \text{ for } i = 1, 2, \ldots, p, \quad (2.118)$$

$$\widetilde{\Omega}_1^{[k]} = \widetilde{Q}_0^{[k-1]} + h \sum_{j=1}^{p} b_j \widetilde{k}_j \quad (2.119)$$

where the Greek letter $\widetilde{\Omega}$ denotes the rescaled solution in the numerical space. In the actual implementation, we discretize (2.78) as mentioned above in (2.83).

Proposition 2.3.18. *If* $e^{[k-1]} \sim \mathcal{O}(h^{s_{k-1}+1})$, *then*

$$Q^{(k-1)}(t) \sim \mathcal{O}(h^{s_{k-1}+1}) \text{ and } G^{(k-1)}(t, Q^{(k-1)}(t)) \sim \mathcal{O}(h^{s_{k-1}+1}). \quad (2.120)$$

Proposition 2.3.19. *Suppose* $e^{[k-1]} \sim \mathcal{O}(h^{s_{k-1}+1})$ *and the rescaled error vector* $\widetilde{e}^{[k-1]} = \frac{1}{h^{s_{k-1}}} e^{[k-1]}$ *has* $M + 1 - s_{k-1}$ *degrees of smoothness in the discrete sense. Then* $\widetilde{Q}^{(k-1)}(t) = \frac{1}{h^{s_{k-1}}} Q^{(k-1)}(t)$ *satisfies*

$$\frac{d^l}{dt^l} \widetilde{Q}^{(k-1)}(t) \sim \mathcal{O}_h(1) \text{ if } l \leq M + 1 - s_{k-1}, \quad \text{for all } t \in [0, H]. \quad (2.121)$$

Proposition 2.3.20. $\widetilde{Q}^{(k-1)}(t)$ *in (2.117) satisfies*

$$\widetilde{Q}_{m+1}^{[k-1]} = \widetilde{Q}_m^{[k-1]} + \sum_{i=1}^{M-s_{k-1}} \frac{h^i}{i!} \widetilde{G}_{i-1}^{(k-1)}(t_m, \widetilde{Q}_m^{[k-1]}) + \mathcal{O}(h^{M+1-s_{k-1}}),$$

or equivalently,

$$\begin{aligned} e_{m+1}^{[k-1]} = e_m^{[k-1]} + h^{s_{k-1}} & \sum_{i=1}^{M-s_{k-1}} \frac{h^i}{i!} \widetilde{G}_{i-1}^{(k-1)}(t_m, \widetilde{Q}_m^{[k-1]}) + \mathcal{O}(h^{M+1}) \\ & - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(\tau) d\tau \end{aligned} \quad (2.122)$$

where

$$\widetilde{G}_{i-1}^{(k-1)}(t, \widetilde{Q}) \doteq \frac{d^{i-1}}{dt^{i-1}} \sum \gamma_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}}^{w_1 \cdots w_{n_i} w_f} \prod_{i=1}^{n_i} (\widetilde{G}_{t^{q_t^i} \widetilde{Q}^{q_{\widetilde{Q}}^i}}^{(k-1)})^{w_i} (\widetilde{G}^{(k-1)})^{w_f}$$

$$\sim \mathcal{O}_h(1), \text{ for } i \leq M - s_{k-1}.$$

As before, $\gamma_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}}^{w_1 \cdots w_{n_i} w_f}$ *are constant coefficients and* w_i, w_f, q_t^i, $q_{\widetilde{Q}}^i$, *and* n_i *are*

nonnegative integers.

The proofs of Propostions 2.3.18, 2.3.19, and 2.3.20 are exactly as given in [18].

The proof of the following proposition is the essential difference between the proofs of the truncation error of the explicit and the implicit cases of an RK method used in the prediction and correction loops of IDC.

Proposition 2.3.21. *The Taylor series for* $\widetilde{e}^{(k-1)}(t_0+h) = \frac{1}{h^{sk-1}} e^{(k-1)}(t_0+h)$ *and for* $\frac{1}{h^{sk-1}}\delta_1^{[k]}$ *in (2.83) coincide up to and including the term* h^r *for a sufficiently smooth error function* $\widetilde{e}^{(k-1)}(t)$ *and assuming the exact solution* $y(t)$ *has* S *degrees of smoothness.*

Proof. First we claim that

$$h^{sk-1}\widetilde{k}_i = k_i + \mathcal{O}(h^{S-1}) \text{ for all } i = 1, 2, \ldots, p, \tag{2.123}$$

and, in particular, setting

$$A_i = h\sum_{j=1}^{p} a_{ij} h^{sk-1}\widetilde{k}_j \text{ and } B_i = h\sum_{j=1}^{p} a_{ij} k_j,$$

we claim that we can write, for all $i = 1, 2, \ldots, p$ and for $n = 2, 3, \ldots, S-1$,

$$\begin{aligned} & h^{sk-1}\widetilde{k}_i - k_i \\ & = h^n \sum_{l_1=1}^{S-2}\sum_{l_2=1}^{S-2}\cdots\sum_{l_n=1}^{S-2}\sum_{j_1=1}^{p}\sum_{j_2=1}^{p}\cdots\sum_{j_n=1}^{p} C_{l_1 i} a_{i j_1} \cdot \\ & \prod_{m=2}^{n} C_{l_m j_{m-1}} a_{j_{m-1} j_m} (h^{sk-1}\widetilde{k}_{j_n} - k_{j_n}) + \mathcal{O}(h^{S-1}), \end{aligned} \tag{2.124}$$

where $C_{lj} = \frac{1}{l!}\frac{\partial^l}{\partial y^l} G(t_0 + c_j h, Q_0) \sum_{v=1}^{l} A_j^{l-v} B_j^{v-1}$.

We prove (2.124), and hence (2.123), by induction on n.

- $n = 2$: Using the definitions (2.116) of the rescaled quantities above and Taylor expanding G about Q_0 in the y-variable, we obtain

$$\begin{aligned} h^s k-1\widetilde{k}_i - k_i &= G(t_0 + c_i h, Q_0 + A_i) - G(t_0 + c_i h, Q_0 + B_i) \\ &= \sum_{l_1=1}^{S-2} \frac{1}{l_1!} \frac{\partial^{l_1}}{\partial y^{l_1}} G(t_0 + c_i h, Q_0)((A_i)^{l_1} - (B_i)^{l_1}) + \mathcal{O}(h^{S-1}) \end{aligned}$$

Factoring $(A_i)^{l_1} - (B_i)^{l_1} = (A_i - B_i) \sum_{v=1}^{l_1} A_i^{l_1 - v} B_i^{v-1}$, we obtain

$$\begin{aligned} h^s k-1\widetilde{k}_i - k_i &= \sum_{l_1=1}^{S-2} C_{l_1 i}(A_i - B_i) + \mathcal{O}(h^{S-1}) \\ &= \sum_{l_1=1}^{S-2} C_{l_1 i} h \sum_{j_1=1}^{p} a_{i j_1}(h^s k-1\widetilde{k}_{j_1} - k_{j_1}) + \mathcal{O}(h^{S-1}), \end{aligned}$$

and repeating the process for $h^s k-1\widetilde{k}_{j_1} - k_{j_1}$,

$$\begin{aligned} &= \sum_{l_1=1}^{S-2} C_{l_1 i} h \sum_{j_1=1}^{p} a_{i j_1} \\ &\left(\sum_{l_2=1}^{S-2} C_{l_2 j_1} h \sum_{j_2=1}^{p} a_{j_1 j_2}(h^s k-1\widetilde{k}_{j_2} - k_{j_2}) + \mathcal{O}(h^{S-1}) \right) \\ &+ \mathcal{O}(h^{S-1}) \\ &= h^2 \sum_{l_1=1}^{S-2} \sum_{l_2=1}^{S-2} \sum_{j_1=1}^{p} \sum_{j_2=1}^{p} C_{l_1 i} C_{l_2 j_1} a_{i j_1} a_{j_1 j_2}(h^s k-1\widetilde{k}_{j_2} - k_{j_2}) \\ &+ \mathcal{O}(h^{S-1}). \end{aligned} \tag{2.125}$$

Thus $h^s k-1\widetilde{k}_i - k_i \sim \mathcal{O}(h^2)$.

- $n+1$: Assume (2.124) holds for $n < S-1$. Following the same process of Taylor

expanding, etc. as in equation (2.125) in the $n = 2$ case for $h^{s}k-1\widetilde{k}_{j_n} - k_{j_n}$, we see that

$$
\begin{aligned}
& h^{s}k-1\widetilde{k}_i - k_i \\
& = h^n \sum_{l_1,\ldots,l_n=1}^{S-2} \sum_{j_1,\ldots,j_n=1}^{p} C_{l_1 i} a_{i j_1} \\
& \prod_{m=2}^{n} C_{l_m j_{m-1}} a_{j_{m-1} j_m} (h^{s}k-1\widetilde{k}_{j_n} - k_{j_n}) + \mathcal{O}(h^{S-1}) \\
& = h^n \sum_{l_1,\ldots,l_n=1}^{S-2} \sum_{j_1,\ldots,j_n=1}^{p} C_{l_1 i} a_{i j_1} \prod_{m=2}^{n} C_{l_m j_{m-1}} a_{j_{m-1} j_m} \\
& \left(\sum_{l_{n+1}=1}^{S-2} C_{l_{n+1} j_n} h \sum_{j_{n+1}=1}^{p} a_{j_n j_{n+1}} (h^{s}k-1\widetilde{k}_{j_{n+1}} - k_{j_{n+1}}) + \mathcal{O}(h^{S-1}) \right) \\
& + \mathcal{O}(h^{S-1}) \\
& = h^{n+1} \sum_{l_1,\ldots,l_{n+1}=1}^{S-2} \sum_{j_1,\ldots,j_{n+1}=1}^{p} C_{l_1 i} a_{i j_1} \\
& \prod_{m=2}^{n+1} C_{l_m j_{m-1}} a_{j_{m-1} j_m} (h^{s}k-1\widetilde{k}_{j_{n+1}} - k_{j_{n+1}}) + \mathcal{O}(h^{S-1}).
\end{aligned}
$$

It follows that (2.123) holds.

Now consider $\delta_1^{[k]}$, which we want equal to the last equation of (2.83).

$$\begin{aligned}
\delta_1^{[k]} &\doteq h^{s}k-1\,\widetilde{\Omega}_1^{[k]} - E^{(k-1)}(t_1) \\
&= h^{s}k-1\,\widetilde{Q}^{(k-1)}(t_0) + h\sum_{j=1}^{p} b_j\, h^{s}k-1\widetilde{k}_j - E^{(k-1)}(t_1) \\
&= Q^{(k-1)}(t_0) - E^{(k-1)}(t_1) + h\sum_{j=1}^{p} b_j\,(k_j + \mathcal{O}(h^{S-1})) \\
&= e_0^{[k-1]} - \int_{t_0}^{t_0+h} \epsilon^{(k-1)}(\tau)d\tau + h\sum_{j=1}^{p} b_j\,k_j + \mathcal{O}(h^S).
\end{aligned}$$

Recall that the Taylor series for $\widetilde{Q}^{(k-1)}(t_0+h)$ and $\widetilde{\Omega}_1^{[k]}$ coincide up to and including the term h^r, since $\widetilde{\Omega}_1^{[k]}$ is an r^{th} order implicit RK approximation to $\widetilde{Q}^{(k-1)}(t_0+h)$.

Hence

$$\frac{1}{h^{s}k-1}Q^{(k-1)}(t_0+h) = \widetilde{Q}^{(k-1)}(t_0+h),$$

and

$$\frac{1}{h^{s}k-1}(e_0^{[k-1]} + E(t_0) + h\sum_{j=1}^{p} b_j\,k_j + \mathcal{O}(h^S)) = \widetilde{\Omega}_1^{[k]},$$

coincide up to and including the h^r term.

Also

$$\widetilde{Q}^{(k-1)}(t_0+h) = \frac{1}{h^{s}k-1}(e^{(k-1)}(t_0+h) + E^{(k-1)}(t_0+h)),$$

and

$$\widetilde{\Omega}_1^{[k]} = \frac{1}{h^{s_k-1}}(\delta_1^{[k]} + E^{(k-1)}(t_0+h) + \mathcal{O}(h^S)).$$

Since $\widetilde{Q}^{(k-1)}(t_0+h) - \widetilde{\Omega}_1^{[k]} = \mathcal{O}(h^{r+1})$, then

$$\mathcal{O}(h^{r+1}) = \frac{1}{h^{s_k-1}} e^{(k-1)}(t_0+h) - \frac{1}{h^{s_k-1}}\delta_1^{[k]} + \mathcal{O}(h^{S-s_k-1}),$$

or

$$\frac{1}{h^{s_k-1}} e^{(k-1)}(t_0+h) - \frac{1}{h^{s_k-1}}\delta_1^{[k]} = \mathcal{O}(h^{r+1}) + \mathcal{O}(h^{S-s_k-1}).$$

Now since

$$S \geq M+2 > M+1 \geq s_{K_{loop}} = \sum_{j=0}^{K_{loop}} r_j,$$

then

$$S - s_{k-1} > \sum_{j=k}^{K_{loop}} r_j \geq r_k,$$

so $S - s_{k-1} \geq r_k + 1$.

This proves Proposition 2.3.21. □

Proposition 2.3.22. *Suppose we solve the error equation* (2.78) *using algorithm*

(2.83). Then the numerical error vector is

$$\begin{aligned}\delta_{m+1}^{[k]} &= \delta_m^{[k]} + h^{s_{k-1}+1}\,\widetilde{G}^{(k-1)}(t_m, \widetilde{\Omega}_m^{[k]}) + \dots \\ &\quad + \frac{h^{s_{k-1}+r_k}}{r_k!}\widetilde{G}^{(k-1)}_{r_k-1}(t_m, \widetilde{\Omega}_m^{[k]}) + h^{s_{k-1}}\,\widetilde{R}^{(k-1)}_{r_k+1}(t_m, \widetilde{\Omega}_m^{[k]}) \\ &\quad - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(\tau)d\tau + \mathcal{O}(h^{M+1}),\end{aligned} \tag{2.126}$$

where $\widetilde{\Omega}_m^{[k]} = \frac{1}{h^{s_{k-1}+1}}(\delta_m^{[k]} + \int_0^{t_m} \epsilon^{(k-1)}(\tau)d\tau)$ *and*

$$\widetilde{R}^{(k-1)}_{r_k+1}(t, \widetilde{\Omega}) = \sum_{q=r_k+1}^{M-s_{k-1}} h^q \left(\sum_{q_t^i + q_{\widetilde{Q}}^i \le q-1} \zeta^{w_1\cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}} \prod {}_{t^{q_t^i}\widetilde{Q}^{q_{\widetilde{Q}}^i}}(\widetilde{G}^{(k-1)}(t, \widetilde{\Omega}))^{w_i} (\widetilde{G}^{(k-1)}(t, \widetilde{\Omega}))^{w_f} \right),$$

with ${}_{t^{q_t^i}\widetilde{Q}^{q_{\widetilde{Q}}^i}}\widetilde{G}^{(k-1)}(t, \widetilde{\Omega}) \sim \mathcal{O}_h(1)$ *for* $q_t^i + q_{\widetilde{Q}}^i \le M - s_{k-1}$. *Here* $\zeta^{w_1\cdots w_{n_i} w_f}_{q_y^1 q_t^1 \cdots q_y^{n_i} q_t^{n_i}}$ *are constant coefficients determined by (2.83).*

Remark 2.3.23. *In implementing algorithm (2.83), one needs to evaluate* $f(t, \eta(t))$ *at some intermediate stage, e.g., at* $t = t_0 + c_j h$. *This results in additional function evaluations of* $f(t, \eta(t))$, *which is usually the most expensive part of an algorithm. In our implementation, we avoid this extra cost by performing a polynomial interpolation. Specifically, given* $f = (f(t_0, \eta_0), \dots, f(t_m, \eta_m), \dots, f(t_M, \eta_M))$, *we construct an* M^{th} *degree Lagrange interpolant,* $L^M(t, f)$ *and approximate* $f(t, \eta(t))|_{t=t_0+c_j h}$ *using* $L^M(t = t_0 + c_j h, f)$ *up to* $\mathcal{O}(h^{M+1})$.

Remark 2.3.24. *To evaluate the integral term in the right hand side of algorithm (2.83), we construct a Lagrange interpolant* $L^M(t, \epsilon^{(k-1)})$ *and use*

$\int_{t_0}^{t_0+c_ih} L^M(t, \epsilon^{(k-1)}(t))dt$, $i = 1, 2, \ldots, p$,
to approximate the integral term $\int_{t_0}^{t_0+c_ih} \epsilon^{(k-1)}(t)dt$, $i = 1, 2, \ldots, p$, *with an error* $\mathcal{O}(h^{M+2})$.

Proof. (of Lemma 2.3.17). The proof is identical to that given in Section 5.2.3 of [18], but we repeat it here for completeness.

1. First we prove part 1. of the lemma. By subtracting the numerical error vector, (2.126), from the integrated error equation, (2.122), we obtain

$$\begin{aligned}
e_{m+1}^{[k]} &= e_{m+1}^{[k-1]} - \delta_{m+1}^{[k-1]} \\
&= e_m^{[k]} + h^s k-1 \sum_{i=r_k+1}^{M+1-s_{k-1}} \frac{h^i}{i!} \widetilde{G}_{i-1}^{(k-1)}(t_m, \widetilde{Q}_m^{[k-1]}) \\
&\quad h^s k-1 \sum_{i=1}^{r_k} \frac{h^i}{i!} \left(\widetilde{G}_{i-1}^{(k-1)} \left(t_m, \frac{e_m^{[k-1]} + E^{(k-1)}(t_m)}{h^s k-1} \right) \right. \\
&\quad \left. -\widetilde{G}_{i-1}^{(k-1)} \left(t_m, \frac{\delta_m^{[k-1]} + E^{(k-1)}(t_m)}{h^s k-1} \right) \right) \\
&\quad - h^s k-1 \widetilde{R}_{r_k+1} \left(t_m, \frac{\delta_m^{[k-1]} + E^{(k-1)}(t_m)}{h^s k-1} \right) + \mathcal{O}(h^{M+2}) \\
&= e_m^{[k]} + u_m^{[k]} + r_{1,m}^{[k-1]} - r_{2,m}^{[k]} + \mathcal{O}(h^{M+2}),
\end{aligned} \tag{2.127}$$

where

$$
\begin{aligned}
u_m^{[k]} &= h^s k-1 \sum_{i=1}^{r_k} \frac{h^i}{i!} \left(\widetilde{G}_{i-1}^{(k-1)} \left(t_m, \frac{e_m^{[k-1]} + E^{(k-1)}(t_m)}{h^s k-1} \right) \right. \\
&\quad \left. -\widetilde{G}_{i-1}^{(k-1)} \left(t_m, \frac{\delta_m^{[k-1]} + E^{(k-1)}(t_m)}{h^s k-1} \right) \right) \\
&= \sum_{i=1}^{r_k} \frac{h^i}{i!} \frac{d^{i-1}}{dt^{i-1}} \left(\widetilde{G}^{(k-1)} \left(t_m, e_m^{[k-1]} + E^{(k-1)}(t_m) \right) \right. \\
&\quad \left. -\widetilde{G}^{(k-1)} \left(t_m, \delta_m^{[k-1]} + E^{(k-1)}(t_m) \right) \right) \\
&= \sum_{i=1}^{r_k} \frac{h^i}{i!} \frac{d^{i-1}}{dt^{i-1}} \left(f(t_m, y(t_m)) - f\left(t_m, \eta^{(k)}(t_m) \right) \right) \\
&= \sum_{i=1}^{r_k} \frac{h^i}{i!} h^{i-1} \frac{d^{i-1}}{dt^{i-1}} \\
&\quad \left(\sum_{j=1}^{S-2} \frac{(-1)^{j+1}}{j!} \frac{\partial^j f}{\partial y^j}(t_m, y(t_m)) (e_m^{[k]})^j + \mathcal{O}((e_m^{[k]})^{S-1}) \right),
\end{aligned}
\tag{2.128}
$$

and since $s_k = s_{k-1} + r_k$,

$$r_{1,m}^{[k-1]} = h^{s_{k-1}} \sum_{i=r_k+1}^{M+1-s_{k-1}} \frac{h^i}{i!} \widetilde{G}_{i-1}(t_m, \widetilde{Q}_m^{[k-1]}) \tag{2.129}$$

$$= h^{s_k+1} \sum_{i=0}^{M-s_k} \frac{1}{(r_k+i+1)!} h^i \frac{d^i}{dt^i} \left(\frac{d^{r_k}}{dt^{r_k}} \widetilde{G}(t_m, \widetilde{Q}_m^{[k-1]}) \right),$$

$$r_{2,m}^{[k-1]} = h^{s_{k-1}} \widetilde{R}_{r_k+1} \left(t_m, \frac{\delta_m^{[k-1]} + E^{(k-1)}(t_m)}{h^{s_{k-1}}} \right) \tag{2.130}$$

$$= h^{s_k+1} \sum_{i=0}^{M+1-s_k} h^i \sum_{q_t^i + q_{\widetilde{Q}}^i \le i+r_k} \zeta_{q_{\widetilde{Q}}^1 q_t^1 \cdots q_{\widetilde{Q}}^{n_i} q_t^{n_i}}^{w_1 \cdots w_{n_i} w_f}$$

$$\cdot \prod_{t^{q_t^i} \widetilde{Q}^{q_{\widetilde{Q}}^i}} (\widetilde{G}^{(k-1)}(t, \widetilde{\Omega}))^{w_i} (\widetilde{G}^{(k-1)}(t, \widetilde{\Omega}))^{w_f}$$

Now we may bound $\|e^{[k-1]}\|_\infty$ using an inductive argument. By definition, $e_0^{[k]} = 0$, so clearly $e_0^{[k]} \sim \mathcal{O}(h^{s_k+1})$. Assume that $e_m^{[k]} \sim \mathcal{O}(h^{s_k+1})$. From (2.128), $u^{[k]} \sim \mathcal{O}(h^{s_k+2})$. By Proposition 2.3.19, $\frac{d^{r_k}}{dt^{r_k}} \widetilde{G}(t_m, \widetilde{Q}_m^{[k-1]}) = \frac{d^{r_k+1}}{dt^{r_k+1}} \widetilde{Q}^{(k-1)}(t)|_{t=t_m} \sim \mathcal{O}(1)$. Therefore, $r_{1,m}^{[k-1]} \sim \mathcal{O}(h^{s_k+1})$ from (2.129), and $r_{2,m}^{[k]} \sim \mathcal{O}(h^{s_k+1})$ from (2.130) and Proposition 2.3.22. Thus $e_{m+1}^{[k]} \sim \mathcal{O}(h^{s_k+1})$, and the inductive proof is complete.

2. Now we prove part 2. of the lemma, the smoothness of the rescaled error vector, using an inductive argument based on s, the degree of smoothness of $\widetilde{e}^{[k]}$. First, the rescaled error vector has at least 0 degrees of smoothness since $\|\widetilde{e}^{[k]}\|_\infty \sim \mathcal{O}(h) < \infty$. Assume that $\widetilde{e}^{[k]}$ has $s < M + 2 - s_k$ degrees of smoothness in the discrete sense. We will prove that $(d_1 \widetilde{e}^{[k]})$ has s degrees of smoothness, from which we can then conclude that $\widetilde{e}^{[k]}$ has $(s+1)$ degrees of smoothness. Using (2.127), the divided difference approximation to the

derivative of the rescaled error vector satisfies

$$(d_1\widetilde{e}^{[k]})_m = \frac{\widetilde{u}_m^{[k]}}{h} + \frac{1}{h}\widetilde{r}_{1,m}^{[k-1]} - \frac{1}{h}\widetilde{r}_{2,m}^{[k]} + \mathcal{O}(h^{M+2-s_k}),$$

where

$$\begin{aligned}
\frac{\widetilde{u}_m^{[k]}}{h} &= \sum_{i=1}^{r_k} \frac{h^{i-1}}{i!}\frac{d^{i-1}}{dt^{i-1}} \sum_{j=1}^{S-2} \frac{(-1)^{j+1}h^{s_k(j-1)}}{j!}\frac{\partial_j f}{\partial y^j}(t_m, y(t_m))(\widetilde{e}_m^{[k]})^j \\
&\quad + \mathcal{O}(h^{(s_k+1)(S-2)-1}), \\
\frac{\widetilde{r}_{1,m}^{[k-1]}}{h} &= \sum_{i=0}^{S-r_k-1} \frac{h^i}{(r_k+i+1)!}\frac{d^i}{dt^i}\left(\frac{d^{r_k}}{dt^{r_k}}\widetilde{G}(t_m, \widetilde{Q}_m^{[k-1]})\right), \\
\frac{\widetilde{r}_{2,m}^{[k]}}{h} &= \sum_{i=0}^{M+2-s_k} h^i \sum_{q_t^i + q_{\widetilde{Q}}^i \le i+r_k} \zeta_{q_{\widetilde{Q}}^1 q_t^1 \cdots q_{\widetilde{Q}}^{n_i} q_t^{n_i}}^{w_1 \cdots w_{n_i} w_f} \\
&\quad \cdot \prod_{t^{q_t^i}\widetilde{Q}^{q_{\widetilde{Q}}^i}} (\widetilde{G}^{(k-1)}(t,\widetilde{\Omega}))^{w_i} (\widetilde{G}^{(k-1)}(t,\widetilde{\Omega}))^{w_f},
\end{aligned}$$

are computed from (2.128), (2.129), and (2.130).

Similarly to previous arguments, $\frac{1}{h}\widetilde{u}^{[k]}$ has s degrees of smoothness in the discrete sense. Also, $\frac{1}{h}\widetilde{r}_1^{[k]}$ has $(M - s_k)$ degrees of smoothness since $\frac{d^{r_k}}{dt^{r_k}}\widetilde{G}(t_m, \widetilde{Q}_m^{[k-1]}) = \frac{d^{r_k+1}}{dt^{r_k+1}}\widetilde{Q}^{(k-1)}(t)|_{t=t_m}$ has $(M-s_k)$ degrees of smoothness. Since $\frac{1}{h}\widetilde{r}_2^{[k]}$ also has $\min(M-s_k, s)$ degrees of smoothness in the discrete sense, we can conclude that $(d_1\widetilde{e}^{[k]})$ has s degrees of smoothness in the discrete sense, which implies that $\widetilde{e}^{[k]}$ has $(M + 1 - s_k)$ degrees of smoothness in the discrete sense, completing the inductive proof.

□

Proof. (of Theorem 2.3.13) We prove Theorem 2.3.13 by induction w.r.t. k, the

index of the correction step in an SDC method. By Lemma 2.3.14, Theorem 2.3.13 is valid for $k = 0$. Assume that Theorem 2.3.13 is valid for $(k-1)$ loops of the correction step, and by Lemma 2.3.17, Theorem 2.3.13 is also valid for k correction loops. $\square$

Chapter 3

Semi–implicit IDC-ARK Methods

3.1 Semi–implicit Integration

For many computational implementations of the above problems in Chapter 1, one must solve many systems of ODEs that involve more than one time scale both accurately and efficiently, either from the original formulation of the problem as an ODE system, or as a result of a method of lines approach for PDEs, or because of employing some other special framework (e.g., as in Section 1.3). Consider a simplified case where the right hand side of an ODE can be split into two parts, one stiff (fast) scale, and one nonstiff (slow) scale.

$$\begin{aligned} y'(t) &= f_S(t,y) + f_N(t,y), \quad t \in [0,T], \\ y(0) &= y_0, \end{aligned} \tag{3.1}$$

where $f_S(t,y)$ contains stiff terms and $f_N(t,y)$ contains the nonstiff terms. Applying explicit numerical integrators to the above initial value problem (IVP) can result in impractical time step restrictions due to the stiff term, while implicit methods may involve costly Newton iterations. Semi–implicit, or implicit–explicit (IMEX)

time integrators, however, can be used to solve the stiff part implicitly and the nonstiff part explicitly, thereby gaining the benefit of the implicit method for the stiffness but potentially saving computational effort by handling some terms explicitly. Popular IMEX integrators include additive Runge-Kutta (ARK) methods, which combine an implicit and an explicit Runge-Kutta method, and integrators constructed from linear multi-step methods, such as Adams or BDF methods. Currently, ARK methods above 5th order have not been constructed without the use of a defect correction framework [45]. Semi–implicit methods constructed from linear multi-step methods require multiple starting values, and their stabilities deteriorate at higher order [36, 4, 48]. We construct arbitrary order semi-implicit integrators by incorporating ARK methods into the IDC framework, and denote these new semi-implicit integrators as IDC-ARK methods. A detailed description of ARK methods follows in Section 3.2, before introducing the construction of IDC-ARK methods.

3.2 Additive Runge–Kutta Methods

One method of splitting an ODE is to partition the right hand side of an IVP into Λ parts [20, 1, 45, 56]:

$$y'(t) = f(t,y) = \sum_{\nu=1}^{\Lambda} f_{[\nu]}(t,y), \quad t \in [0,T] \tag{3.2}$$
$$y(0) = y_0$$

and to treat each $f_{[\nu]}$ with a separate numerical method. When different p-stage Runge-Kutta integrators are applied to each $f_{[\nu]}$, the entire numerical method is called an additive Runge-Kutta (ARK) method. If we define the numerical solution after one timestep h as η_1, which is an approximation to the exact solution $y(t_0+h)$,

then one step of a p-stage ARK method is given by

$$Y_i = y_0 + h \sum_{\nu=1}^{\Lambda} \sum_{j=1}^{p} a_{ij}^{[\nu]} f_{[\nu]}(t_0 + c_j^{[\nu]} h, Y_j),$$
$$\eta_1 = y_0 + h \sum_{\nu=1}^{\Lambda} \sum_{i=1}^{p} b_i^{[\nu]} f_{[\nu]}(t_0 + c_i^{[\nu]} h, Y_i), \tag{3.3}$$

where the coefficients are presented in the Butcher table in the case of $c_j^{[\nu]} = c_j$ for all $\nu = 1, \ldots, \Lambda$:

$$\begin{array}{c|cccc|c|cccc}
c_1 & a_{11}^{[1]} & a_{12}^{[1]} & \cdots & a_{1p}^{[1]} & & a_{11}^{[\Lambda]} & a_{12}^{[\Lambda]} & \cdots & a_{1p}^{[\Lambda]} \\
c_2 & a_{21}^{[1]} & a_{22}^{[1]} & \cdots & a_{2p}^{[1]} & & a_{21}^{[\Lambda]} & a_{22}^{[\Lambda]} & \cdots & a_{2p}^{[\Lambda]} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\
c_p & a_{p1}^{[1]} & a_{p2}^{[1]} & \cdots & a_{pp}^{[1]} & & a_{p1}^{[\Lambda]} & a_{p2}^{[\Lambda]} & \cdots & a_{pp}^{[\Lambda]} \\
\hline
 & b_1^{[1]} & b_2^{[1]} & \cdots & b_p^{[1]} & \cdots & b_1^{[\Lambda]} & b_2^{[\Lambda]} & \cdots & b_p^{[\Lambda]}
\end{array}$$

Under certain conditions on the RK coefficients, an ARK method of desired order can be obtained without splitting error [21, 1, 56, 45].

Without loss of generality, we consider the case $\Lambda = 2$, where the IVP (3.2) simplifies to the IVP (3.1) where one part, $f_S(t, y)$, is stiff, and the other, $f_N(t, y)$, is nonstiff. An IMEX version of ARK is then applied to (3.1), as shown for example

in the Butcher table,

$$\begin{array}{c|cccc|cccc|c}
0 & 0 & & & & 0 & & & & \\
c_2 & a_{21}^S & a_{22}^S & & & a_{21}^N & 0 & & & \\
\vdots & \vdots & \vdots & \ddots & & \vdots & & \ddots & & \\
c_p & a_{p1}^S & a_{p2}^S & \cdots & a_{pp}^S & a_{p1}^N & a_{p2}^N & \cdots & a_{p,p-1}^N & 0 \\
\hline
 & b_1^S & b_2^S & \cdots & b_p^S & b_1^N & b_2^N & \cdots & b_{p-1}^N & b_p^N
\end{array} \tag{3.4}$$

where to reduce computational cost, an implicit DIRK method, with zero in the first diagonal entry, is applied to f_S, and an explicit method is applied to f_N. Note that, although one can formulate an ARK method with different stage nodes, say c_j^N and c_j^S, taking the same stage nodes, $c_j \doteq c_j^N = c_j^S$, simplifies the order conditions, as presented in [45].

In the literature, the definition of ARK for the case of $\Lambda = 2$ usually is presented in the form of (3.3) [1, 45, 20, 56]. We reformulate the definition of ARK ($\Lambda = 2$) below to suit our analysis. The equivalence of definitions can be seen by simple substitution. Our alternate definition can be given as follows:

Definition 3.2.1. *Let p denote the number of stages and a_{ij}^N, a_{ij}^S; b_j^N, b_j^S; c_j be real coefficients. Then the method*

$$\begin{aligned}
k_i^\alpha &= f_\alpha(t_0 + c_i h, y_0 + h(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p} a_{ij}^S k_j^S)), \\
\eta_1 &= y_0 + h \sum_{i=1}^{p} b_i^N k_i^N + b_i^S k_i^S,
\end{aligned} \tag{3.5}$$

for $\alpha = N, S$ and $i = 1, 2, \ldots, p$, is a p-stage ARK method for solving the IVP (3.1).

Definition 3.2.2. *An ARK method has order r if for a sufficiently smooth IVP* (3.1),

$$\|y(t_0 + h) - \eta_1\| \leq Kh^{r+1},$$

i.e., the Taylor series for the exact solution $y(t_0 + h)$ and η_1 coincide up to and including the term h^r.

3.3 Semi–implicit IDC-ARK Methods

Now we present a brief description of the formulation of IDC methods with ARK base schemes. Then we provide both a general formulation that constructs arbitrary order semi-implicit IDC methods incorporating (arbitrary order) ARK integrators and an example involving a second order ARK integrator.

3.3.1 General Formulation of IDC-ARK

Generalizing the formulation from [18] for IDC-RK methods, we obtain a general framework for semi-implicit IDC-ARK integrators.

Consider the IVP (3.1), where the right hand side is split into stiff and nonstiff parts, $f_S(t, y)$ and $f_N(t, y)$, as in Section 3.2. The time interval, $[0, T]$, is discretized into intervals $[t_n, t_{n+1}]$, $n = 0, 1, \ldots, N$, such that

$$0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{3.6}$$

with timestep $H_n = t_{n+1} - t_n$. Each interval $[t_n, t_{n+1}]$ is discretized again into

M uniform subintervals with quadrature nodes denoted by

$$t_{n,m} = t_n + mh,\ m = 0, 1, \ldots, M, \tag{3.7}$$

where $h = \frac{H_n}{M}$. We apply the IDC method on each time interval $[t_n, t_{n+1}]$ and drop the subscript n since the same method is applied to each interval. Although it is possible to use nonuniform quadrature nodes in place of (3.7), we anticipate that nonuniform nodes will not produce the desired order of accuracy results, based on the analysis in [17]. Note that this choice of nodes does not preclude the effectiveness of nonuniform timesteps H_n; for example, we implement adaptive IDC-ARK methods in Chapter 4, which obviously requires adjusting the size of H_n.

A summary of the IDC-ARK algorithm is as follows. First, in the prediction loop, we compute a provisional solution to the IVP (3.1) with an r_0th order ARK method. Then for $k = 1, \ldots, K_{loop}$ correction loops, we compute an approximation to the error using an r_kth order ARK method and update the provisional solution with this approximate error. Thus an IDC method with order of accuracy $r_0 + \cdots + r_k + \cdots + r_{K_{loop}} \leq M + 1$ is obtained.

Next we provide the details that are necessary for both the implementation of the IDC-ARK method and the proof of truncation error in Section 3.4.

- **Prediction loop:** Use an r_0th order numerical method to obtain a provisional solution to the IVP (3.1)

  $$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]}),$$

  which is an r_0th order approximation to the exact solution

  $$y = (y_0, y_1, \ldots, y_m, \ldots, y_M),$$

where $y_m = y(t_m)$ is the exact solution at t_m, for $m = 0, 1, \ldots, M$. We apply a p_0-stage r_0th order ARK scheme (3.5) as follows, for $\alpha = N, S$; $i = 1, 2, \ldots, p_0$ and $m = 0, 1, \ldots, M-1$:

$$k_i^\alpha = f_\alpha(t_m + c_i h, \eta_m^{[0]} + h(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_0} a_{ij}^S k_j^S)),$$

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h \sum_{i=1}^{p_0} (b_i^N k_i^N + b_i^S k_i^S).$$

- **Correction loop:** Use the error function to improve the order of accuracy of the numerical solution at each iteration.

  For $k = 1$ to K_{loop} (K_{loop} is the number of correction loops):

  1. Denote the *error function* from the previous step as

  $$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{3.8}$$

  where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Note that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

  2. Denote the *residual function* as

  $$\epsilon^{(k-1)}(t) \doteq (\eta^{(k-1)})'(t) - f_S(t, \eta^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)),$$

and compute the integral of the residual. For example,

$$\int_{t_0}^{t_{m+1}} \epsilon^{(k-1)}(\tau)\, d\tau \tag{3.9}$$
$$\approx \eta_{m+1}^{[k-1]} - y_0 - \sum_{j=0}^{M} \gamma_{m,j}(f_S(t_j, \eta_j^{[k-1]}) + f_N(t_j, \eta_j^{[k-1]})),$$

where $\gamma_{m,j}$ are the coefficients that result from approximating the integral by interpolatory quadrature formulas, as in [25] (or as described in Section 2.2, $\gamma_{m,j}$ equals the quantity given in equation (2.32)), and $\eta_j^{[k-1]} = \eta^{(k-1)}(t_j)$.

3. Compute the *numerical error vector*, denoted by

$$\delta^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]}), \tag{3.10}$$

which is an r_kth order approximation to the error

$$e^{[k-1]} = (e_0^{[k-1]}, \ldots, e_m^{[k-1]}, \ldots, e_M^{[k-1]}),$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function (3.8) at t_m.

To compute $\delta^{[k]}$ by an r_kth order ARK method, we first discretize the

integral form of the *error equation*,

$$
\begin{aligned}
(Q^{(k-1)})'(t) &= f_S(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \\
&\quad + f_N(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)) \\
&= f_S(t, \eta^{(k-1)}(t) + Q^{(k-1)}(t) - E^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \\
&\quad + f_N(t, \eta^{(k-1)}(t) + Q^{(k-1)}(t) - E^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)) \\
&\doteq F_S(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) + F_N(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) \\
&\doteq G_S(t, Q^{(k-1)}(t)) + G_N(t, Q^{(k-1)}(t)), \\
Q^{(k-1)}(0) &= 0,
\end{aligned}
\tag{3.11}
$$

where

$$
\begin{aligned}
Q^{(k-1)}(t) &= e^{(k-1)}(t) + E^{(k-1)}(t), \qquad (3.12) \\
F_\alpha(t, e^{(k-1)}(t)) &= f_\alpha(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_\alpha(t, \eta^{(k-1)}(t)), \\
\alpha &= N, S, \\
E^{(k-1)}(t) &= \int_{t_0}^{t} \epsilon^{(k-1)}(\tau)\, d\tau.
\end{aligned}
$$

The f_α, $\alpha = N, S$ notation above (with Q and the approximation for E) is used in the numerical computation, while the G_α, $\alpha = N, S$ notation is employed in the theoretical explanations and proof in Section 3.4.

We apply a p_k-stage r_kth order ARK method to (3.11) and denote the r_kth order numerical approximation to $Q_m^{[k-1]} = Q^{(k-1)}(t_m)$ by $\Omega_m^{[k]}$.

We therefore have, for $\alpha = N, S$, $i = 1, 2, \ldots, p_k$ and $m = 0, 1, \ldots, M-1$:

$$\begin{aligned} k_i^\alpha &= G_\alpha\big(t_m + c_i h, \Omega_m^{[k]} + h\,(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_k} a_{ij}^S k_j^S)\big) \\ &= F_\alpha\big(t_m + c_i h, \Omega_m^{[k]} \\ &\quad + h\,(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_k} a_{ij}^S k_j^S) - E^{[k-1]}(t_m + c_j h)\big), \end{aligned} \tag{3.13}$$

$$\Omega_{m+1}^{[k]} = \Omega_m^{[k]} + h \sum_{i=1}^{p_k} (b_i^N k_i^N + b_i^S k_i^S). \tag{3.14}$$

From equations (3.12), we compute our numerical error vector (3.10),

$$\delta^{[k]} = \Omega^{[k]} - E^{[k-1]},$$

where the components of $E^{[k-1]}$ are $E_m^{[k-1]} = E^{(k-1)}(t_m)$. In fact, we use

$$\begin{aligned} \delta_{m+1}^{[k]} &= \Omega_{m+1}^{[k]} - \int_{t_0}^{t_{m+1}} \epsilon^{(k-1)}(\tau)\, d\tau \\ &\overset{(3.9)}{\approx} \Omega_{m+1}^{[k]} \\ &\quad - \Big(\eta_{m+1}^{[k-1]} - y_0 - \sum_{j=0}^{M} \gamma_{m,j}\big(f_S(t_j, \eta_j^{[k-1]}) + f_N(t_j, \eta_j^{[k-1]})\big)\Big), \end{aligned}$$

where we have approximated the integral by interpolatory quadrature formulas, as in [25].

In computing (3.14), one needs to evaluate $f_\alpha(t, \eta(t))$, $\alpha = N, S$, at some intermediate stage, e.g., at $t = t_0 + c_j h$. This results in additional function evaluations of $f_\alpha(t, \eta(t))$, which is usually the most expensive part of an algorithm. In our implementation, we avoid this extra cost by

performing a polynomial interpolation. Specifically, given

$$f_\alpha = (f_\alpha(t_0, \eta_0), \ldots, f_\alpha(t_m, \eta_m), \ldots, f_\alpha(t_M, \eta_M)),$$

we construct an M^{th} degree Lagrange interpolant, $L^M(t, f_\alpha)$ and approximate $f_\alpha(t, \eta(t))|_{t=t_0+c_j h}$ using $L^M(t = t_0+c_j h, f_\alpha)$ up to $\mathcal{O}(h^{M+1})$. Again note that f_α, F_α are more useful for understanding the numerical implementation, while G_α is employed in the theory in Section 3.4.

4. Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

3.3.2 Example with Second Order ARK

Now we present an example where a second order ARK scheme is employed in the IDC framework. We consider the IMEX RK2 scheme from [3], and illustrate that our formulation is equivalent to Layton's formulation [47] for this specific IMEX RK2 scheme. Writing the ARK scheme (we denote it by ARK2ARS) in a Butcher table, we have:

$$\begin{array}{c|ccc|cc} 0 & 0 & & & & \\ g & 0 & & & g & \\ 1 & 0 & 1-g & g & d & 1-d \\ \hline & 0 & 1-g & g & 0 & 1-g \quad g \end{array}$$

where $g = 1 - \frac{\sqrt{2}}{2}$ and $d = -\frac{2\sqrt{2}}{3}$.

The prediction loop solving (3.1), for $\alpha = N, S$ and $m = 0, 1, \ldots, M-1$, is:

$$\begin{aligned}
k_1^\alpha &= f_\alpha(t_m, \eta_m^{[0]}),\\
k_2^\alpha &= f_\alpha(t_m + gh, \eta_m^{[0]} + hg(k_1^N + k_2^S)),\\
k_3^\alpha &= f_\alpha(t_{m+1}, \eta_m^{[0]} + h(dk_2^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S)),\\
\eta_{m+1}^{[0]} &= \eta_m^{[0]} + h((1-g)(k_1^N + k_2^S) + g(k_3^N + k_3^S)),
\end{aligned}$$

The correction loop solving (3.11) is:

$$\begin{aligned}
k_1^\alpha &= F_\alpha(t_m, \Omega_m^{[k]} - E^{[k-1]}(t_m)),\\
k_2^\alpha &= F_\alpha(t_m + gh, \Omega_m^{[k]} + hg(k_1^N + k_2^S) - E^{[k-1]}(t_m + gh)),\\
k_3^\alpha &= F_\alpha(t_{m+1}, \Omega_m^{[k]} + h(dk_1^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S)\\
&\quad - E^{[k-1]}(t_{m+1})),\\
\Omega_{m+1}^{[k]} &= \Omega_m^{[k]} + h((1-g)(k_2^N + k_2^S) + g(k_3^N + k_3^S)),\\
\delta_{m+1}^{[k]} &= \Omega_{m+1}^{[k]} - E^{[k-1]}(t_{m+1}).
\end{aligned}$$

Letting $\eta_m^{[k]} = \delta_m^{[k]} + \eta_m^{[k-1]}$ and $\Omega_m^{[k]} = \delta_m^{[k]} + E_m^{[k-1]}$, and noting $\mathcal{Q}_m^{m+c}(f_S(t, \eta^{(k-1)}(t)) + f_N(t, \eta^{(k-1)}(t)))$ from [47] is an approximation of $\int_{t_m}^{t_m+ch} {}_\epsilon(k-1)(\tau)d\tau$, we see that the formulation of IDC constructed with ARK2ARS in [47] is equivalent to ours, namely:

$$\begin{aligned}
\phi_{m+g}^{(1)} &= \eta_{m+g}^{[k-1]} + \Omega_m^{[k]} + hg(k_1^N + k_2^S) - E^{[k-1]}(t_m + gh),\\
\phi_{m+1}^{(2)} &= \eta_{m+1}^{[k-1]} + \Omega_m^{[k]} + h(dk_1^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S) - E^{[k-1]}(t_{m+1}),\\
\eta_{m+1}^{[k]} &= \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k]}.
\end{aligned}$$

The notation on the left is as in [47], while the notation on the right is ours. Although

the two formulations are equivalent for this type of second order ARK method, the incorporation of arbitrary order ARK methods in the corrections is unclear in [47] but is obvious in equation (3.14).

3.4 Theoretical Analysis of IDC-ARK Methods

In this section the analysis for semi-implicit IDC-ARK methods is presented. Recall some mathematical preliminaries related to the smoothness of discrete data sets were introduced in Section 2.3.2. The local truncation error estimates of semi-implicit IDC-ARK are provided, along with a corollary for the local truncation error of implicit IDC-RK integrators, as an alternative to the presentation in Section 2.3.2. This paper closely follows the formulation presented in [18], as well as Section 2.3.2. It is recommended that the interested reader who wishes to understand the analysis in this work first reads Section 2.3.2, which presents some mathematical preliminaries regarding the smoothness of discrete and continuous data sets,and the basic framework for analyzing the local truncation error.

The following theorem states that, with the assumption that the IVP is smooth enough, using an rth order ARK method in the correction loops results in an r-order increase in accuracy of the IDC method, but the overall IDC order of accuracy will not be higher than $M+1$, the number of nodes in the subinterval $[t_n, t_{n+1}]$. E.g., if an rth order ARK method is used in each IDC loop, then the IDC method has order

$$\min(r * (1 \textit{ prediction } + K_{loop} \textit{ corrections}), M+1).$$

Theorem 3.4.1. *Let $y(t)$, the solution to the IVP (3.1), have at least σ (and f_S and f_N have at least $\sigma - 1$) degrees of smoothness in the continuous sense, where*

$\sigma \geq M+2$. Consider one time interval of an IDC method with $t \in [0, H]$ and $M+1$ uniformly distributed quadrature points (3.7). Suppose an r_0th order ARK method (3.5) *is used in the prediction step and $(r_1, r_2, \ldots, r_{K_{loop}})$th order ARK methods are used in K_{loop} correction steps. For simplicity, assume that $c_j \doteq c_j^N = c_j^S$ and the number of stages for the implicit and explicit parts of each ARK method is the same. Let $s_k = \sum_{j=0}^{k} r_j$. If $s_{K_{loop}} \leq M+1$, then the local truncation error is of order $\mathcal{O}(h^{s_{K_{loop}}+1})$.*

The proof of Theorem 3.4.1 follows by induction from Lemmas 3.4.3 and 3.4.4, below, for the prediction and correction steps, respectively, which discuss the local truncation error and smoothness of the rescaled error when general high order ARK schemes are applied in the prediction and correction loops of IDC methods. Lemma 3.4.3 is the first case, and Lemma 3.4.4 is the induction step. Note that the smoothness results of the rescaled error vectors in both lemmas are essential for the proof of the correction loop in Lemma 3.4.4. The smoothness proofs of Lemmas 3.4.3 and 3.4.4 (analogous to those in [18] and in Section 2.3.2) are quite technical. Since they are presented earlier for the implicit IDC-RK methods, here we simply state and sketch a proof for the smoothness of the rescaled error vector following a forward-backward Euler (FEBE) prediction step in Lemma 3.4.2. The smoothness proof for the more general case is similar in spirit. Once the smoothness of the rescaled error vector is proved, the magnitude of the error vector follows directly from Definition 3.2.2.

Lemma 3.4.2. *(FEBE prediction step): Consider an IDC method constructed using $M+1$ uniformly distributed nodes, and a FEBE integrator for the prediction step. Let $y(t)$, the solution to the IVP* (3.1), *have at least $\sigma \geq M+2$ degrees of smoothness, and let $\vec{\eta}^{[0]} = (\eta_0^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$, be the numerical solution computed after the prediction step. Then the rescaled error vector, $\vec{\tilde{e}}^{[0]} = \frac{1}{h}\vec{e}^{[0]}$, has M degrees of*

smoothness in the discrete sense.

Proof. We drop the superscript [0] as there is no ambiguity. Since

$$\eta_{m+1} = \eta_m + hf_N(t_m, \eta_m) + hf_S(t_{m+1}, \eta_{m+1}), \tag{3.15}$$

and

$$\begin{aligned} y_{m+1} &= y_m + \int_{t_m}^{t_{m+1}} f_N(\tau, y(\tau))d\tau + \int_{t_m}^{t_{m+1}} f_S(\tau, y(\tau))d\tau, \\ &= y_m + hf_N(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!}\frac{d^i f_N}{dt^i}(t_m, y_m) \\ &\quad + hf_S(t_{m+1}, y_{m+1}) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!}\frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1}) + \mathcal{O}(h^\sigma). \end{aligned} \tag{3.16}$$

Subtracting equation (3.15) from equation (3.16) gives

$$\begin{aligned} e_{m+1} &= e_m + h(f_N(t_m, y_m) - f_N(t_m, \eta_m)) \\ &\quad + h(f_S(t_{m+1}, y_{m+1}) - f_S(t_{m+1}, \eta_{m+1})) \\ &\quad + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!}\frac{d^i f_N}{dt^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!}\frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1}) + \mathcal{O}(h^\sigma), \end{aligned}$$

where $e_{m+1} = y_{m+1} - \eta_{m+1}$ is the error at t_{m+1}. Let

$$u_m = (f_N(t_m, y_m) - f_N(t_m, \eta_m)) + (f_S(t_{m+1}, y_{m+1}) - f_S(t_{m+1}, \eta_{m+1})),$$

and

$$r_m = \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!}(\frac{d^i f_N}{dt^i}(t_m, y_m) + \frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1})).$$

To prove the smoothness of the rescaled error vector, $\tilde{e} = e/h$, we will use an inductive approach with respect to s, the degree of smoothness. First, note that a discrete differentiation of the rescaled error vector gives,

$$(d_1\tilde{e})_m = \frac{\tilde{e}_{m+1} - \tilde{e}_m}{h} = \frac{u_m}{h} + \frac{r_m}{h^2} + \mathcal{O}(h^{\sigma-2}), \tag{3.17}$$

We are now ready to prove that $\vec{\tilde{e}}$ has M degrees of smoothness by induction. Since $\|\vec{\tilde{e}}\|_\infty \sim \mathcal{O}(h)$, thus, $\vec{\tilde{e}}$ has at least zero degrees of smoothness in the discrete sense. Assume that $\vec{\tilde{e}}$ has $s \leq M-1$ degrees of smoothness. We will show that $\overrightarrow{d_1\tilde{e}}$ has s degrees of smoothness, from which we can conclude that $\vec{\tilde{e}}$ has $(s+1)$ degrees of smoothness.

First,

$$\begin{aligned}
u_m &= \sum_{i=1}^{\sigma-2} \frac{1}{i!} e_m^i \frac{\partial^i f_N}{\partial y^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{1}{i!} e_{m+1}^i \frac{\partial^i f_S}{\partial y^i}(t_{m+1}, y_{m+1}) \\
&\quad + \mathcal{O}((e_m)^{\sigma-1}) + \mathcal{O}((e_{m+1})^{\sigma-1}) \\
&= \sum_{i=1}^{\sigma-2} \frac{1}{i!} \tilde{e}_m^i h^i \frac{\partial^i f_N}{\partial y^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{1}{i!} \tilde{e}_{m+1}^i h^i \frac{\partial^i f_S}{\partial y^i}(t_{m+1}, y_{m+1}) \\
&\quad + \mathcal{O}((h\tilde{e}_m)^{\sigma-1}) + \mathcal{O}((h\tilde{e}_{m+1})^{\sigma-1})
\end{aligned}$$

where we have performed a Taylor expansion of $f_N(t, \eta_m)$ about $y = y_m$ and of $f_S(t, \eta_{m+1})$ about $y = y_{m+1}$. Denote f_{y^i} to represent either $\frac{\partial^i f_N}{\partial y^i}$ or $\frac{\partial^i f_S}{\partial y^i}$. Since f_{y^i} has $(\sigma - i - 1)$ degrees of smoothness in the continuous sense, $\overrightarrow{f_{y^i}} = [f_{y^i}(t_0, y_0), \ldots, f_{y^i}(t_M, y_M)]$ has $(\sigma - i - 1)$ degrees of smoothness in the discrete sense. Consequently, $h^{i-1}\overrightarrow{f_{y^i}}$ has $(\sigma - 2)$ degrees of smoothness, which implies that $\frac{u_m}{h}$ has $\min(\sigma - 2, s)$ degrees of smoothness. Also $\frac{r_m}{h^2}$ has at least s degrees of smoothness from the smoothness property of the IVP (3.1). Hence $\overrightarrow{d_1\tilde{e}}$ has s degrees

of smoothness $\implies \vec{\tilde{e}}$ has $(s+1)$ degrees of smoothness. Since this argument holds for $\sigma \geq M+2$, we can conclude that $\vec{\tilde{e}}$ has M degrees of smoothness. $\square$

Now we present the lemmas required for the proof of Theorem 3.4.1.

Lemma 3.4.3. *(Prediction step): Let $y(t)$, the solution to the IVP (3.1), have at least σ (and f_S and f_N have at least $\sigma - 1$) degrees of smoothness in the continuous sense, where $\sigma \geq M+2$. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (3.7). Let $\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ be the numerical solution on the quadrature points (3.7), obtained using an r_0th order ARK method (as described in Theorem 3.4.1) at the prediction step. Then:*

1. *The error vector $e^{[0]} = y - \eta^{[0]}$ satisfies $\|e^{[0]}\|_\infty \sim \mathcal{O}(h^{r_0+1})$.*
2. *The rescaled error vector $\tilde{e}^{[0]} = \frac{1}{h^{r_0}} e^{[0]}$ has $\min(\sigma - r_0, M)$ degrees of smoothness in the discrete sense.*

Lemma 3.4.4. *(Correction step): Let $y(t)$, the solution to the IVP (3.1), have at least σ (and f_S and f_N have at least $\sigma - 1$) degrees of smoothness in the continuous sense, where $\sigma \geq M+2$. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (3.7). Suppose $e^{[k-1]} \sim \mathcal{O}(h^{s_{k-1}+1})$ and the rescaled error vector $\tilde{e}^{[k-1]} = \frac{1}{h^{s_{k-1}}} e^{[k-1]}$ has $M + 1 - s_{k-1}$ degrees of smoothness in the discrete sense. Then, after the kth correction step, provided an r_kth order ARK method (as described in Theorem 3.4.1) is used and $k \leq K_{loop}$,*

1. *The error vector $e^{[k]}$ satisfies $\|e^{[k]}\|_\infty \sim \mathcal{O}(h^{s_k+1})$.*
2. *The rescaled error vector $\tilde{e}^{[k]} = \frac{1}{h^{s_k}} e^{[k]}$ has $M+1-s_k$ degrees of smoothness in the discrete sense.*

Note that most results and the proofs (e.g. smoothness) are similar to the analysis in Section 2.3.2 (implicit IDC-RK) and in [18] (explicit IDC-RK). The main difference between the proof of the truncation error for IDC-RK (explicit or implicit) and IDC-ARK methods lies in a portion of the proof of Lemma 3.4.4, stated below as Proposition 3.4.5, following some notational details. The proposition asserts that the rescaled error vectors (rescaled by $h^s k-1$), obtained from formally applying ARK methods to a rescaled error equation, are equivalent to the error resulting from the numerical formulation that we implement. Thus, since an r_kth order ARK method applied to the *rescaled* error equations achieves r_kth order, the *unscaled* numerical implementation is $\mathcal{O}(h^s k-1^{+r} k)$, provided the exact solution $y(t)$ and the functions f_N, f_S are sufficiently smooth.

For purposes of clarity, before stating the proposition, we first introduce some notational details related to the rescaling, and recall the error equation (3.11). The analysis for the correction steps in this section will rely on this form of the error equation.

Assume that after the $(k-1)$th correction, the method is $\mathcal{O}(h^s k-1)$. We rescale the error $e^{(k-1)}$ (3.8), and $Q^{(k-1)}$ and G_α from the error equation (3.11), by $h^s k-1$. Then the rescaled equations (3.18) and (3.19) are $\mathcal{O}(1)$ (see Section 2.3.2, or [18]).

Define the rescaled quantities as

$$\widetilde{e}^{(k-1)}(t) = \frac{1}{h^s k-1} e^{(k-1)}(t), \tag{3.18a}$$

$$\widetilde{Q}^{(k-1)}(t) = \frac{1}{h^s k-1} Q^{(k-1)}(t), \tag{3.18b}$$

$$\widetilde{G}_\alpha^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)) = \frac{1}{h^s k-1} G_\alpha(t, h^s k-1 \widetilde{Q}^{(k-1)}(t)), \quad \alpha = N, S. \tag{3.18c}$$

Then the rescaled error equation is given by

$$
\begin{aligned}
(\widetilde{Q}^{(k-1)})'(t) &= \widetilde{G}_N^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)) + \widetilde{G}_S^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)), \\
\widetilde{Q}^{(k-1)}(t) &= 0.
\end{aligned} \tag{3.19}
$$

A p-stage, r^{th} order ARK method (3.5) applied to (3.19) gives, for $i = 1, 2, \ldots, p$ and $\alpha = N, S$:

$$
\begin{aligned}
\widetilde{k}_i^{\alpha} &= \widetilde{G}_{\alpha}^{(k-1)}(t_0 + c_i h, \widetilde{Q}_0^{[k-1]} + h(\sum_{j=1}^{i-1} a_{ij}^N \widetilde{k}_j^N + \sum_{j=1}^{p} a_{ij}^S \widetilde{k}_j^S)), \\
\widetilde{\Omega}_1^{[k]} &= \widetilde{Q}_0^{[k-1]} + h \sum_{i=1}^{p} (b_i^N \widetilde{k}_i^N + b_i^S \widetilde{k}_i^S),
\end{aligned} \tag{3.20}
$$

where the Greek letter $\widetilde{\Omega}$ denotes the rescaled solution in the numerical space, and $\widetilde{Q}_0^{[k-1]} = \widetilde{Q}^{(k-1)}(t_0)$. In the actual implementation, we discretize (3.11) as in equations (3.14) with F_N, F_S. We use (3.14) with G_N, G_S instead of F_N, F_S for the analysis.

The following proposition is where the essential difference between the proof of the truncation error for IDC-RK [18] and IDC-ARK methods lies.

Proposition 3.4.5. *The Taylor series for the rescaled exact error,* $\widetilde{e}^{(k-1)}(t_0+h) = \frac{1}{h^s k-1} e^{(k-1)}(t_0 + h)$, *and for the rescaled numerical error,* $\frac{1}{h^s k-1}\delta_1^{[k]}$ *in* (3.14), *coincide up to and including the term* h^r *for a sufficiently smooth error function* $\widetilde{e}^{(k-1)}(t)$, *and the exact solution* $y(t)$ *has* σ *and* f_N, f_S *have* $\sigma - 1$ *degrees of smoothness.*

Proof. It suffices to prove the following **claim**:

$$
h^s k-1 \widetilde{k}_i^{\alpha} = k_i^{\alpha} + \mathcal{O}(h^{\sigma-1}), \quad \forall\, i = 1, 2, \ldots, p, \quad \alpha = N, S, \tag{3.21}
$$

where k_i^α and $\widetilde{k}_i^\alpha$ are as in equations (3.14) and (3.20). Set

$$\begin{aligned} A_i &\doteq h\Big(\sum_{j=1}^{i-1} a_{ij}^N h^s k-1 \widetilde{k}_j^N + \sum_{j=1}^{p} a_{ij}^S h^s k-1 \widetilde{k}_j^S\Big) \\ &= h \sum_{j=1}^{p} (a_{ij}^N h^s k-1 \widetilde{k}_j^N + a_{ij}^S h^s k-1 \widetilde{k}_j^S), \end{aligned} \tag{3.22}$$

$$B_i \doteq h\Big(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p} a_{ij}^S k_j^S\Big) = h \sum_{j=1}^{p} (a_{ij}^N k_j^N + a_{ij}^S k_j^S), \tag{3.23}$$

$$C_{lj}^\alpha \doteq \frac{1}{l!}\frac{\partial^l}{\partial y^l} G\alpha(t_0 + c_j h, Q_0^{(k-1)}) \sum_{v=1}^{l} A_j^{l-v} B_j^{v-1},$$

where $\alpha = N, S$, and the equalities in (3.22) and (3.23) hold since $a_{ij}^N = 0$ for $j \geq i$. To prove (3.21), we prove that $h^s k-1 \widetilde{k}_i^\alpha - k_i^\alpha = \mathcal{O}(h^n)$ for $n = 1, 2, \ldots, \sigma - 1$, for $i = 1, 2, \ldots, p$, and $\alpha = N, S$ using induction with respect to n; i.e., we show that

$$\begin{aligned} h^s k-1 \widetilde{k}_i^\alpha - k_i^\alpha = h^n \sum_{l,n} \sum_{j,n} (\beta_{lj,n}^{\alpha N} (h^s k-1 \widetilde{k}_{jn}^N - k_{jn}^N) + \beta_{lj,n}^{\alpha S} (h^s k-1 \widetilde{k}_{jn}^S - k_{jn}^S)) \\ + \mathcal{O}(h^{\sigma-1}), \end{aligned} \tag{3.24}$$

where the notation is as follows.

$$\sum_{l,n} = \sum_{l_1}^{\sigma-2} \sum_{l_2}^{\sigma-2} \cdots \sum_{l_n}^{\sigma-2}, \qquad \sum_{j,n} = \sum_{j_1}^{p} \sum_{j_2}^{p} \cdots \sum_{j_n}^{p},$$

$$\beta_{lj,n}^{\alpha N} = \sum_{1}^{2^{n-1}} \prod_{1}^{n} Ca, \qquad \beta_{lj,n}^{\alpha S} = \sum_{1}^{2^{n-1}} \prod_{1}^{n} Ca,$$

where for the β coefficients, C represents some C_{lj}^N or C_{lj}^S, a represents some a_{ij}^N or a_{ij}^S, and ${}^\alpha N$ and ${}^\alpha S$ denote that the C, a coefficients that appear in the sums

depend on α and N or α and S, respectively. For example, if $n = 2$, then

$$\beta^{\alpha N}_{lj,2} = C^{\alpha}_{l_1 i} a^{N}_{ij_1} C^{N}_{l_2 j_1} a^{N}_{j_1 j_2} + C^{\alpha}_{l_1 i} a^{S}_{ij_1} C^{S}_{l_2 j_1} a^{N}_{j_1 j_2},$$
$$\beta^{\alpha S}_{lj,2} = C^{\alpha}_{l_1 i} a^{N}_{ij_1} C^{N}_{l_2 j_1} a^{S}_{j_1 j_2} + C^{\alpha}_{l_1 i} a^{S}_{ij_1} C^{S}_{l_2 j_1} a^{S}_{j_1 j_2},$$

which is clarified in the expansions below.

We prove (3.24), and hence (3.21), using induction with respect to n, the power of h in equation (3.24).

- $n = 1$: Using the rescaled quantities (3.18) and Taylor expanding $G_\alpha(t, y)$ about $Q_0^{(k-1)}$ in y, we obtain for $\alpha = N, S$:

$$\begin{aligned} & h^s k-1 \widetilde{k}_i^{\alpha} - k_i^{\alpha} \qquad (3.25) \\ & = h^s k-1 \widetilde{G}_{\alpha}^{(k-1)}(t_0 + c_i h, \widetilde{Q}^{(k-1)}(t_0) + h \sum_{j=1}^{p} (a_{ij}^{N} \widetilde{k}_j^{N} + a_{ij}^{S} \widetilde{k}_j^{S})) \\ & \quad - G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + h \sum_{j=1}^{p} (a_{ij}^{N} k_j^{N} + a_{ij}^{S} k_j^{S})) \\ & = G_\alpha(t_0 + c_i h, h^s k-1 (\widetilde{Q}^{(k-1)}(t_0) + h \sum_{j=1}^{p} (a_{ij}^{N} \widetilde{k}_j^{N} + a_{ij}^{S} \widetilde{k}_j^{S}))) \\ & \quad - G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + B_i) \\ & = G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + A_i) - G_\alpha(t_0 + c_i h, Q_0^{(k-1)} + B_i) \\ & = \sum_{l_1=1}^{\sigma-2} \frac{1}{l_1!} \frac{\partial^{l_1}}{\partial y^{l_1}} G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0))((A_i)^{l_1} - (B_i)^{l_1}) \\ & \quad + \mathcal{O}(h^{\sigma-1}). \qquad (3.26) \end{aligned}$$

Factoring $(A_i)^{l1} - (B_i)^{l1} = (A_i - B_i)\sum_{v=1}^{l1} A_i^{l1-v} B_i^{v-1}$, we obtain

$$
\begin{aligned}
& h^s k-1 \widetilde{k}_i^\alpha - k_i^\alpha \\
&= \sum_{l_1=1}^{\sigma-2} C_{l_1 i}^\alpha (A_i - B_i) + \mathcal{O}(h^{\sigma-1}) \\
&= h \sum_{l_1=1}^{\sigma-2} C_{l_1 i}^\alpha \sum_{j_1=1}^{p} (a_{ij_1}^N (h^s k-1 \widetilde{k}_{j_1}^N - k_{j_1}^N) + a_{ij_1}^S (h^s k-1 \widetilde{k}_{j_1}^S - k_{j_1}^S)) \\
&\quad + \mathcal{O}(h^{\sigma-1}) \\
&= h \sum_{l_1=1}^{\sigma-2} \sum_{j_1=1}^{p} (C_{l_1 i}^\alpha a_{ij_1}^N (h^s k-1 \widetilde{k}_{j_1}^N - k_{j_1}^N) + C_{l_1 i}^\alpha a_{ij_1}^S (h^s k-1 \widetilde{k}_{j_1}^S - k_{j_1}^S)) \\
&\quad + \mathcal{O}(h^{\sigma-1}) \\
&= h^1 \sum_{l,1} \sum_{j,1} (\beta_{lj,1}^{\alpha N} (h^s k-1 \widetilde{k}_{j_1}^N - k_{j_1}^N) + \beta_{lj,1}^{\alpha S} (h^s k-1 \widetilde{k}_{j_1}^S - k_{j_1}^S)) \\
&\quad + \mathcal{O}(h^{\sigma-1}), \quad \forall i = 1, 2, \ldots, p.
\end{aligned} \tag{3.27}
$$

Note here that $\beta_{lj,1}^{\alpha N} = C_{l_1 i}^\alpha a_{ij_1}^N$, and $\beta_{lj,1}^{\alpha S} = C_{l_1 i}^\alpha a_{ij_1}^S$.

- $n+1$: Assume (3.24) holds for $n < \sigma - 1$ (for both $\alpha = N$ and S). Following the same process of Taylor expanding as in equations (3.26) and (3.27) in the

$n = 1$ case , we see that for $\alpha = N, S$:

$$
\begin{aligned}
& h^s k-1 \widetilde{k}_i^\alpha - k_i^\alpha \\
& = h^n \sum_{l,n} \sum_{j,n} \left(\beta_{lj,n}^{\alpha N} (h^s k-1 \widetilde{k}_{jn}^N - k_{jn}^N) + \beta_{lj,n}^{\alpha S} (h^s k-1 \widetilde{k}_{jn}^S - k_{jn}^S) \right) \\
& \quad + \mathcal{O}(h^{\sigma-1}) \\
& \overset{(3.27)}{=} h^n \sum_{l,n} \sum_{j,n} (\beta_{lj,n}^{\alpha N} (h \sum_{l_{n+1}=1}^{\sigma-2} \sum_{j_{n+1}=1}^{p} \\
& \quad (C_{l_{n+1} j_n}^N a_{j_n j_{n+1}}^N (h^s k-1 \widetilde{k}_{j_{n+1}}^N - k_{j_{n+1}}^N) \\
& \quad + C_{l_{n+1} j_n}^N a_{j_n j_{n+1}}^S (h^s k-1 \widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1})) \\
& \quad + \beta_{lj,n}^{\alpha S} (h \sum_{l_{n+1}=1}^{\sigma-2} \sum_{j_{n+1}=1}^{p} (C_{l_{n+1} j_n}^S a_{j_n j_{n+1}}^N (h^s k-1 \widetilde{k}_{j_1}^N - k_{j_1}^N) \\
& \quad + C_{l_{n+1} j_n}^S a_{j_n j_{n+1}}^S (h^s k-1 \widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1})) \\
& \quad + \mathcal{O}(h^{\sigma-1}) \\
& = h^{n+1} \sum_{l,n+1} \sum_{j,n+1} \\
& \quad ((\beta_{lj,n}^{\alpha N} C_{l_{n+1} j_n}^N a_{j_n j_{n+1}}^N + \beta_{lj,n}^{\alpha S} C_{l_{n+1} j_n}^S a_{j_n j_{n+1}}^N)(h^s k-1 \widetilde{k}_{j_1}^N - k_{j_1}^N) \\
& \quad + (\beta_{lj,n}^{\alpha N} C_{l_{n+1} j_n}^N a_{j_n j_{n+1}}^S + \beta_{lj,n}^{\alpha S} C_{l_{n+1} j_n}^S a_{j_n j_{n+1}}^S) \\
& \quad (h^s k-1 \widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1}) \\
& = h^{n+1} \sum_{l,n+1} \sum_{j,n+1} (\beta_{lj,n+1}^{\alpha N} (h^s k-1 \widetilde{k}_{j_{n+1}}^N - k_{j_{n+1}}^N) \\
& \quad + \beta_{lj,n+1}^{\alpha S} (h^s k-1 \widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1}), \quad \forall i = 1, 2, \ldots, p,
\end{aligned}
$$

where

$$\beta^{\alpha N}_{lj,n+1} = \beta^{\alpha N}_{lj,n} C^{N}_{l_{n+1}j_n} a^{N}_{j_n j_{n+1}} + \beta^{\alpha S}_{lj,n} C^{S}_{l_{n+1}j_n} a^{N}_{j_n j_{n+1}},$$
$$\beta^{\alpha S}_{lj,n+1} = \beta^{\alpha N}_{lj,n} C^{N}_{l_{n+1}j_n} a^{S}_{j_n j_{n+1}} + \beta^{\alpha S}_{lj,n} C^{S}_{l_{n+1}j_n} a^{S}_{j_n j_{n+1}}.$$

It follows that (3.21) holds. The rest of the proof of Proposition 3.4.5 is similar to the implicit IDC-RK analysis in Section 2.3.2. □

Now we state the implicit IDC-RK results again as a corollary to the semi-implicit theorem.

Corollary 3.4.6. *Let $y(t)$ be the solution to the IVP*

$$y'(t) = f(t, y), \quad t \in [0, T],$$
$$y(0) = y_0,$$

and $y(t)$ has at least σ ($\sigma \geq M+2$) and f has at least $\sigma - 1$ degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with $t \in [0, H]$ and uniformly distributed quadrature points (3.7). *Suppose an r_0th order implicit RK method is used in the prediction step and $(r_1, r_2, \ldots, r_{K_{loop}})$th order implicit RK methods are used in K_{loop} correction steps. Let $s_k = \sum_{j=0}^{k} r_j$. If $s_{K_{loop}} \leq M + 1$, then the local truncation error is of order $\mathcal{O}(h^{s_{K_{loop}}+1})$.*

Proof. The result follows from Theorem 3.4.1, if we set $f_N = 0$. □

3.5 Stability

The stability of an explicit (or an implicit) scheme can be measured by the conventionally accepted definition of a stability region [37], as presented in Section 2.2.3,

Definition 2.2.2.

For a semi-implicit numerical method, stability is less easily defined. In the literature, various measures for stability have been proposed for semi-implicit [60, 9] or additive Runge–Kutta methods [62, 56, 11]. In fact, all of these definitions can be applied to any semi-implicit scheme. We focus on two of these definitions. Minion suggests splitting the Dahlquist-type equation in such a way that the stiff part is represented as the real part of the eigenvalue and the nonstiff part is represented by the imaginary part of the eigenvalue, which is useful for seeing whether the numerical method works for a convection-diffusion problem, but may not be reasonable when considering a problem with stiff oscillations. Liu and Zou define an $A(\alpha)$-stable semi-implicit method (similar to [62], except in [62], they also present a boundary locus-type method), which could cover a wider class of problems [56].

3.5.1 Stability for Convection-Diffusion Type Problem

First consider the definition from [60].

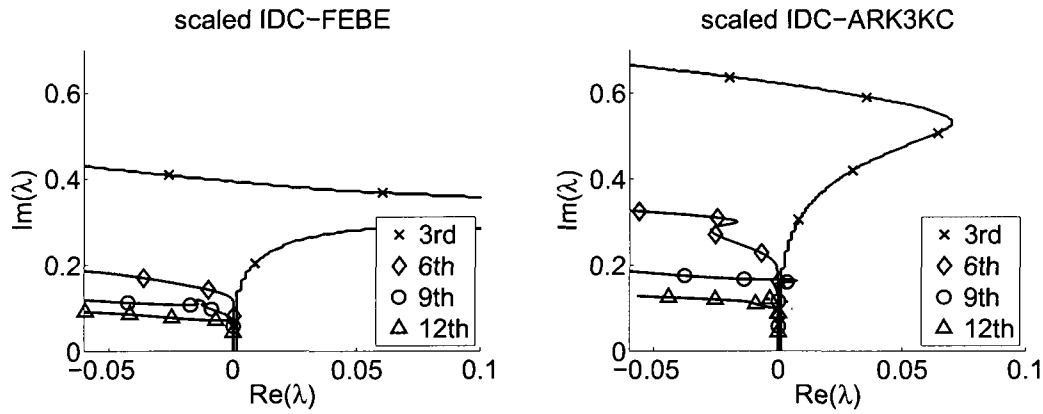
Definition 3.5.1. *The amplification factor for a semi–implicit numerical method, $Am(\lambda)$, with $\lambda = \alpha + i\beta$, is interpreted as the numerical solution to*

$$y'(t) = \alpha y(t) + i\beta y(t), \quad y(0) = 1, \tag{3.28}$$

where the explicit component of the numerical method is applied to the imaginary part $i\beta y(t)$, and the implicit component of the numerical method is applied to the real part $\alpha y(t)$, after a time step of size 1 for $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$, i.e., $Am(\lambda) = y(1)$.

Definition 3.5.2. *The stability region, S, for a semi–implicit numerical method, is the subset of the complex plane $\mathbb{C}$, consisting of all λ such that $Am(\lambda) \leq 1$ for ODE 3.28, i.e., $S = \{\lambda : Am(\lambda) \leq 1\}$.*

In Figure 3.1a, the stability regions for 3rd, 6th, 9th, and 12th order IDC-FEBE (IDC constructed using forward and backward Euler) with 2, 5, 8, and 11 correction loops, respectively, are computed numerically and plotted. Figure 3.1b shows 3rd, 6th, 9th, and 12th order IDC-ARK3KC (IDC constructed with 3rd order ARK3KC) with 0, 1, 2, and 3 correction loops, respectively. The shaded regions represent the stable regions. Note that for the same order method, IDC-ARK3KC shows a marked improvement in stability over IDC-FEBE. This correlates with the results in [18], where it was shown that IDC-RK methods were found to possess better stability properties than SDC constructed with forward Euler integrators [18]. Such a stability measure provides insight on the stability property of a



(a) IDC-FEBE (b) IDC-ARK3KC

Figure 3.1: Stability regions for 3rd, 6th, 9th, and 12th order IDC constructed using (a) forward and backward Euler with 2, 5, 8, and 11 correction loops (and 3, 6, 9, 12 uniform quadrature nodes), respectively, and (b) 3rd order ARK3KC with 0, 1, 2, and 3 correction loops (and 3, 6, 9, and 12 uniform quadrature nodes), respectively. The crosses denote the 3rd order stability regions, diamonds for 6th order, circles for 9th order, and triangles for 12th order. The stability regions are scaled by the number of implicit function evaluations $((\#stages)(M+1)M$, where FEBE has 1 stage, ARK3KC has 4 stages).

semi-implicit method (such as an IDC-ARK method), especially when it is applied to a convection-diffusion problem (e.g., the problem in Section 3.6.2), where the

convection term is treated explicitly and diffusion term implicitly.

3.5.2 Stability for Other Types of Problems

We remark that the stability property of the semi-implicit scheme greatly depends on the splitting of equation (3.1), therefore, the stability analysis must be adapted for specific problems. Hence we also present a definition from [56] that may apply to a broader class of problems, although our results are inconclusive.

Rather than the Dahlquist equation (2.45), we consider a test problem of the form

$$\begin{aligned} y'(t) &= \lambda_S y + \lambda_N y, \\ y(0) &= 1. \end{aligned} \tag{3.29}$$

λ_S and λ_N depict the eigenvalues of the Jacobian of f_S and f_N, respectively, where $Re(\lambda_S) \leq 0$, $Re(\lambda_N) \leq 0$, and $|Re(\lambda_S)| \gg |Re(\lambda_N)|$. The *semi-implicit amplification factor*, denoted $Am(\lambda_S, \lambda_N)$, is determined by applying the semi-implicit numerical method (e.g., (3.4)) to the test problem (3.29) for one time step, obtaining

$$y_1 = Am(\lambda_S, \lambda_N) \tag{3.30}$$

Definition 3.5.3. *[56] An $A(\alpha)$-stability region for a semi-implicit numerical method is*

$$S_\alpha^{f_S \cap f_N} = \{\lambda_N \in \mathbb{C} \quad s.t. \quad |Am(\lambda_S, \lambda_N)| \leq 1 \quad \forall \lambda_S \in S_\alpha^{f_S}\}, \tag{3.31}$$

where

$$S_{\alpha}^{fS} = \{\lambda_S \in \mathbb{C} \; s.t. \; |arg(-\lambda_S)| \leq \alpha, \; \lambda_S \neq 0\} \cup \{0\}. \tag{3.32}$$

An L-stable semi-implicit method is an A-stable (i.e., $A(\frac{\pi}{2})$-stable) which also satisfies

$$\lim_{\lambda_S \to 0} |Am(\lambda_S, \lambda_N)| = 0. \tag{3.33}$$

Stability regions were numerically and analytically determined for some ARK methods to confirm the (A-stable type) stability regions presented in [56]. Definition 3.5.3 with $\alpha = \pi/2$ was used to calculate the regions. ARK2A1, ARK3A3, and ARK4A2 were chosen due to the obvious difference in size and/or shape among the methods and between their explicit stability regions (i.e., the stability region arising when the explicit part of the ARK method is calculated alone as an RK method) and their semi-implicit stability regions. The general algorithm for computing the semi-implicit stability regions, using (3.29) and (3.30), follows.

1. Grid the left half complex plane for λ_S.

2. For each fixed λ_S, find the stability region (according to the usual definition, Definition 2.2.2) in terms of λ_N.

3. Take the intersection of the stability regions from step 2 to obtain the ARK stability region.

Remark 3.5.4. *In step* 1, *it is necessary to refine the grid for* λ_S *close to the imaginary axis. These values of* λ_S *appear to be the troublesome ones that shrink the ARK stability region to smaller than the explicit stability region. It may be unnecessary to include* λ_S *with large negative real parts. These observations are supported by [55, 62] and by the structure of a typical amplification factor's equation.*

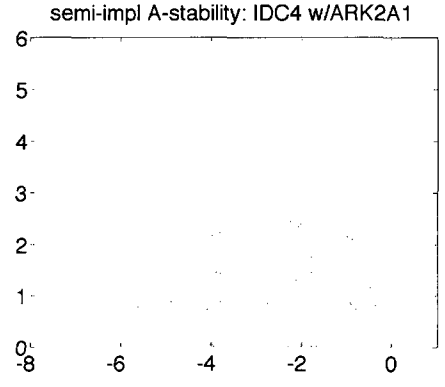
E.g., in

$$Am(\lambda_S, \lambda_N) = \frac{(1 - \lambda_S - \frac{1}{2}\lambda_S^2) + (1 - \lambda_S)\lambda_N + \frac{1}{2}\lambda_N^2}{1 - 2\lambda_S + \lambda_S^2}, \tag{3.34}$$

the equation for the amplification factor of ARK2A1, we see that large negative real values of λ_S make $Am(\lambda_S, \lambda_N)$ approximately constant in modulus, while values of λ_S closer to the imaginary axis may increase the modulus on their own or through interaction with λ_N. Furthermore, [62] presents a particular case where it is rigorously shown that the maximum of the amplification factor occurs on the imaginary axis.

At first glance, ARK methods appear to reduce the stability region; however, when one considers that this restriction only applies to the nonstiff eigenvalue, while the stiff eigenvalue may be anywhere in the left-half complex plane, a smaller stability region may not be unsatisfactory [56]. Nevertheless, we are still limited by the lack of I-stability (imaginary axis stability, [36]) for the nonstiff component.

The stability regions for IDC-ARK schemes were ascertained numerically as for ARK schemes. A fourth order IDC-ARK method using ARK2A1 has a semi-

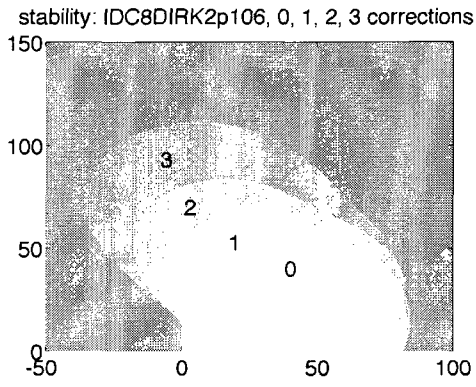


(a) IDC-ARK2A1

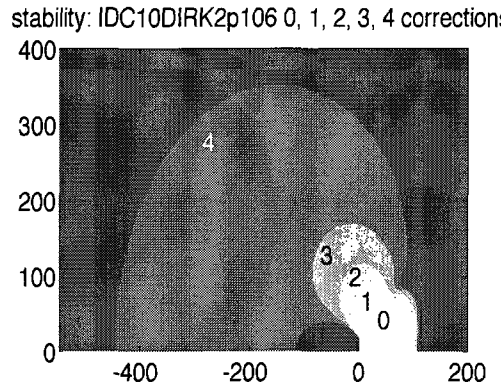
Figure 3.2: Stability region for fourth order IDC with ARK2A1 in prediction and correction loops. λ_N is plotted, consistent with Definition 3.5.3, and $\alpha = \pi/2$.

implicit stability region (Figure 3.2a) that is much larger than the semi-implicit stability regions for both fourth order ARK methods shown in [56]. Unfortunately, the 8th order IDC-ARK (with ARK2A1) method's stability region is not A-stable at all (in the semi-implicit sense of Definition 3.5.3); however, the numerical solution for (3.35) and (3.36) computed via the same 8th order IDC-ARK method seems to be stable, perhaps even for $\epsilon = 0.01$.

To investigate the loss of (semi-implicit) A-stability, we first fixed λ_S, and plotted some explicit stability regions only. We achieved very reasonable stability regions that increased with the order of the IDC method, as expected from the results in [18]. Then we fixed λ_N, and plotted some implicit stability regions only. We also plotted some stability regions for implicit IDC-RK methods constructed with a second order DIRK method from [2] (Figures 3.3a and 3.3b). Here we discovered the problem. Clearly the stability regions decrease in size as the number of correction loops increases and frequently the higher order methods lose A-stability. In particular, we notice that in Figure 3.3b, the 10th order method loses A-stability; even $A(\alpha)$-stability is lost. Hence, although the explicit region increases, as seen in [18], the implicit region loses $A(\alpha)$-stability as number of IDC corrections increases, and the semi-implicit method struggles between improving and worsening its stability. Thus far the results of measuring semi-implicit stability via Definition 3.5.3 do not look clear, although the main difficulty appears to arise in the implicit part of the method, where increasing the order of an implicit method decreases its stability region. This decrease is well-known for BDF methods, but generally is unpredictable for implicit RK methods. Also, [56]'s method given in Definition 3.5.3 appears to be overly restrictive since some numerical solutions still appeared to be stable despite the numerical method's loss of semi-implicit stability. Therefore, we chose not to pursue this means of measuring semi-implicit stability further.



(a) IDC-DIRK2, 8 nodes



(b) IDC-DIRK2, 10 nodes

Figure 3.3: Stability regions for IDC with 2nd order DIRK (from [2]) in prediction and correction loops. The labels are the number of correction loops, and are placed outside of the stability regions. From $0, 1, \ldots, 4$ correction loops, the IDC methods have corresponding order of accuracy $2, 4, \ldots, 10$. Stability regions are standard, according to Definition 2.2.2.

3.6 Numerical Results

In this section we describe some numerical experiments with various IVPs and an advection diffusion equation. Numerical experiments support Theorem 3.4.1, which states that the semi-implicit IDC-ARK methods have similar order of accuracy rules as for IDC with explicit RK methods [18]. Additionally, in most cases, higher order IDC-ARK presents an efficiency advantage over existing IMEX ARK schemes. Several different ARK methods of various orders were tested within the IDC prediction and correction loops: second order ARK2A1 [56] and ARK2ARS [3], third order ARK3KC [45] and ARK3BHR1 (Appendix 1. in [8]), and fourth order ARK4A2 [56] and ARK4KC [45]. For efficiency, these IDC-ARK methods were compared with their constituent ARK methods. This section first lists three ODE test problems, followed by the combined results for order of accuracy, comments on order reduction, and efficiency results. Then an advection-diffusion equation is described, followed by its order of accuracy results and comments on an improved CFL condition.

We further comment that the focus of our study is not the choice of f_N, f_S, and thus we make naive choices of f_N and f_S in each example below, acknowledging that better choices are likely to exist (in particular, there is much study on the choice of f_N, f_S for PDEs, such as in [43, 44, 34]). Note also, that, in most of the literature, the examples of semi-implicit methods used to solve problems with stiff regions only test convergence and efficiency up to the time just *before* the stiff layer, at which point they claim that adaptive steps should be taken [45, 8, 3]. In this work, we have tested through times that extend *beyond* the stiff layer, and hence our results may appear to be worse than they should. Testing our methods in the standard way is likely to produce much better results than shown here.

3.6.1 ODE Test Problems

Test Problems

The following test problems, an initial layer problem and Van der Pol's oscillator, were used in numerical determination of order of accuracy and efficiency. Each IVP was first computed in a completely nonstiff situation ($\epsilon = 1$). Then varying levels of stiffness, which caused each IVP to have a stiff and nonstiff component, were considered.

1. **Initial layer problem**

   Pareschi and Russo's example [45], with nonequilibrium, or perturbed, initial conditions $y_1(0) = \pi/2$ and $y_2(0) = 1/2$ is tested:

$$y_1'(t) = -y_2(t), \quad t \in [0, 4], \tag{3.35a}$$

$$y_2'(t) = y_1(t) + \frac{1}{\epsilon}(\sin(y_1(t)) - y_2(t)), \tag{3.35b}$$

where we choose

$$f_N = \begin{pmatrix} -y_2 \\ y_1 \end{pmatrix}, \quad f_S = \begin{pmatrix} 0 \\ \frac{1}{\epsilon}(\sin y_1 - y_2) \end{pmatrix}.$$

2. **Van der Pol's equation**

A standard nonlinear oscillatory test problem, Van der Pol's equation [60], is examined:

$$y_1'(t) = y_2(t), \quad t \in [0, 4], \tag{3.36a}$$

$$y_2'(t) = \frac{1}{\epsilon}(-y_1(t) + (1 - y_1(t)^2)y_2(t)), \tag{3.36b}$$

with initial conditions $y_1(0) = 2$ and $y_2(0) = 2/3$, and we choose

$$f_N = \begin{pmatrix} y_2 \\ 0 \end{pmatrix}, \quad f_S = \begin{pmatrix} 0 \\ \frac{1}{\epsilon}(-y_1 + (1 - y_1^2)y_2) \end{pmatrix}.$$

Order of Accuracy

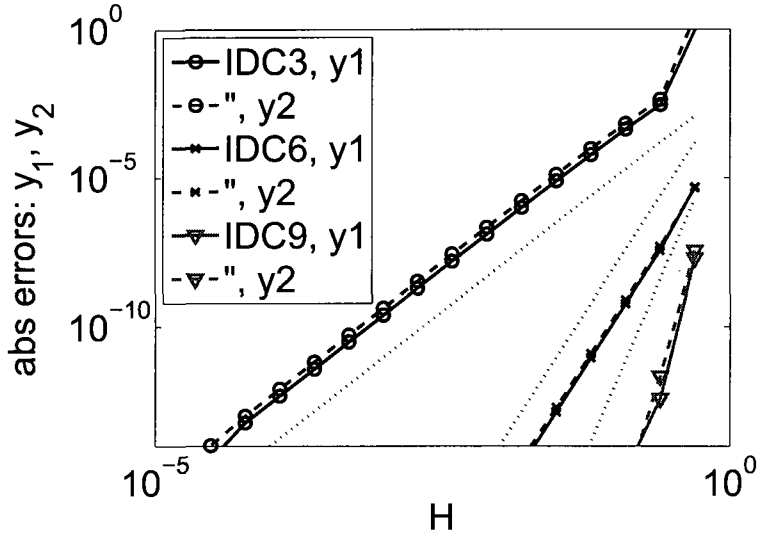
In the following convergence study, error is plotted versus the stepsize H, where the error at the final time is calculated by comparing successive solutions. For the IVPs, the error is the absolute error at the final time.

In agreement with Theorem 3.4.1, the order of accuracy of the IDC method increases by the order of the ARK method used for each prediction and correction loop, as seen for the initial layer problem in Figure 3.4a and for Van der Pol's oscillator in Figures 3.4b and 3.5a. Here we choose $\epsilon = 1$, pertaining to the nonstiff scenario. The circles in Figures 3.4a and 3.4b correspond to the error of the 3rd order ARK3KC method (equivalent to one prediction loop of IDC). It has slope ≈ 3, as

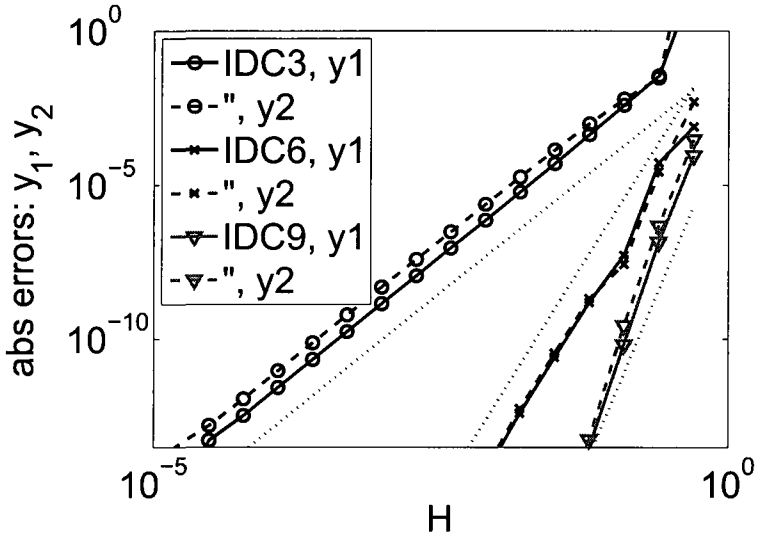
expected, since a third order ARK method is used and there are $M+1=3$ nodes in each subinterval $[t_n, t_{n+1}]$. After one correction loop computed using a third order ARK method (and $M+1=6$), the crosses show sixth order convergence — an increase in the order by three. Similarly, a second correction increases the order to 9, as seen with the triangles. Thus three loops (1 prediction + 2 corrections) of a 3rd order ARK method produces a 9th order IDC method, when there are $M+1=9$ nodes in each subinterval $[t_n, t_{n+1}]$. Similarly, in Figure 3.5a, we see a fourth order increase with each loop of the IDC method when the fourth order ARK4KC method is used (giving an eighth order method after two loops). We also solved these test problems with various orders of ARK methods (2, 3, 4) and IDC-ARK (4, 6, 8, 9) constructed using second order ARK2A1 [56] and ARK2ARS [3], third order ARK3KC [45] and ARK3BHR1 (Appendix 1. in [8]), and fourth order ARK4A2 [56] and ARK4KC [45] (not all shown). In each implementation, we anticipate that *# loops * ARK order = expected IDC order*. In general, the expected order of accuracy is seen for nonstiff problems.

Order Reduction

For stiff problems, a phenomenon known as order reduction is observed. The effects of increasing the stiffness of the problems (i.e. decreasing ϵ) on the order of convergence can be seen in Figures 3.6a and 3.6b. The plots show the error for computing the initial layer problem, IVP 3.35 and Van der Pol's oscillator, (3.36) in a stiff case ($\epsilon = 10^{-3}$) using third order ARK3KC in 0, 1, and 2 correction loops of IDC. Order reduction that is ϵ-dependent is apparent for both the initial layer problem, IVP (3.35), and the oscillator, IVP (3.36). Once the stepsize is below ϵ, however, it appears that the expected IDC order is achieved. For the initial layer IVP (3.35), the order reduction clearly still allows for convergence of the method within the order reduction regime (Figure 3.6a). Van der Pol's oscillator exhibits

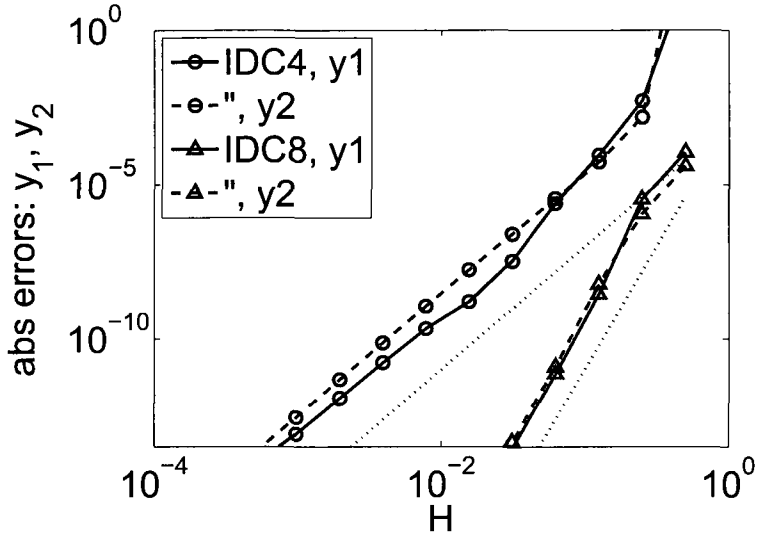


(a) IVP 3.35, $\epsilon = 1$ (nonstiff)

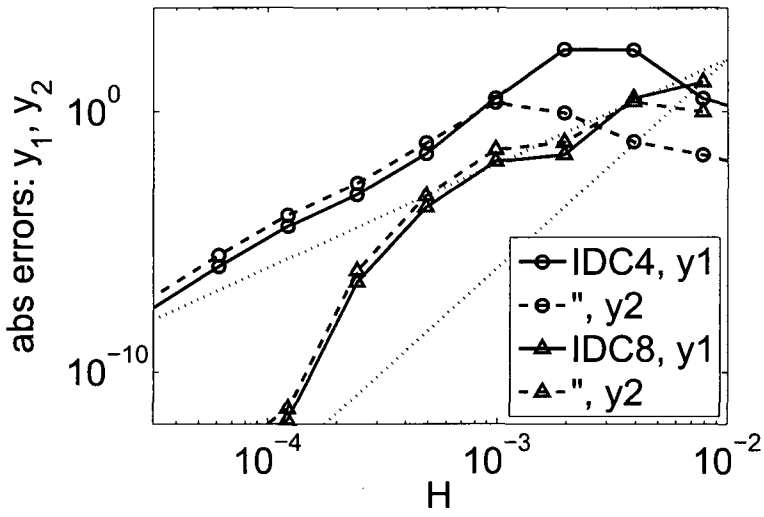


(b) VdP (3.36), $\epsilon = 1$ (nonstiff)

Figure 3.4: Convergence study of absolute error at $T = 4$ vs H, for 3rd, 6th, and 9th order IDC constructed using 3rd order ARK3KC with 0, 1, and 2 correction loops (and 3, 6, and 9 quadrature nodes), respectively. The order of accuracy is clearly seen, as the dotted reference lines (with slopes of 3, 6, 9) indicate.

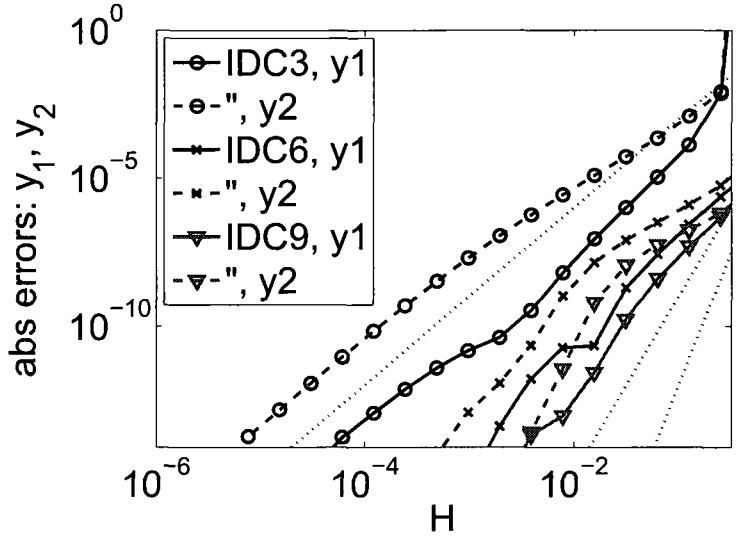


(a) VdP (3.36), $\epsilon = 1$ (nonstiff)

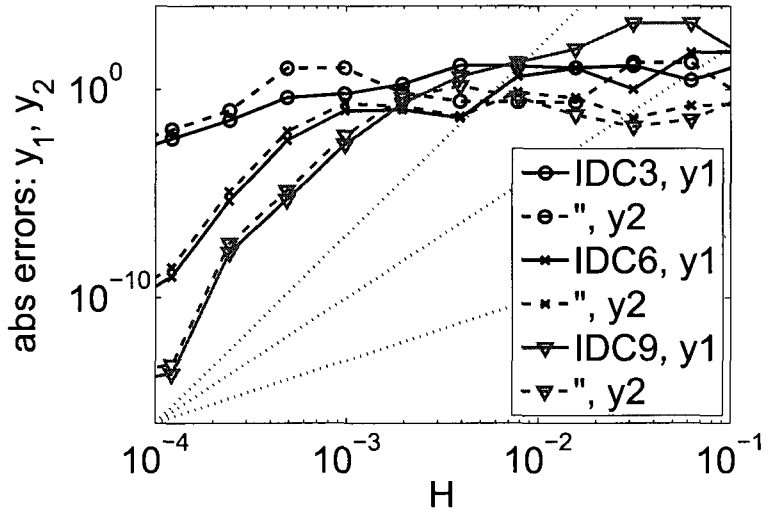


(b) VdP (3.36), $\epsilon = 10^{-3}$ (mildly stiff)

Figure 3.5: Convergence study of absolute error at $T = 4$ vs H, for 4th and 8th order IDC constructed using 4th order ARK4KC with 0 and 1 correction loops (and 4 and 8 quadrature nodes), respectively. The order of accuracy is clearly seen in (a), as the dotted reference lines (with slopes of 4, 8) indicate.



(a) IVP 3.35, $\epsilon = 10^{-3}$ (mildly stiff)



(b) VdP (3.36), $\epsilon = 10^{-3}$ (mildly stiff)

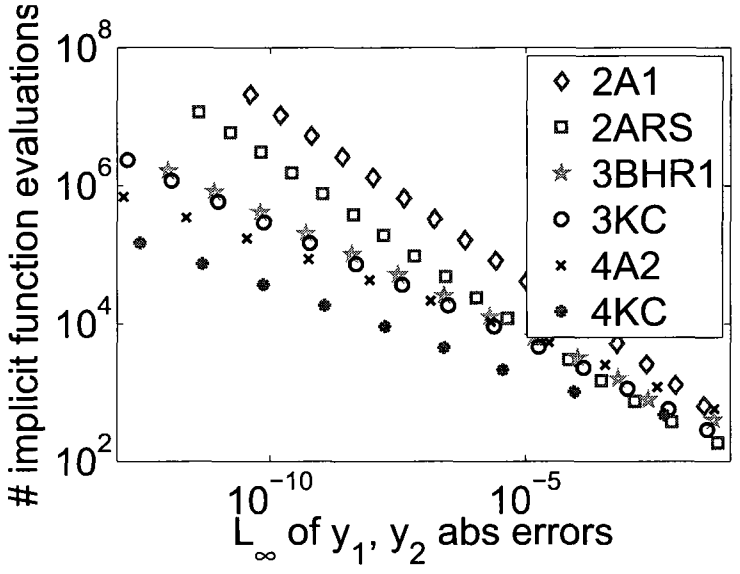
Figure 3.6: Convergence study of absolute error at $T = 4$ vs H, for 3rd, 6th, and 9th order IDC constructed using 3rd order ARK3KC with 0, 1, and 2 correction loops (and 3, 6, and 9 quadrature nodes), respectively. The dotted reference lines (with slopes of 3, 6, 9) indicate the expected order of the methods, but note order reduction.

instability for larger timesteps; however, once the stepsize is small enough, order reduction is apparent, and then as the stepsize is further decreased, it appears that the expected IDC order is achieved (Figures 3.6b and 3.5b). The better behavior of solutions to the IVP (3.35) is not surprising, since many of the ARK methods were designed to solve problems similar to (3.35) (see, e.g. [56, 45, 21]), but the oscillator problem (3.36) has recurring stiff peaks, which are difficult to handle even with most popular integrators.

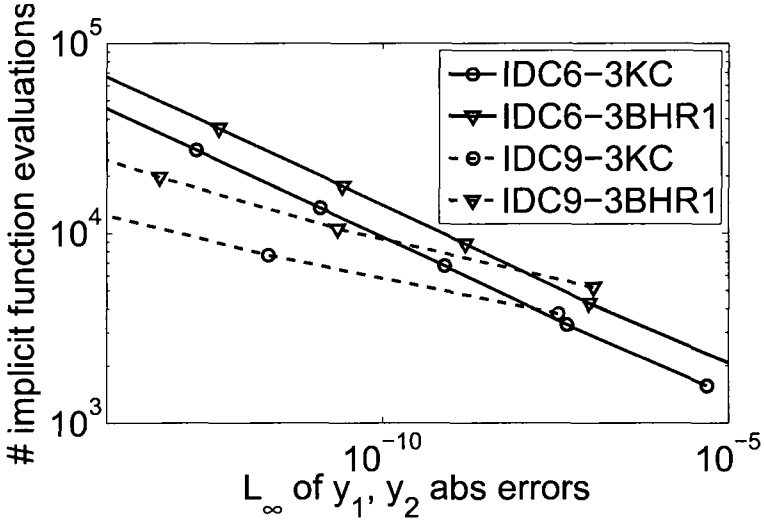
Efficiency

To measure efficiency, we plot the number of implicit function evaluations versus the error. As in the convergence plots, the error is found by comparing successive solutions (absolute error at the final time). The number of implicit function evaluations is counted numerically by tracking the number of times the "stiff" function and the Jacobian are called, including the Newton iterations. In general, the higher order IDC-ARK methods are more efficient than popular ARK methods. We see two general "rules of thumb." First, if the order of the method is higher, then fewer function evaluations are required to reach a low error tolerance. This trend can be observed for ARK methods (equivalent to a prediction loop) (Figure 3.7a) and when more correction loops are taken (Figure 3.8a).

Second, the efficiency of an IDC-ARK method generally depends upon its constituent ARK method; i.e., if ARK method a is more efficient than ARK method b, then IDC constructed using ARK a is more efficient than IDC constructed using ARK b. For example, an IDC method using the 3rd order ARK3KC frequently is more efficient than the other methods we tested (see, e.g., Figure 3.7b), but IDC using ARK2A1 performs relatively inefficiently (as expected since ARK2A1 alone requires the highest number of function evaluations among the ARK methods we tested in Figure 3.7a). We note that the behavior for the nonstiff ($\epsilon = 1$, Figure 3.8a)

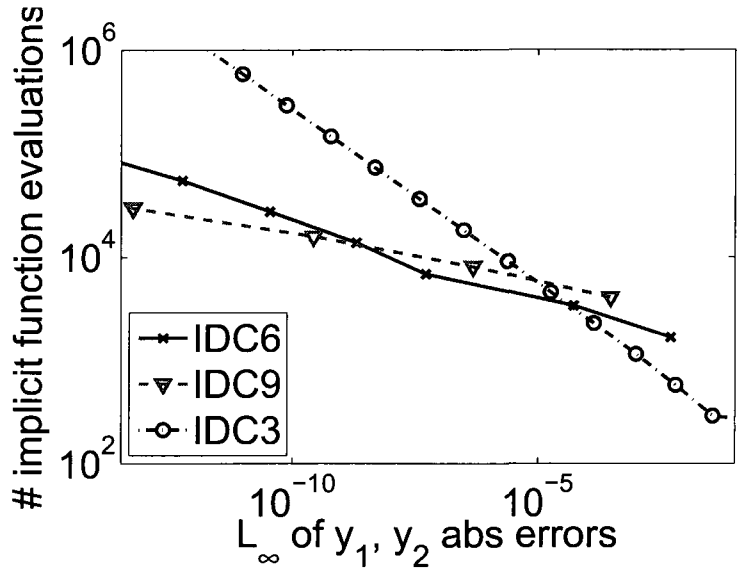


(a) VdP (3.36), $\epsilon = 1$ (nonstiff)

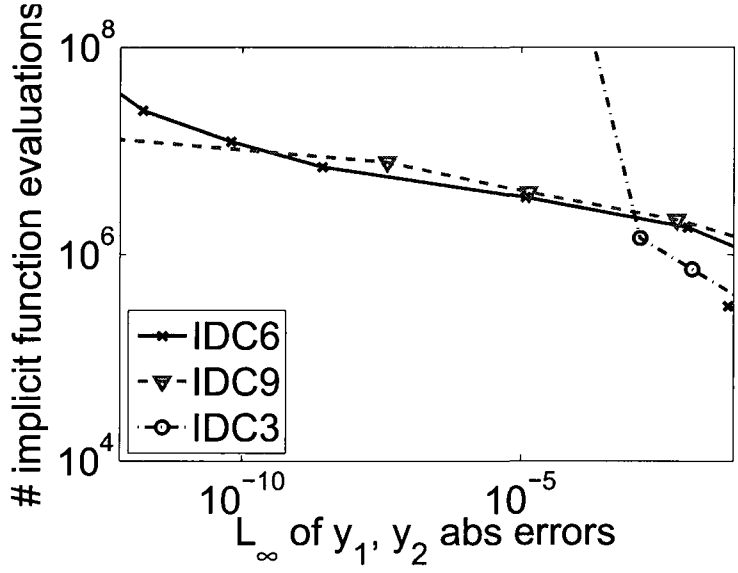


(b) IVP 3.35, $\epsilon = 1$ (nonstiff)

Figure 3.7: Efficiency studies of # implicit function evaluations vs absolute error at $T = 4$, using (a) ARK methods of second (2A1 [56], 2ARS [3]), third (3BHR1 [8], 3KC [45]), and fourth (4A2 [56], 4KC [45]) orders (no IDC). (b) for 6th and 9th order IDC constructed using 3rd order ARK3KC and 3rd order ARK3BHR1 with 1 and 2 correction loops (and 6 and 9 quadrature nodes), respectively.



(a) VdP (3.36), $\epsilon = 1$ (nonstiff)



(b) VdP (3.36), $\epsilon = 10^{-3}$ (mildly stiff)

Figure 3.8: Efficiency study of # implicit function evaluations vs absolute error at $T = 4$, for 3rd, 6th, and 9th order IDC constructed using 3rd order ARK3KC with 0, 1, and 2 correction loops (and 3, 6, and 9 quadrature nodes), respectively).

and mildly stiff ($\epsilon = 10^{-3}$, Figure 3.8b) cases is similar.

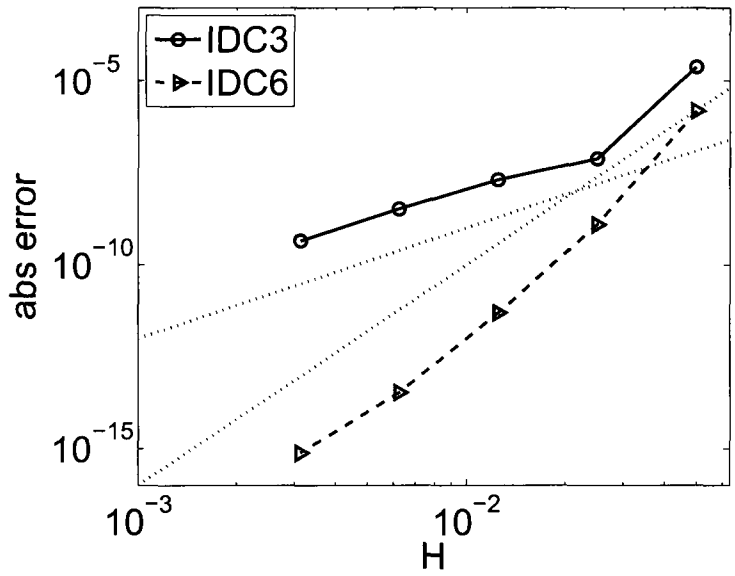
3.6.2 Advection-Diffusion Example

We now consider an advection-diffusion equation

$$u_t = -u_x + u_{xx}, \quad t \in [0, 0.1], \quad x \in [0, 1], \tag{3.37a}$$

$$u(x, 0) = 2 + \sin(4\pi x), \tag{3.37b}$$

with periodic boundary conditions. We solve the advection-diffusion BVP (3.37) via method of lines with fast Fourier transform (FFT) for the spatial derivatives and time-stepping with semi-implicit IDC-ARK, where the advection term is treated explicitly ($f_N = -u_x$) and the diffusion term is treated implicitly ($f_S = u_{xx}$). This choice of methods ensures that the error is dominated by the time-stepping. We expect that treating the diffusion term implicitly will remove the requirement that the CFL condition must satisfy $\Delta t \leq c\Delta x^2$, and that the CFL condition will be controlled only by the advection term so that $\Delta t \leq c\Delta x$. To verify this hypothesis, we plot the error for IVP 3.37 at $T = 0.1$ versus $H = \Delta t = 0.5\Delta x$, timestepping with IDC constructed with 3rd order ARK3KC, in Figure 3.9a. The error is the L_1 (spatial) norm of the absolute errors at the final time and was computed by comparing the numerical solution to a reference solution computed with small Δx and Δt. Shown are IDC methods with zero and one correction loops (3rd and 6th order convergence, respectively). Since the expected convergence orders are seen when Δt is proportional to Δx, we conclude that it is reasonable to say that treating the diffusion term implicitly improves the CFL condition to $\Delta t \leq c\Delta x$. This improvement is exactly what we expect for a semi-implicit method.



(a) Advection-diffusion BVP (3.37)

Figure 3.9: CFL study showing absolute error at $T = 0.1$ vs $H = \Delta t = 0.5\Delta x$. FFT is used for the spatial discretization, and time-stepping is done with 3rd and 6th order IDC constructed with 3rd order ARK3KC with 0 and 1 correction loops of IDC, respectively, where the advection term is treated explicitly and the diffusion term is treated implicitly. This choice of methods ensures that the error is dominated by the time-stepping. Since the expected order of accuracy in time is seen, as the dotted reference lines (with slopes of 3, 6) indicate, it appears that the CFL condition is $\Delta t \leq c\,\Delta x$.

3.7 Concluding Remarks

We have provided a general framework for the simple construction of arbitrary order IMEX methods. These semi-implicit IDC-ARK methods use ARK integrators as base schemes inside the IDC framework and can thus be constructed to higher order easily, with no need to consider the complicated order conditions that are typically required for the construction of ARK methods. High order IDC-ARK methods can be used to solve multiscale differential equations that involve disparate time scales more efficiently than popular ARK schemes when a low error tolerance is desired. We have performed an analysis of the local truncation error of IDC-ARK methods, shown improved stability over IDC-FEBE methods, and conducted numerical results showing order of convergence, efficiency, and the potential for an improved CFL condition. We plan to examine the use of asymptotic preserving ARK schemes (as in [64], and Section 6.1) in the prediction and correction loops of IDC as a potential way to mitigate the issue of order reduction that arises with increased stiffness. We also have begun an investigation of embedded IDC methods, including IDC-ARK, and their implementation in an adaptive setting (see Chapter 4 for preliminary results). We anticipate that adaptivity will provide additional efficiency gains over IDC-ARK, and in fact, adaptive timestepping is the natural way to handle many multiscale problems.

Chapter 4

Adaptive IDC Methods

4.1 Introduction

In this chapter, we consider initial value problems of the form

$$\begin{aligned} y' &= f_S(t,y) + f_N(t,y), \\ y(0) &= y_0, \quad t \in [0,T], \end{aligned} \tag{4.1}$$

where $y \in \mathbb{R}^n$, and the function containing stiff terms, $f_S(t,y) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, and the function containing non-stiff terms, $f_N(t,y) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$, are Lipschitz continuous in the second arguments. When $f_S = 0$, i.e., equation (4.1) contains only non-stiff terms, explicit integrators such as popular explicit Runge–Kutta (RK) methods or Adams–Bashforth methods can be applied. When $f_N = 0$, i.e., equation (4.1) contains only stiff terms, implicit RK methods or Adams–Moulton methods can be applied [2]. When equation (4.1) contains both stiff and non-stiff terms, implicit–explicit (IMEX) methods [4] or additive Runge–Kutta (ARK) methods can be applied [21].

Although in previous chapters we only consider fixed step sizes, to guarantee

accuracy in the solution of equation (4.1), estimation of the error and adaptive step size control is often desirable. For example, in Van der Pol's oscillator (3.36), there are stretches of time where the solution changes only mildly, punctuated by oscillatory stiff peaks in the derivative. The stiff oscillations cause methods implemented with a fixed step size to exhibit order reduction, as explained in section 3.6.1. Adaptive time-stepping may be needed for other types of problems as well. For problems containing only non-stiff terms, a widely accepted approach is to use a matched pair of explicit RK methods [37], also known in the literature as embedded RK methods. The premise is to select a pair of RK methods (i) whose Taylor series approximations agree with the corresponding Taylor terms of the solution up to orders p and $p+1$, and (ii) whose stage weights, a_{ij}, and stage nodes, c_j, overlap as much as possible for efficiency. The matched pairs are often expressed compactly in a Butcher Tableau,

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

where A is an $s \times s$ lower triangular matrix, $b, \hat{b}$ and c are s-vectors. An example of a pair of matched explicit RK method is the Runge–Kutta–Fehlberg Method (RKF45), which generates a fourth and fifth order method using six stages. This approach of matched pairs can be extended to implicit RK and ARK methods. In [33], matched diagonally implicit RK pairs are discussed, and [45] discusses embedded ARK methods and error control.

In recent papers [18, 17, 15], the present authors have illustrated that integral deferred correction (IDC) methods are competitive with popular RK and ARK methods. In brief, IDC methods are a class of deferred (or defect) correction methods

that generate a sequence of approximations that successively reduce a residual. First introduced by Dutt, Greengard and Rokhlin in [25], high order IDC methods can be constructed systematically without solving tedious order conditions. This work exploits the framework of an IDC method, using computed approximations of the numerical error for step size control. In Section 4.2, IDC methods are reviewed, and numerical comparisons between IDC and existing embedded methods are given in Section 4.3. Conclusions are given in Section 4.5.

4.2 Integral Deferred Correction

IDC methods are essentially predictor-corrector schemes. In this section, we first discuss how the error for an approximate solution is computed. Then, the IDC framework and algorithm is given. Finally, an adaptive time step control is discussed.

4.2.1 Error Equation

Given an approximate solution, $\eta(t)$, to the exact solution, $y(t)$, of (4.1), the error of the approximate solution is

$$e(t) = y(t) - \eta(t). \tag{4.2}$$

Defining the residual, $\epsilon(t)$, as

$$\epsilon(t) = \eta'(t) - f_S(t, \eta(t)) - f_N(t, \eta(t)), \tag{4.3}$$

then the derivative of the error given in equation (4.2) satisfies

$$\begin{aligned} e'(t) &= y'(t) - \eta'(t) \\ &= f_S\left(t, y(t)\right) - f_S\left(t, \eta(t)\right) + f_N\left(t, y(t)\right) - f_N\left(t, \eta(t)\right) - \epsilon(t). \end{aligned}$$

The integral form of the error equation can then be obtained,

$$\begin{aligned} \left[e(t) + \int_0^t \epsilon(\tau)\, d\tau\right]' =& f_S\left(t, \eta(t) + e(t)\right) - f_S\left(t, \eta(t)\right) \\ &+ f_N\left(t, \eta(t) + e(t)\right) - f_N\left(t, \eta(t)\right). \end{aligned}$$

An iteration scheme is implemented to generate successively more accurate approximations,

$$\begin{aligned} \left[e^{(k-1)}(t) + \int_0^t \epsilon^{(k-1)}(\tau)\, d\tau\right]' =& f_S\left(t, \eta^{(k)}(t)\right) - f_S\left(t, \eta^{(k-1)}(t)\right) \\ &+ f_N\left(t, \eta^{(k)}(t)\right) - f_N\left(t, \eta^{(k-1)}(t)\right), \end{aligned} \quad (4.4)$$

where $\eta^{(k)}(t) = \eta^{(k-1)}(t) + e^{(k-1)}(t)$. It was shown in [18], under mild assumptions, that if a pth order single step method is used to approximate equation (4.4), and $\eta^{(k-1)}$ has q orders of accuracy, then $\eta^{(k)}$ has $(q+p)$ orders of accuracy.

4.2.2 Implementation of an IDC Method

The time domain $[0, T]$ is first discretized into N intervals,

$$t_n = n\Delta t, \quad n = 0, \ldots, N \quad \text{where} \quad \Delta t = \frac{T}{N}.$$

Then, each interval, $I_n = [t_n, t_{n+1}]$, is further discretized into M sub-intervals,

$$t_{n,m} = t_n + m\,\delta t, \quad m = 0, \ldots, M \quad \text{where} \quad \delta t = \frac{\Delta t}{M}.$$

As described in [25], discretizing each interval uniformly into M subintervals allows for an $(M+1)^{st}$ order IDC method. The IDC algorithm, described in Algorithm 7, is iterated completely in each time interval $[t_n, t_{n+1}]$ to define the starting value of the next interval, $[t_{n+1}, t_{n+2}]$. Equation (4.4) is rewritten as

$$\begin{aligned} Q^{(k-1)\prime}(t) = f_S &\left(Q^{(k-1)}(t) + y(0) + \int^t (f_N(\tau, \eta^{(k-1)}) + f_S(\tau, \eta^{(k-1)}))d\tau \right) \\ &- f_S(t, \eta^{(k-1)}) \\ &+ f_N \left(Q^{(k-1)}(t) + y(0) + \int^t (f_N(\tau, \eta^{(k-1)}) + f_S(\tau, \eta^{(k-1)}))d\tau \right) \\ &- f_N(t, \eta^{(k-1)}), \end{aligned} \tag{4.5}$$

where $Q^{(k-1)}(t) = e^{(k-1)}(t) + \int^t \epsilon^{(k-1)}(\tau)d\tau$, and the integral $\int^t (f_N(\tau, \eta^{(k-1)}) + f_S(\tau, \eta^{(k-1)}))d\tau$ is approximated at the gridpoints $t_{n,m}$ by an integration matrix (2.31), to accomodate numerical implementation.

4.2.3 Adaptive Time Step Control

In Algorithm 5, we wish to adjust the size of the time step based upon the size of the error between the numerical and the exact solutions. This adjustment should be done such that the time step is decreased if the error is too large and increased if the error is significantly smaller than necessary, since we wish to take as few time steps as possible. The procedure is as follows. First calculate the solution after one time step. Then calculate an approximation to the error. The stepsize Δt and the error *err* are input for Algorithm 5, which determines whether the error is acceptable and

Prediction
Solve (4.1) via ARK/RK.
Corrections
for $k = 1, \ldots, K_{loop}$
- Solve (4.5) for $Q^{[k-1]}$, the approximation to $Q^{(k-1)}$ at gridpoints $t_{n,m}$, via ARK/RK.
- Update numerical solution:

$$\begin{aligned}\eta^{[k]} &= \eta^{[k-1]} + Q^{[k-1]} - \int \epsilon^{[k-1]}(\tau)d\tau \\ &= Q^{[k-1]} + y(0) + \int (f_S(\tau, \eta^{[k-1]}) + f_N(\tau, \eta^{[k-1]}))d\tau.\end{aligned}$$

end

Algorithm 1: IDC method

whether the stepsize should be increased, decreased, or maintained. In this work, we do not consider the consequences of the choice of α, β, and γ, but they could be investigated more thoroughly using, e.g., ideas from [33].

Input: Δt: time step; *err*: error of new update; TOL: tolerance; $0 < \alpha < 1$: parameter for refining time step; $0 < \gamma < 1$: parameter for coarsening time step; $\beta > 1$: parameter for coarsening time step (in Figures 4.2a, 4.2b, 4.3a, and 4.1 below, we use $\alpha = 0.5$, $\gamma = 0.1$, $\beta = 2$)
Output: $\Delta\hat{t}$: new time step; *accept*: flag whether to accept new update;

if $err > TOL$ **then** `/* reject step */`
  $\Delta\hat{t} = \alpha\Delta t, \quad accept = 0;$
else if $err < \gamma * TOL$ **then** `/* coarsen step */`
  $\Delta\hat{t} = \beta\Delta t, \quad accept = 1;$
else $\Delta\hat{t} = \Delta t, \quad accept = 1$; `/* accept and leave unchanged */`

Algorithm 2: Determination of step size

For embedded RK (or embedded ARK) methods, two solutions of order p and $p+1$, resp. are calculated. Then, for example, the error can be approximated by comparing the difference between those two solutions. Each correction iteration of an IDC method includes an error estimator (the correction $\delta^{[k-1]}$), making it a natural choice for adaptive implementation. For IDC methods, the error approximation is

already built into the IDC method, since each correction loop solves (4.4) for the error. If we consider that IDC-RK (IDC-ARK) methods are RK (ARK) methods, then we notice that embedded IDC-RK (IDC-ARK) methods are a natural analogue to embedded RK (ARK) methods. In particular, if the final correction of an IDC method is order q, and the solution just before the final correction is order p, then the embedded IDC method gives two solutions of order p and $p+q$, resp. An obvious choice for the final correction is $q = 1$, to avoid extra computational expense when approximating the error. Note for IDC methods, Δt is adapted, while $\delta t = \Delta t/M$ remains uniform for *each* choice of Δt.

4.3 Numerical Comparisons

In this section, a numerical study is performed on such as Van der Pol's oscillator problem, a stiff ODE, comparing the performance of IDC-ARK methods and other popular embedded methods. In the figures and table in this section (with the exception of Figures 4.1a and 4.1b), we use the notation in [59] to compare various characteristics of the numerical methods and to facilitate future tests with additional IVPs in [59]. The notation is as follows:

- *solver* The numerical method with which the ODE was solved.
- *atol* Absolute error tolerance provided by the user.
- *scd* Minimum number of significant correct digits the numerical solution has at the end of the integration interval, i.e.

$$scd = -log_{10}||(\text{ relative error at the end of the integration interval })||_\infty \tag{4.6}$$

- *steps* Total number of steps the solver takes, including steps that are rejected because of error test failures and/or convergence test failures.

- *accept* Total number of accepted steps.

- *# imp f* Number of evaluations of the implicitly evaluated function, f_S (modified from [59] to count only implicit function evaluations). Implicit function evaluations are counted by keeping track of how many times the Matlab script that evaluates the implicit part (f_S) is called. We use only implicit function evaluations based on the assumption (as in [60]) that implicit calls will dominate due to Newton iterations, although this assumption is not always valid.

- *#Jac* Number of evaluations of the Jacobian of f_S. Note: The Jacobian is used in Newton iterations as part of the implicit method and is calculated analytically at every Newton iteration in every step, which is likely not the best way to handle the Jacobian. A smarter choice of Jacobian calculation should give far fewer evaluations. The tolerance for the Newton iterations is fixed at $= 10^{-13}$, and the maximum number of iterations is 10.

We also include the following additional characteristics with notation:

- *# refine* Number of times the timestep is refined, over the entire calculation.

- *# coarsen* Number of times the timestep is coarsened, over the entire calculation.

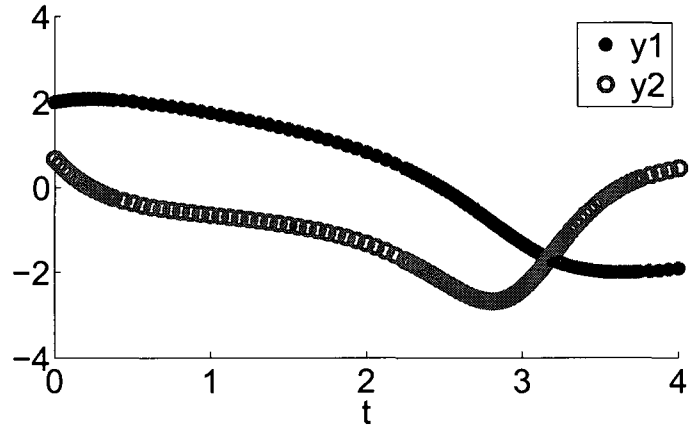
- *minstep* The smallest timestep taken over the integration interval.

- *maxstep* The largest timestep taken over the integration interval.

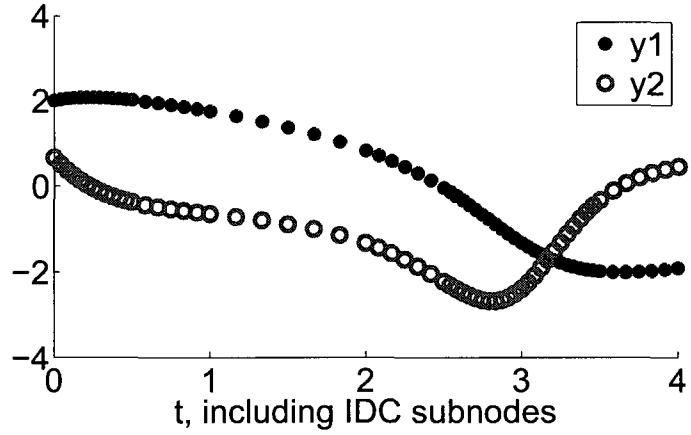
4.3.1 Van der Pol Oscillator

We consider the application of adaptive semi-implicit methods to Van der Pol's oscillator, as described by equation (3.36), with varying levels of stiffness (stiffness is increased by decreasing ϵ). As a preliminary test, we considered a recursive implementation of some adaptive IDC-ARK and ARK methods where the stepsize is only reduced when the error is too large and then reset to the original Δt for the next step. Algorithm 5 is far more practical, but we present the results from the recursive implementation as an example of the improvement that can be possible by using IDC methods. In comparison with embedded ARK methods, comparable or improved efficiency seems likely with appropriate choice of base scheme integrators. For example, I constructed adaptive IDC-ARK methods and found that fewer time gridpoints are required when using a 7(6) order embedded IDC-ARK method (Figure 4.1b) rather than a 4(3) order ARK method (from [45]) to solve Van der Pol's oscillator problem (Figure 4.1a) to the same error tolerance. The 7(6) IDC-ARK method solves the prediction and the first correction loops via a 3rd order ARK method from [45] and the final correction approximates the error via semi-implicit first order forward and backward Euler.

Now we examine the results from adapting the time step via Algorithm 5. In the figures, *tol* represents the quantity explained below as *atol*, and is the tolerance for the error when choosing whether or not to refine the timestep. The number of implicit function evaluations is described by $\#$ *imp* f below and is the number of times the Matlab script calls the stiff function f_S (which is evaluated via the implicit part of the numerical method). From Figures 4.2a, 4.2b, and 4.3a, we see that, for any choice of tolerance, ARK4(3) is more efficient than the same order IDC method constructed via a semi-implicit forward and backward Euler combination, IDC4(3)FEBE. This conclusion holds for the nonstiff ($\epsilon = 10^{-1}$), mildly stiff ($\epsilon =$

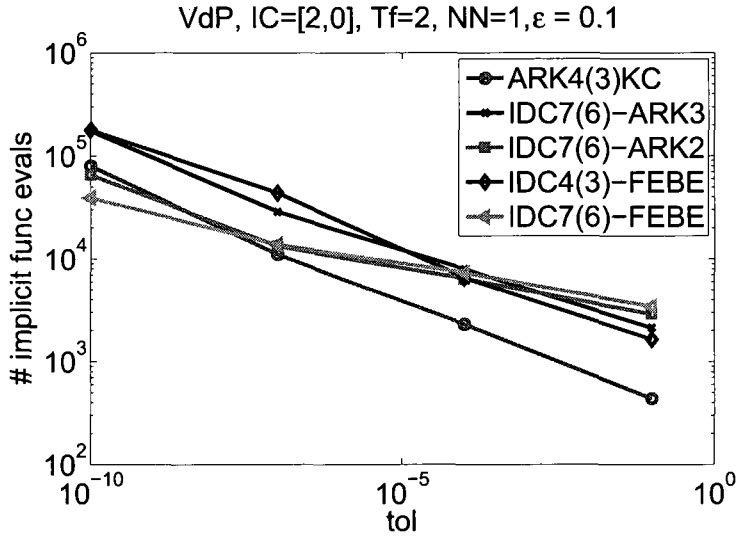


(a) ARK4(3)KC

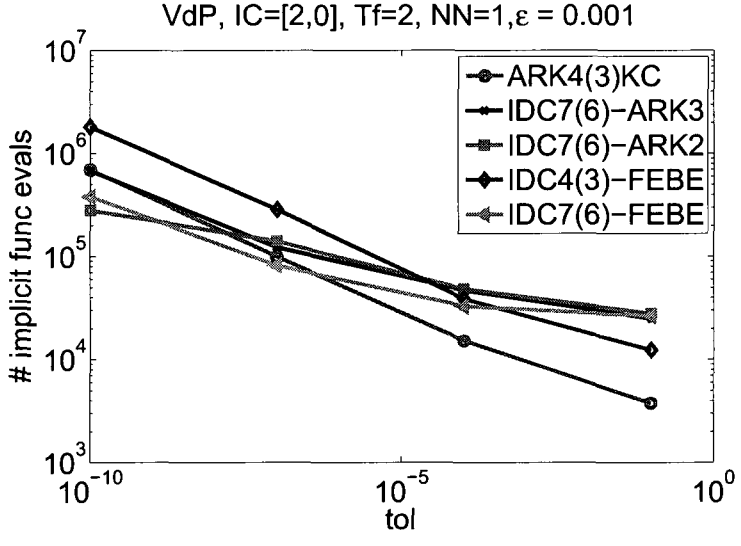


(b) IDC7(6)-ARK3KC/FBE

Figure 4.1: Preliminary results comparing (4.1a) an adaptive ARK method (embedded order 4(3) from [45]) and (4.1b) an adaptive IDC method constructed with a 3rd order ARK method from [45] (both recursively implemented). Both plots show solution to Van der Pol's oscillator ($\epsilon = 1$) for the same recursion error tolerance, 10^{-5}.

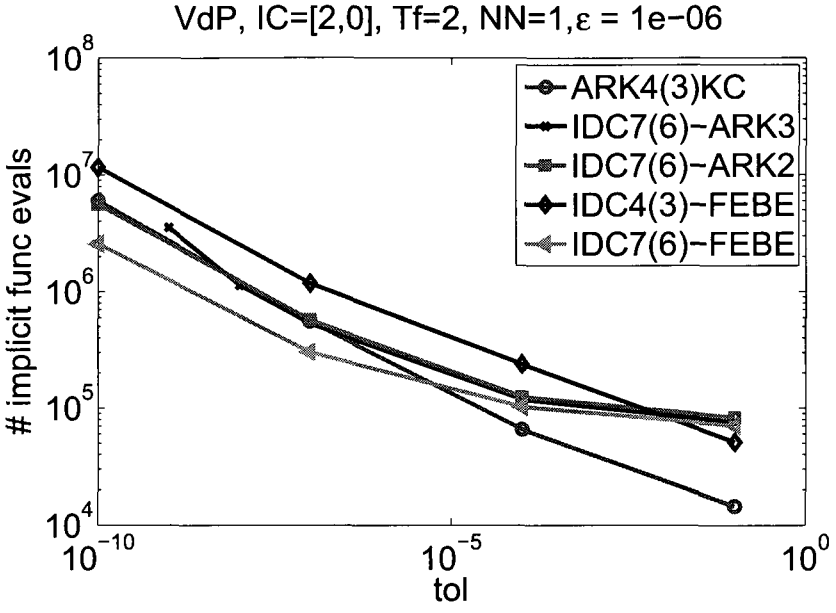


(a) $\epsilon = 0.1$



(b) $\epsilon = 0.001$

Figure 4.2: Efficiency study comparing semi-implicit ARK and IDC-ARK methods used to solve Van der Pol's oscillator. Plots show *# imp f* vs *atol*, as described in the text.



(a) $\epsilon = 10^{-6}$

Figure 4.3: Efficiency study comparing semi-implicit ARK and IDC-ARK methods used to solve Van der Pol's oscillator. Plots show *# imp f* vs *atol*, as described in the text.

solver	scd	atol	steps	accept	#imp f	#Jac	#refine	#coarsen	minstep	maxstep
IDC7(6)ARK3KC	4.22	0.1	223	120	46673	28833	103	56	2.11e-07	0.250
IDC7(6)ARK3KC	6.35	0.0001	371	232	74090	44410	139	60	1.19e-07	0.250
IDC7(6)ARK3KC	8.70	1e-07	1833	1508	338131	191491	325	49	2.98e-08	0.00781
IDC7(6)ARK2ARS	3.96	0.1	229	121	53343	29298	108	54	2.11e-07	0.250
IDC7(6)ARK2ARS	5.75	0.0001	369	233	81586	42841	136	63	1.19e-07	0.250
IDC7(6)ARK2ARS	8.62	1e-07	1886	1464	384666	186636	422	51	2.98e-08	0.00781
IDC7(6)ARK2ARS	11.77	1e-10	18801	16997	3767729	1793624	1804	229	4.66e-10	2.44e-04
IDC7(6)FEBE	3.62	0.1	203	112	43294	27460	91	50	4.23e-07	0.250
IDC7(6)FEBE	4.38	0.0001	315	183	63617	39047	132	60	1.06e-07	0.250
IDC7(6)FEBE	5.85	1e-07	1052	740	191672	109616	312	69	2.98e-08	0.250
IDC7(6)FEBE	9.45	1e-10	9872	8528	1668603	898587	1344	114	6.21e-10	3.73e-09
IDC4(3)FEBE	2.53	0.1	563	426	31381	19558	137	63	2.98e-08	0.250
IDC4(3)FEBE	4.09	0.0001	2963	2299	150631	88408	664	79	3.73e-09	0.250
IDC4(3)FEBE	6.74	1e-07	15703	13921	753783	424020	1782	84	9.31e-10	0.0625
IDC4(3)FEBE	10.84	1e-10	167779	129062	7583877	4060518	38717	6619	1.46e-11	0.00195
ARK4(3)KC	1.62	0.1	392	292	8621	5877	100	49	2.98e-08	0.500
ARK4(3)KC	6.31	0.0001	1779	1551	39114	26661	228	64	7.45e-09	0.0625
ARK4(3)KC	10.00	1e-07	15057	13360	331170	225771	1697	63	9.31e-10	0.00195
ARK4(3)KC	11.62	1e-10	163678	137267	3581616	2435870	26411	3594	5.82e-11	6.10e-05

Table 4.1: Test characteristics from [59] for Van der Pol's oscillator solved with various semi-implicit methods when $\epsilon = 10^{-6}$. Note here that all characteristics are for steps of size Δt, including for the IDC methods, since we accept or reject each Δt step, not each δt step.

10^{-3}), and stiff ($\epsilon = 10^{-6}$) cases of Van der Pol's oscillator. However, increasing the order of the IDC method (since a higher order ARK method is not easily constructed without IDC [45]) compares favorably to ARK4(3) in some cases. For larger error tolerances, ARK4(3) requires fewer function evaluations, but, in the nonstiff case, IDC7(6) methods constructed with FEBE or possibly with ARK2 require fewer implicit function evaluations when a small tolerance is desired (Figure 4.2a). In the mildly stiff case, IDC7(6) constructed with ARK2 or FEBE requires fewer function evaluations than ARK4(3) when a small tolerance is desired (Figure 4.2b). When $\epsilon = 10^{-6}$ (stiff case), only IDC7(6)FEBE requires fewer function evaluations than ARK4(3) to compute the solution to a small tolerance (Figure 4.3a). This result may be due to backward Euler's large stability region, hence improving the stability properties of IDC7(6). Note that IDC-FEBE methods are essentially identical to SISDC methods in [60], except we keep the IDC nodes uniform, whereas SISDC methods typically use some type of Gaussian nodes.

In the Table 4.1, we use the notation from [59], as described above, to compare semi-implicit ARK, IDC-FEBE, and IDC-ARK methods for solving Van der Pol's oscillator when $\epsilon = 10^{-6}$ with "exact" solution $y_1 = 0.1706167732170483 * 10$, $y_2 = -0.892809701024795$. Note that all characteristics are for steps of size Δt, including for the IDC methods, since we accept or reject each Δt step, not each δt step, and since the substeps of an IDC method can be considered stages of an RK method. Comparison of the characteristics in the table show that the 7(6) IDC-ARK and IDC-FEBE methods require far fewer steps than the 4(3) ARK method. Also, for smaller error tolerances, 7(6) IDC methods require fewer implicit function and Jacobian evaluations. Overall, the IDC methods of the same order as the ARK method show worse behavior. For large error tolerances, the ARK method is better, but for small error tolerances, the higher order IDC methods appear to be

comparable to the ARK method, even advantageous in some cases.

4.4 Possible Efficiency Simplifications

One idea for making the adaptive IDC more efficient is to make a more efficient estimation of the error, $e^{[k]}$. The following idea has not been developed further nor tested numerically. It is simply presented to suggest that a modified adaptive IDC method has potential to give better results than indicated in this work. Consider one step Δt, where M substeps of size δt are taken. From [41], we know that the high order accurate approximate solution, $\eta^{K_{loop}}$, can be written as a series expansion

$$\eta^{K_{loop}} = \eta^{[0]} + \delta^{[0]} + C\delta^{[0]} + C^2\delta^{[0]} + \ldots C^{K_{loop}}\delta^{[0]}, \tag{4.7}$$

where $\eta^{K_{loop}}$ and $\delta^{[0]}$ are vectors of values over the interval $[t_{n,0}, t_{n,M} = t_{n+1}]$, $\delta^{[0]}$ is the approximation to $e^{[0]}$ at the first correction loop, and C is the matrix that incorporates the various methods and integrations required to update the error correction from $\delta^{[k-1]}$ to $\delta^{[k]}$. If we can estimate some size of C, then we will not need to calculate $\delta^{[k]}$ for each correction before determining whether to adapt the stepsize. A possible algorithm, for each step Δt, follows.

1. Find the prediction $\eta^{[0]}$.
2. Find the first correction $\delta^{[0]}$.
3. Find C.
4. Estimate the size of C. Use the size of C to estimate the size of C^2, C^3, ..., $C^{K_{loop}}$ (as few or as many as needed).
5. Determine whether the size of C (or C^2, ...) should give an acceptable error that is within the desired tolerance.

6. If the size is acceptable, perform the appropriate number of correction loops. If the size is not acceptable, refine or enlarge the stepsize as needed, and then begin again with finding the prediction, step 1.

4.5 Conclusions

An adaptive implementation of IDC-RK and IDC-ARK methods is straightforward, as well as necessary, since many problems are intractable without adaptive time-stepping. Semi-implicit IDC-ARK and IDC-FEBE methods can sometimes present an efficiency advantage over ARK methods in an adaptive time-stepping setting, while at other times, embedded ARK methods are more efficient. We expect clearer advantages when comparing adaptively implemented implicit IDC-RK methods with implicit RK methods, and in solving other IVPs such as initial layer IVPs.

Chapter 5

Split IDC Methods

5.1 Introduction and Motivation

In this chapter, we are motivated by the general need for studying plasma physics through modeling and numerical simulations, since many natural and industrial processes (such as solar weather, integrated circuits manufacturing, military and space research, and alternative energy sources) involve plasmas. Our contribution to this area of research is the development of novel split Vlasov solvers that are constructed to high order in time via IDC methods. In Section 5.1.1, we summarize a review of developments in plasma models and simulations, and in Section 5.1.2, we focus more specifically on the Vlasov-Poisson system, some popular methods of numerical solution, and introduce our work.

5.1.1 General Motivation: Modeling and Simulating Plasmas

The information outlined in this section is a summary of the excellent review, [74], which contains many more details and clarifications of the history of plasma physics,

modeling, and simulations. First we consider some components of a self-consistent plasma model to set the stage for the various developments in modeling and numerics. Such a model usually contains some or all of the following components: some method of calculating velocity, mass density, and pressure, if the plasma is modeled as a fluid; some means of closing the Navier-Stokes equations; some means of calculating density, temperature, and velocity of a species in a plasma; a method of obtaining the electron distributions when they are not Maxwellian due to nonequilibrium conditions. The multitude of factors that might be considered in a plasma model is not limited to these components and is such that fully treating all contributing factors is not reasonable analytically nor computationally, so existing models are of necessity reduced models. Some of these reduced descriptions are touched on next.

In plasmas, electron transport has frequently been modeled by a Boltzmann-type equation, which is numerically solved ot find the electron distribution function. One popular means of solution is to first simplify the problem by representing the distribution function by the first two terms of its expansion (e.g., an expansion in spherical harmonics), commonly called Lorentz' two-term expansion. Later, higher order approximations that included more terms of the expansion were introduced to overcome some limitations of two-term expansions. Also, the means of calculating some transport coefficients using the Boltzmann equation and its solutions were developed, which is important both for comparison with and adding insight to experimental results.

Considering the ratios of the densities of the various particles that compose a plasma is also meaningful, and when plasmas are in or near local thermal equilibrium, then the mass action law can be used to calculate these ratios, frequently through minimization of a variational problem. Other factors such as excited states and departures from the ideal gas must also be heeded, however.

Modeling methods such as Boltzmann equations and local thermal equilibrium

methods are known as continuum (or fluid) representations since they are continuous with respect to space and describe averaged quantities. However, continuum models are not always appropriate; e.g., when deviations from equilibrium are large. Boltzmann equations are frequently called kinetic equations, though kinetic may have a different meaning and is sometimes used to refer to particle methods also. Particle methods are one alternative to continuum methods that in many instances may capture the physics more realistically. Often particle methods (such as Monte Carlo methods, which track a collection of particles in space and time, where the particles represent the species population) and, more recently, molecular dynamics are used to study the particle behaviors, but they are computationally demanding. It is increasingly popular to combine various methods in studying plasmas, so that, for example, one part of the problem is handled with a continuum method, while another part is handled with a particle method.

Although there are many methods currently available, improvements are still vital to effectively model and simulate plasmas. In the next section, we describe more specifically Vlasov equations, which can be considered collisionless Boltzmann equations, and introduce how we contribute to the ongoing research by presenting our novel Vlasov solver.

5.1.2 Specific Motivation: Improved Vlasov Solvers

As an initial study, the work in this chapter focuses on the collisionless kinetic model for a cold plasma, with mobile electrons and an immobile ion background, though eventually such a work is meant to be expanded to other models and equations. The

model is given by the Vlasov–Poisson equations,

$$f_t + v \cdot \nabla_x f + E(t,x) \cdot \nabla_v f = 0, \tag{5.1}$$
$$E(t,x) = -\nabla_x \phi, \quad -\Delta_x \phi = -1 + \rho(t,x),$$

where $f(t,x,v)$ describes the probability of finding a particle with velocity v at position x at time t, E is the electric field, ϕ is the self-consistent electrostatic potential, and $\rho(t,x) = \int f(t,x,v)dv$ is the electron charge density and the 1 represents the uniformly distributed infinitely massive ions in the background. All physical constants in (5.1) have been normalized to one. Other models include the Vlasov-Maxwell equations, which capture the main behavior of the particle interactions and the self-consistent field, whenever the Debye sphere contains a large enough number of particles, and collisional effects can be neglected. If collisional effects should be considered, there are numerous options, including the BGK-Poisson, BGK-Maxwell, or Fokker-Planck equations [67, 22, 10].

Various approaches for the numerical solution of Vlasov equations include, but are not limited to, Eulerian, Lagrangian, and semi-Lagrangian methods. Eulerian methods solve (5.1) over a fixed xv-grid, and [26] compares some of these methods. The fixed numerical grid allows for easy implementation, but a CFL condition is usually present. Also, as the problem dimension increases, Eulerian methods have the disadvantage of a significant increase in computational cost; however, they may employ high order accurate numerical methods to overcome this cost by using a coarser grid. In places where the distribution is small, they can frequently achieve high precision [67, 26].

Lagrangian methods involve tracking portions of the plasma as they evolve in phase space by introducing a flow map of the electron distribution, reformulating (5.1) into a system of ODEs (Newton's equations of motion) in terms of the flow

map, discretizing Newton's equations, and using the resulting flow map solution to procure the electron distribution, f. Classical Lagrangian methods include Particle-in-Cell (PIC) methods, such as those methods reviewed in [75]. PIC methods have the advantage of accurately maintaining the physics of the problem, including the nonlinear effects, but require significant computational time and memory and may have difficulties resolving the tail of the particle distribution due to numerical noise.

Semi-Lagrangian methods combine some of the aspects of both Eulerian and Lagrangian methods. Essentially, semi-Lagrangian methods find the distribution function by following the characteristics backward to their base values, which are interpolated from a fixed grid. Their application to Vlasov equations was presented in [13], and later applied beyond the electrostatic system to guiding center approximations and a reduced relativistic Vlasov equation in [71].

Vlasov solvers using second order accurate Strang splitting [73] in time and either Fourier or spline interpolation were constructed in [13]. Several others have combined the splitting from [13] with various interpolation methods to obtain Euler or semi-Lagrangian methods. For example, semi-Lagrangian and flux balance methods using pointwise weighted essentially nonoscillatory (PWENO) interpolation was applied together with Strang splitting in [12]. In some cases they are able to avoid spurious oscillations without repressing physical oscillations. In these methods, the flux balance method is conservative, while the semi-Lagrangian method is not. Conservative methods, which conserve certain physical properties, applied to Vlasov equations are also presented in [23, 67]. Methods in [23] include a parabolic spline method and utilization of a cubic spline, with unknowns reconstructed on a nonuniform mesh in a conservative way. By including slope-limiters, they were able to maintain positivity. Their spline-based methods appeared to capture fine filamentation more precisely than Lagrangian-based interpolation methods. Conservative semi-Lagrangian WENO Vlasov solvers with dimensional Strang splitting are

given in [67]. The CFL time step restriction was removed, and the higher order spatial methods effectively captured the physics which the Vlasov equations describe. When resolved equally, the ninth order spatial method was four times cheaper than the third order method [67].

[12] mentions that, although semi-Lagrangian methods are not limited by a CFL condition, they are not easily coupled with a collisional term that is stiff compared to the transport terms. This limitation is because the accuracy in time is not greater than second order in nonstiff regions when, as is the typical choice, Strang splitting is applied. Thus we are motivated to examine high order splitting for Vlasov equations. Several constructions of high order splitting methods have been developed [77, 38, 69, 46, 49, 50, 39]. These are equivalent to composition methods, as can be seen in [77, 38]. [50] use Richardson extrapolation and defect correction (in the differential form) to construct high order splitting methods. They also compare some of these methods, and in particular, they comment that Yoshida's high order splitting methods require a number of coefficients that increases exponentially with the order of the splitting method, although they advantageously may be constructed to arbitrary (even) orders. Unlike [77], [39] finds higher order splitting methods that do not require a backward step for orders greater than two by using complex rather than only real coefficients in the methods; however, the number of coefficients required scales similarly to [77]. A fourth order splitting method for a linear Vlasov equation ($E = E(x,t)$, but E does not depend on the distribution, effectively giving a drifting rotating problem, see (5.151) in Section 5.9.3 below) was presented in [69]. The method uses cubic spline interpolation in both x and v, and the fourth order time splitting method is a composition of grid shifts. A CFL condition is given. Note however, that the application of this fourth order method to a nonlinear problem is not clear.

Motivated by Schaeffer's paper, and by the success of the Vlasov solver in [67],

we expand the results from [67] to examine the same spatial discretization coupled with higher than 2nd order time splitting. In particular, we focus on using Integral Deferred Correction (IDC) methods to increase the order of low order splitting methods, and we examine the effectiveness of our novel construction in solving equations similar in form to both linear and nonlinear Vlasov equations, generalizable to solving other PDEs with split operators. Although we do not yet consider collisional terms, they are the logical next step in our work.

IDC methods are motivated by defect correction methods [72, 27] and, more recently, Spectral Deferred Correction (SDC) methods [25]. By construction, they are accurate and efficient time integrators because they easily extend simple lower order methods to higher order schemes by correcting provisional solutions. Other related methods for problems containing stiff terms include semi-implicit SDC, multi-implicit SDC, and Krylov deferred correction [60, 47, 9, 41]. SDC methods, IDC methods, and their variants can be applied in areas such as chemical rate equations, hyperbolic conservation laws with or without relaxation, Vlasov equations in the plasma physics setting, and other similar (frequently multi-scale) problems [60, 47, 9, 41, 15]. Additionally, recent developments allow parallelization of IDC algorithms, opening up new possibilities for increased computational speed [16].

The current investigation of splitting methods that use IDC integrators for PDEs begins with a review of some basic concepts needed for understanding the numerical solution of PDEs and an introduction to WENO methods, which we use in our numerical tests, in Sections 5.2 and 5.3. Then we review the fundamentals of semi-Lagrangian methods in Section 5.4 and of low order splitting methods in Section 5.5. Section 5.6 presents an overview of IDC methods and explains the construction of a high order splitting method by utilizing low order splitting methods. In particular, the type of nonlinearity in the Vlasov system requires a new formulation of the error equation and the residual in the IDC correction loop. Next we present an analysis

comparing our high order split IDC methods with Yoshida's high order split methods described in [77]. A proof of mass conservation is given in Section 5.8. Numerical results are in Section 5.9. We apply the split IDC methods to constant advection, rotation, drifting rotation, and Vlasov-Poisson problems, where we study classic plasma problems, such as the warm two stream instability and Landau damping. We wish to determine the benefits and limitations of a solver that is not only higher order in space but also higher order in time in a semi-Lagrangian setting.

5.2 Method of Lines

Sometimes time-dependent PDEs can be solved numerically by discretizing the spatial derivatives first to obtain a semi-discretized system of ODEs, then numerically integrating the system of ODEs. This procedure is called the method of lines. It typically can be used when the time variable is distinct from the space variable(s), and the solution does not have a sharp front that is a function of both space and time [2].

For example, a semi-discretization of the linear advection equation

$$\partial_t u + c\partial_x u = 0, \quad x \in [0,1], \quad t \geq 0,$$

in space via a simple upwind scheme gives N_x ODEs

$$\partial_t U_j + c\frac{U_j - U_{j-1}}{\Delta x} = 0, \quad j = 1, \ldots N_x, \tag{5.2}$$

where $U_j = u(x_j)$, $x_j = j\Delta x$, and $\Delta x = (1-0)/N_x$. Then each ODE in (5.2) can be solved by the integrator of choice, e.g. backward Euler. Here we apply method of lines to hyperbolic PDEs, such as constant advection equations, a rotating problem,

and a Vlasov-Poisson system and use IDC methods for the time integration.

5.3 WENO

Many times, conventional spatial discretizations applied to problems that have piecewise smooth solutions containing discontinuities result in spurious oscillations in the numerical solution. The purpose of Essentially Non-oscillatory (ENO) and Weighted Essentially Non-oscillatory (WENO) methods is to make a higher order spatial approximation of problems that have piecewise smooth solutions containing discontinuities. In particular, ENO and WENO are designed for hyperbolic conservation laws, the background of which is outlined in Section 1.4. The following information is compiled from [66, 51, 70]. ENO and WENO are in fact procedures within a higher order spatial approximation procedure, so we outline the necessary elements for the entire procedure.

Two general categories of spatial discretizations are Finite Difference (FD) and Finite Volume (FV) methods. FD methods use pointwise values of the solutions and approximate the solutions from the differential form of an equation, e.g., from an equation such as (1.12), whereas FV methods use cell averages of the solutions and approximate the solutions from the integral form of an equation, e.g., from an equation such as (1.14).

By way of illustration, consider FV methods applied to the integral form of a hyperbolic conservation law, integrated over the cell $I_i = [x_{i-1/2}, x_{i+1/2}]$.

$$\int_{I_i} u_t dx + \int_{I_i} f(u)_x dx = 0 \tag{5.3}$$

becomes

$$\partial_t \int_{I_i} u dx + f(u(x_{i+1/2})) - f(u(x_{i-1/2})) = 0, \tag{5.4}$$

and rearranging and dividing by Δx, we obtain

$$\partial_t \frac{1}{\Delta x} \int_{I_i} u dx = -\frac{1}{\Delta x} \left(f(u(x_{i+1/2})) - f(u(x_{i-1/2})) \right). \tag{5.5}$$

The cell average is defined in the usual way as

$$\bar{u}_i(t) = \frac{1}{\Delta x} \int_{I_i} u dx. \tag{5.6}$$

Since, for FV methods, only the cell averages are known, but the flux $f(u(x_{i+1/2}))$ is at point values, then an approximation of $f(u(x_{i+1/2}))$ is used instead, given by the numerical flux $\hat{f}_{i+1/2}$. Thus, for MOL, discretizing in x first, we obtain an ODE of the form

$$\partial_t \bar{u}_i(t) = -\frac{1}{\Delta x} \left(\hat{f}_{i+1/2}(t) - \hat{f}_{i-1/2}(t) \right). \tag{5.7}$$

Finally, we obtain the updated cell average, $\bar{u}_i^{n+1}$, at time t^{n+1}, by discretizing in t, e.g., with forward Euler:

$$\frac{\bar{u}_i^{n+1} - \bar{u}_i^n}{\Delta t} = -\frac{1}{\Delta x} \left(\hat{f}_{i+1/2}^n - \hat{f}_{i-1/2}^n \right). \tag{5.8}$$

The mystery remaining is how to obtain $\hat{f}_{i\pm1/2}$, the numerical flux at the cell boundaries. We would like to use $f(u(x_{i+1/2}))$, but we do not know the value of u at the cell boundary $x_{i+1/2}$. We only know $\bar{u}_i$ to the left (denoted by $^-$) of the cell boundary or $\bar{u}_{i+1}$ to the right (denoted by $^+$). Hence the calculation of the

numerical flux must depend upon the left ($^-$) and right ($^+$) values of approximations to $u_{i+1/2}$, or on the left and right cell averages:

$$\hat{f}_{i+1/2} = \hat{f}(u^-_{i+1/2}, u^+_{i+1/2}) \tag{5.9}$$

$$= \hat{f}(\bar{u}_i, \bar{u}_{i+1}), \tag{5.10}$$

where $\hat{f}_{i+1/2}$ should increase with respect to $\bar{u}_i$ and decrease with respect to $\bar{u}_{i+1}$.

A simple example of a numerical flux for a first order scheme is illustrated below. Since equation (1.12) can be written as

$$u_t + f'(u)u_x = 0, \tag{5.11}$$

then, considering that $f'(u)$ is related to the slope of the characteristic, it is natural to choose the numerical flux as

$$\hat{f}(\bar{u}_i, \bar{u}_{i+1}) = \begin{cases} f(\bar{u}_i) & \text{if} \quad f'(u) > 0 \\ f(\bar{u}_{i+1}) & \text{if} \quad f'(u) < 0 \end{cases} \tag{5.12}$$

5.3.1 Reconstruction

For higher than first order schemes one needs a process called reconstruction. The reconstruction problem is: given the cell averages $\bar{u}_i$, reconstruct the solution $u(x_{i+1/2})$ at the cell boundary $x_{i+1/2}$ with higher order. For the first order stencil, we use only the immediately neighboring cell, either I_i or I_{i+1}. More neighboring cells are used to achieve a higher order stencil. In general, let

$$S_r = \{I_{i-r}, \ldots, I_i, \ldots, I_{i-r+k-1}\} \tag{5.13}$$

denote a stencil, where k cells are used for the stencil, and $r \in \{k-1, \ldots, 1, 0, -1\}$. In the hyperbolic setting, which we delineate here, one must always use an odd number k of neighboring cells because the characteristics have a direction. One chooses more neighbors from the direction from which the characteristic comes. In a hyperbolic setting, choosing an even number of neighboring cells results in an unstable method (e.g., centered stencil). Under the right conditions, the order of accuracy of the reconstruction equals the number of neighboring cells used in the stencil. This process is very similar to interpolation, e.g., for a 3rd order reconstruction where $k = 3$, we find the polynomial $p(x) = a_0 + a_1 x + a_2 x^2$ such that p satisfies

$$\frac{1}{\Delta x} \int_{I_{i-1}} p(x)dx = \bar{u}_{i-1}, \tag{5.14}$$

$$\frac{1}{\Delta x} \int_{I_i} p(x)dx = \bar{u}_i, \tag{5.15}$$

$$\frac{1}{\Delta x} \int_{I_{i+1}} p(x)dx = \bar{u}_{i+1}. \tag{5.16}$$

Then, since a_0, a_1, and a_2 are known, we can estimate $u_{i+1/2}$ from the left as

$$u^-_{i+1/2} = p(x_{i+1/2}) \tag{5.17}$$

when p is constructed from a left stencil, $S_1 = \{I_{i-1}, I_i, I_{i+1}\}$. Similarly, $u^+_{i+1/2} = p(x_{i+1/2})$, when p is constructed from a right stencil, $S_0 = \{I_i, I_{i+1}, I_{i+2}\}$.

Two methods of reconstructing solutions $u^-_{i+1/2}$, $u^+_{i+1/2}$ at the cell boundaries for a higher order approximation scheme are ENO and WENO methods.

5.3.2 ENO

When the solution u contains discontinuities, using all k cells in a stencil S_r gives oscillatory behavior near the discontinuities. Consider the case when $k = 3$. Away

from a discontinuity, we use three cells for the stencil, e.g., $S_1 = \{I_{i-1}, I_i, I_{i+1}\}$, and achieve a third order approximation. Near a discontinuity, e.g., at $x_{i+1/2}$, where the characteristics move from left to right, instead of using three cells as in S_1, consider two smaller overlapping stencils $\tilde{s}_1 = \{I_{i-1}, I_i\}$, $\tilde{s}_0 = \{I_i, I_{i+1}\}$. Note that stencil $\tilde{s}_1$ will not cause oscillations in a reconstructed solution because it contains cells only on one side of the discontinuity, whereas stencil $\tilde{s}_0$ contains the discontinuity and thus will result in oscillatory behavior. Hence in the case $|\bar{u}_{i-1} - \bar{u}_i| << |\bar{u}_i - \bar{u}_{i+1}|$, ENO will choose stencil $\tilde{s}_1$ as the "better" stencil. Note that choosing these stencils near the discontinuity will result in a lower order approximation, but only near the discontinuity. Away from the discontinuity, the approximation remains fully third order.

In general for FV methods, ENO uses a linear combination of the cell averages $\bar{u}_i$ to reconstruct the solutions $u_{i+1/2}$ at the cell boundaries. ENO with FD methods is similar, except the numerical flux (rather than the solution) is reconstructed at the cell boundaries. The details are explained below for WENO methods. For an ENO reconstruction method that uses k cells, the method is order k away from the discontinuities, and lower order at the discontinuities.

5.3.3 WENO

WENO is similar to ENO, except WENO uses weighted combinations of the stencils to achieve higher order approximations. A WENO method where each stencil contains k cells has order $2k - 1$ away from discontinuities and order k near discontinuities. We present the procedures for WENO methods in detail for the FD case involving one-dimensional scalar flux splitting, followed by an extension to 1-D systems.

1-D Scalar Case

First, we present the following outline, given by Procedure 2.5 and 2.2 of [70], of the spatial approximation of equation (1.12).

1. Since the numerical flux in equation (5.9) should increase in its first variable and decrease in its second variable, the first step is to form a smooth flux splitting

$$f(u) = f^+(u) + f^-(u), \tag{5.18}$$

where

$$\frac{df^+}{du}(u) \geq 0, \quad \frac{df^-}{du}(u) \leq 0. \tag{5.19}$$

For example, the Lax-Friedrichs flux given by

$$f^+(u) = \frac{1}{2}(f(u) + \alpha u), \tag{5.20}$$

$$f^-(u) = \frac{1}{2}(f(u) - \alpha u), \tag{5.21}$$

where $\alpha = \max_u |f'(u)|$.

2. Identify the cell average of some function $v(x)$ at x_i with one part of the split flux, i.e., set

$$\bar{v}_i = f^+(u_i). \tag{5.22}$$

Then reconstruct the cell boundary values from the left, $v^-_{i+1/2}$, for all i, e.g., via a WENO method as in the subsequent procedure.

3. Set the positive numerical flux equal to the reconstructed value from the pre-

vious step:

$$\hat{f}^+_{i+1/2} = v^-_{i+1/2}. \tag{5.23}$$

4. Now set

$$\bar{v}_i = f^-(u_i), \tag{5.24}$$

and reconstruct the cell boundary values from the right, $v^+_{i+1/2}$, for all i, e.g., via a WENO method as in the subsequent procedure.

5. Set the negative numerical flux equal to the reconstructed value from the previous step:

$$\hat{f}^-_{i+1/2} = v^+_{i+1/2}. \tag{5.25}$$

6. Form the numerical flux from the positive and negative fluxes at the cell boundaries:

$$\hat{f}_{i+1/2} = \hat{f}^+_{i+1/2} + \hat{f}^-_{i+1/2}, \tag{5.26}$$

i.e.,

$$\hat{f}_{i+1/2} = v^-_{i+1/2} + v^+_{i+1/2}. \tag{5.27}$$

7. Form the ODE

$$\frac{du_i(t)}{dt} = -\frac{1}{\Delta x}(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}), \tag{5.28}$$

to be solved by the desired numerical integrator.

Now we present the 1-D WENO reconstruction procedure from [70]. Given the cell averages $\bar{v}_i$ of some function $v(x)$ for each cell I_i, the goal is to obtain upwind-based $(2k-1)$th order approximations, $v^+_{i-1/2}$ and $v^-_{i+1/2}$, to $v(x)$ at the cell boundaries.

1. First find the k reconstructed values

$$v^{(r)}_{i+1/2} = \sum_{j=0}^{k-1} c_{rj} \bar{v}_{i-r+j}, \tag{5.29}$$

where the coefficients c_{rj} come from reconstructing the polynomial whose cell averages coincide with the cell averages of v (see, e.g., Section 5.3.1 for more details on the case $k = 3$), and the stencil $S_r = \{I_{i-r}, I_{i-r+1}, \ldots, I_{i-r+k-1}\}$ containing k cells is used, for $r = 0, 1, \ldots, k-1$.

Also obtain the k reconstructed values

$$v^{(r)}_{i-1/2} = \sum_{j=0}^{k-1} \tilde{c}_{rj} \bar{v}_{i-r+j} \tag{5.30}$$

$$= \sum_{j=0}^{k-1} c_{r,j} \bar{v}_{i-(r+1)+j}, \tag{5.31}$$

where $\tilde{c}_{rj} = c_{r-1,j}$, over the stencil S_r, for $r = -1, 0, 1, \ldots, k-1$.

2. Find the constants d_r and $\tilde{d}_r$ such that

$$v_{i+1/2} = \sum_{r=0}^{k-1} d_r v_{i+1/2}^{(r)} \tag{5.32}$$

$$= v(x_{i+1/2}) + \mathcal{O}(\Delta x^{2k-1}), \tag{5.33}$$

$$v_{i-1/2} = \sum_{r=0}^{k-1} \tilde{d}_r v_{i-1/2}^{(r)} \tag{5.34}$$

$$= v(x_{i-1/2}) + \mathcal{O}(\Delta x^{2k-1}). \tag{5.35}$$

Note that $\tilde{d}_r = d_{k-1-r}$.

3. Find the smoothness indicators

$$\beta_r = \sum_{l=1}^{k-1} \int_{I_i} \Delta x^{2l-1} \left(\frac{\partial^l p_r(x)}{\partial^l x} \right)^2 dx, \tag{5.36}$$

for upwinding in the left to right direction. For upwinding in the opposite direction, modify the procedure symmetrically with respect to $x_{i+1/2}$ to find the smoothness indicators $\tilde{\beta}_r$.

4. Form the weights ω_r and $\tilde{\omega}_r$:

$$\omega_r = \frac{\alpha_r}{\sum_{s=0}^{k-1} \alpha_s}, \quad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}, \tag{5.37}$$

$$\tilde{\omega}_r = \frac{\tilde{\alpha}_r}{\sum_{s=0}^{k-1} \tilde{\alpha}_s}, \quad \tilde{\alpha}_r = \frac{\tilde{d}_r}{(\epsilon + \tilde{\beta}_r)^2}, \tag{5.38}$$

for $r = 0, 1, \ldots, k-1$ and $0 < \epsilon << 1$.

5. Use the weights, ω_r and $\tilde{\omega}_r$, and the kth order reconstructions, $v_{i+1/2}^{(r)}$ and $v_{i-1/2}^{(r)}$, to form the $(2k-1)$th order reconstructions, $v_{i+1/2}^{-}$ and $v_{i-1/2}^{+}$, at

the cell boundaries:

$$v^-_{i+1/2} = \sum_{r=0}^{k-1} \omega_r v^{(r)}_{i+1/2}, \tag{5.39}$$

$$v^+_{i-1/2} = \sum_{r=0}^{k-1} \tilde{\omega}_r v^{(r)}_{i-1/2}. \tag{5.40}$$

Now, for example, we provide the various coefficients, smoothness indicators, and weights for the case $k = 3$, i.e., for a WENO method that is fifth order away from discontinuities and third order near discontinuities. The coefficients of the reconstructed polynomials (5.29) are

$$c_{00} = 1/3, \quad c_{01} = 5/6, \quad c_{02} = -1/6, \tag{5.41}$$

$$c_{10} = -1/6, \quad c_{11} = 5/6, \quad c_{12} = 1/3, \tag{5.42}$$

$$c_{20} = 1/3, \quad c_{20} = -7/6, \quad c_{20} = 11/6, \tag{5.43}$$

$$\tilde{c}_{00} = 11/6, \quad \tilde{c}_{01} = -7/6, \quad \tilde{c}_{02} = 1/3, \tag{5.44}$$

$$\tilde{c}_{10} = 1/3, \quad \tilde{c}_{11} = 5/6, \quad \tilde{c}_{12} = -1/6, \tag{5.45}$$

$$\tilde{c}_{20} = -1/6, \quad \tilde{c}_{21} = 5/6, \quad \tilde{c}_{22} = 1/3. \tag{5.46}$$

The constants d_r, $\tilde{d}_r$ are given by

$$d_0 = 3/10, \quad d_1 = 3/5, \quad d_2 = 1/10, \tag{5.47}$$

$$\tilde{d}_0 = 1/10, \quad \tilde{d}_0 = 3/5, \quad \tilde{d}_0 = 3/10. \tag{5.48}$$

The smoothness indicators are

$$\beta_0 = \frac{13}{12}(\bar{v}_i - 2\bar{v}_{i+1} + \bar{v}_{i+2})^2 + \frac{1}{4}(3\bar{v}_i - 4\bar{v}_{i+1} + \bar{v}_{i+2})^2, \tag{5.49}$$

$$\beta_1 = \frac{13}{12}(\bar{v}_{i-1} - 2\bar{v}_i + \bar{v}_{i+1})^2 + \frac{1}{4}(\bar{v}_{i-1} - \bar{v}_{i+1})^2, \tag{5.50}$$

$$\beta_2 = \frac{13}{12}(\bar{v}_{i-2} - 2\bar{v}_{i-1} + \bar{v}_i)^2 + \frac{1}{4}(\bar{v}_{i-2} - 4\bar{v}_{i-1} + 3\bar{v}_i)^2, \tag{5.51}$$

$$\tilde{\beta}_0 = \frac{13}{12}(\bar{v}_{i+3} - 2\bar{v}_{i+2} + \bar{v}_{i+1})^2 + \frac{1}{4}(\bar{v}_{i+3} - 4\bar{v}_{i+2} + 3\bar{v}_{i+1})^2, \tag{5.52}$$

$$\tilde{\beta}_1 = \frac{13}{12}(\bar{v}_{i+2} - 2\bar{v}_{i+1} + \bar{v}_i)^2 + \frac{1}{4}(\bar{v}_{i+2} - \bar{v}_i)^2, \tag{5.53}$$

$$\tilde{\beta}_2 = \frac{13}{12}(\bar{v}_{i+1} - 2\bar{v}_i + \bar{v}_{i-1})^2 + \frac{1}{4}(3\bar{v}_{i+1} - 4\bar{v}_i + \bar{v}_{i-1})^2. \tag{5.54}$$

1-D Systems

Now, instead of the scalar conservation law (1.12), consider the hyperbolic system of m equations,

$$\partial_t u + f'(u)\partial_x u = 0, \tag{5.55}$$

where the $m \times m$ Jacobian $f'(u)$ has m real eigenvalues,

$$\lambda_1(u) \leq \ldots \lambda_m(u), \tag{5.56}$$

and a complete set of independent eigenvectors (columns),

$$r_1(u), \ldots, r_m(u), \tag{5.57}$$

also referred to as *right eigenvectors*. The right eigenvectors can be written as columns of a matrix, $R(u) = (r_1(u) \ldots r_m(u))$. Then $R^{-1}(u)f'(u)R(u) = \Lambda(u)$,

where

$$\Lambda(u) = \begin{pmatrix} \lambda_1(u) & & \\ & \ddots & \\ & & \lambda_m(u) \end{pmatrix}. \tag{5.58}$$

$R^{-1}(u)$ can be written as $R^{-1}(u) = (l_1(u) \dots l_m(u))$, where the $l_i(u)$ are column vectors referred to as *left eigenvectors*, i.e., $l_i(u)f'(u) = \lambda_i(u)l_i(u)$. The simplest way to solve the system (5.55) is to apply the scalar WENO method to each of its m components. However, applying a WENO method to these conservative variables may not correctly account for the upwind direction. More reliably, one may solve (5.55) by first transforming the system from the conservative to characteristic variables, then applying a WENO method componenet-wise to the characteristic variables, and finally transforming back to the conservative variables. The procedure for solving (5.55) in the characteristic-wise manner using FD and flux splitting follows (from [70]).

1. For each fixed $x_{i+1/2}$, compute an approximation to $R(u_{i+1/2})$, $R^{-1}(u_{i+1/2})$, and $\Lambda(u_{i+1/2})$ using, e.g., a simple mean: $u_{i+1/2} \approx \frac{1}{2}(u_{i+1} + u_i)$.

2. Write $v_j = R^{-1}u_j$ and $g_j = R^{-1}f(u_j)$ for j in a neighborhood of i.

3. Apply the WENO procedure (component-wise) to v_j and g_j, rather than to u_j and f_j, for each lth component $v_{j,l}$, $g_{j,l}$, $l = 1, \dots, m$, where for a Lax-Friedrich's flux, we use

$$\alpha = \max_{1 \le j \le N} |\lambda_l(u_j)| \tag{5.59}$$

for the lth component of the characteristic variables, and obtain $\hat{g}^{\pm}_{i+1/2}$ (rather than $\hat{f}^{\pm}_{i+1/2}$).

4. Then transform back from characteristic to physical space:

$$\hat{f}^{\pm}_{i+1/2} = R\hat{g}^{\pm}_{i+1/2}. \tag{5.60}$$

5. Finally, form the flux $\hat{f}_{i+1/2} = \hat{f}^{+}_{i+1/2} + \hat{f}^{-}_{i+1/2}$ and solve:

$$\frac{du_i(t)}{dt} = -\frac{1}{\Delta x}(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}). \tag{5.61}$$

As an example, we calculate R, Λ, and R^{-1} for the shallow water equations given in [64].

$$\partial_t \begin{pmatrix} h \\ w \end{pmatrix} + \partial_x \begin{pmatrix} w \\ h + \frac{1}{2}h^2 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{h}{\epsilon}(\frac{h}{2} - v) \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\epsilon}(\frac{h^2}{2} - w) \end{pmatrix} \tag{5.62}$$

$$\begin{pmatrix} \partial_t h \\ \partial_t w \end{pmatrix} + \begin{pmatrix} \partial_x w \\ \partial_x h + h\partial_x h \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\epsilon}(\frac{h^2}{2} - w) \end{pmatrix} \tag{5.63}$$

$$\begin{pmatrix} \partial_t h \\ \partial_t w \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 1+h & 0 \end{pmatrix} \begin{pmatrix} \partial_x h \\ \partial_x w \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{\epsilon}(\frac{h^2}{2} - w) \end{pmatrix}. \tag{5.64}$$

The eigenvalues are $\lambda_1 = \sqrt{1+h}$, $\lambda_2 = -\sqrt{1+h}$, i.e.,

$$\Lambda = \begin{pmatrix} \sqrt{1+h} & 0 \\ 0 & -\sqrt{1+h} \end{pmatrix}. \tag{5.65}$$

Choosing the right eigenvectors $r_1(u) = \begin{pmatrix} 1 \\ \sqrt{1+h} \end{pmatrix}$, $r_2(u) = \begin{pmatrix} 1 \\ -\sqrt{1+h} \end{pmatrix}$, we obtain

$$R = \begin{pmatrix} 1 & 1 \\ \sqrt{1+h} & -\sqrt{1+h} \end{pmatrix}, \tag{5.66}$$

and

$$R^{-1} = \frac{1}{2} \begin{pmatrix} 1 & \frac{1}{\sqrt{1+h}} \\ 1 & -\frac{1}{\sqrt{1+h}} \end{pmatrix}. \tag{5.67}$$

5.4 Semi-Lagrangian Methods

In this section, we describe semi-Lagrangian methods. Suppose we want to solve the 1D normalized Vlasov-Poisson system

$$\partial_t f + v\partial_x f + E\partial_v f = 0, \tag{5.68}$$

$$\partial_{xx}\phi = 1 - \rho, \tag{5.69}$$

$$E = -\partial_x \phi, \tag{5.70}$$

$$f(0, x, v) = f_0(x, v), \tag{5.71}$$

where f denotes the exact solution, and $\rho = \int f dv$.

The solution f is constant along the particle trajectories. Assume the solution at time t_n is known. Then the solution at time t_{n+1} is given by

$$f(t_{n+1}, x, v) = f(t_n, X(t_n, t_{n+1}, x, v), V(t_n, t_{n+1}, x, v)),$$

where $(X(t_n, t_{n+1}, x, v), V(t_n, t_{n+1}, x, v))$ are the characteristic curves satisfying

$$\frac{dX}{dt} = V(t), \quad \frac{dV}{dt} = E(X(t), t).$$

For the semi-Lagrangian method, one approximates f at each space-velocity point (x_i, v_j), $i = 1 : N_x$, $j = 1 : N_v$ by updating f at each time step from its value at the base of the characteristic $(X(t_n, t_{n+1}, x_i, v_j), V(t_n, t_{n+1}, x_i, v_j))$. The value at the base is computed using high order interpolation, such as cubic splines or WENO reconstruction, from the known values on the grid. [26]

For example, we can solve $u_t + au_x = 0$ exactly, obtaining $u(x,t) = u_0(x - at)$, using a semi-Lagrangian method, as given in Algorithm 3.

1 Let u = vector of desired solution to be evaluated at $x = x_{new}$,
2 u_0 = vector of initial values already known at $x = x_{new}$.
3 Input $(x_{new}, u_0, a, \Delta t)$.
4 Set $x_{old} = x_{new} - a\Delta t$;
5 Account for boundary conditions, e.g., on a $2L$ periodic domain:
6 $x_{old} =$ mod $(x_{old} + L, 2L)$;
7 Interpolate u_0 to x_{old} values (from u_0 at x_{new}) to obtain u; e.g., via Matlab command:
8 $u = interp1(x_{new}, u_0, x_{old}, 'linear')$;

Algorithm 3: Example of solving a simple 1D advection equation via a semi-Lagrangian method.

For the Vlasov equation, it is advantageous to use a semi-Lagrangian method in conjunction with an operator splitting method because a convenient splitting choice allows for an exact solution of a 1D linear advection equation at each split step, as Section 5.5, following, elucidates.

5.5 Splitting Methods

fNow let us examine the idea of operator splitting to solve a PDE of the form

$$\partial_t f + A\partial_x f + B\partial_v f = 0, \quad f(x, v, t_0) = f_0(x, v), \tag{5.72}$$

where $A = A(v, t)$, $B = B(x, t)$. Note that the Vlasov system (5.1) satisfies these conditions with $A = v$, $B = E$. The PDE (5.72) may not be simple to solve, and in particular, for the Vlasov system (5.1) is two-dimensional and nonlinear. With a clever choice of operator splitting, one may instead solve several simpler subproblems exactly in order to approximate the solution to (5.72). For the Vlasov equation, these subproblems are one-dimensional and linear.

For example, we may approximate the solution by noting that (5.72) is equivalent to the sum of the following two 1D subproblems,

$$\partial_t f + A\partial_x f = 0, \quad \partial_t f + B\partial_v f = 0,$$

and, assuming reasonable discretizations of the derivatives in x and v, solving the split equations:

- Take a full timestep ($t_0 \to t_0 + \Delta t$) to solve:

$$\partial_t \tilde{f} + A\partial_x \tilde{f} = 0, \quad \tilde{f}(x, v, t_0) = f_0(x, v), \tag{5.73}$$

Since the exact solution is known, one may simply find the solution as

$$\tilde{f}(x, v, t_0 + \Delta t) = f_0(x - A\Delta t, v). \tag{5.74}$$

- Take another full timestep ($t_0 \to t_0 + \Delta t$) to solve:

$$\partial_t \tilde{\tilde{f}} + B\partial_v \tilde{\tilde{f}} = 0, \quad \tilde{\tilde{f}}(x, v, t_0) = \tilde{f}(x, v, t_0 + \Delta t). \tag{5.75}$$

Similarly to above, one may find the solution as

$$\tilde{\tilde{f}}(x, v, t_0 + \Delta t) = \tilde{f}(x, v - B\Delta t, t_0 + \Delta t). \tag{5.76}$$

Then $\tilde{\tilde{f}}$ is a first order solution to (5.72), if the operators $A\partial_x$ and $B\partial_v$ do not commute, and if (5.73) and (5.75) are solved with a numerical method that is at least first order in time. I.e., the error in the solution is due to both the splitting and the choice of numerical method(s) used at each step of the splitting If $A\partial_x$ and $B\partial_v$ do commute, then there is no splitting error. [73, 24]. However, for this splitting, note that each subproblem is a 1D linear advection equation, which can be solved exactly via a semi-Lagrangian method, so the only error is the splitting error.

The popular example of a 2nd order splitting method is Strang splitting (other names include Marchuk splitting, fractional step method) [13, 73, 58]. We describe below one timestep of size Δt, from t_0 to $t_0 + \Delta t$.

- Take a half timestep ($t_0 \to t_0 + \Delta t/2$) to solve:

$$\partial_t f^* + A\partial_x f^* = 0, \qquad f^*(x, v, t_0) = f_0(x, v). \tag{5.77}$$

Since the exact solution is known, one may simply find the solution as

$$f^*(x, v, t_0 + \Delta t/2) = f_0(x - A\Delta t/2, v). \tag{5.78}$$

- Then take a full timestep ($t_0 \rightarrow t_0 + \Delta t$) to solve:

$$\partial_t f^{**} + B\partial_v f^{**} = 0, \qquad f^{**}(x, v, t_0) = f^*(x, v, t_0 + \Delta t/2). \tag{5.79}$$

Similarly to above, one may find the solution as

$$f^{**}(x, v, t_0 + \Delta t) = f^*(x, v - B\Delta t, t_0 + \Delta t/2). \tag{5.80}$$

- Finally, take another half timestep ($t_0 + \Delta t/2 \rightarrow t_0 + \Delta t$) to solve:

$$\partial_t f_{split} + A\partial_x f_{split} = 0, \qquad f_{split}(x, v, t_0 + \Delta t/2) = f^{**}(x, v, t_0 + \Delta t). \tag{5.81}$$

Similarly to above, one may find the solution as

$$f_{split}(x, v, t_0 + \Delta t) = f^*(x - A\Delta t/2, v, t_0 + \Delta t). \tag{5.82}$$

f_{split} is a second order solution to (5.72), if the operators $A\partial_x$ and $B\partial_v$ do not commute, and if (5.73) and (5.75) are solved with a numerical method that is at least second order in time. However, for this splitting, just as for the first order splitting, note that each subproblem is a 1D linear advection equation, which can be solved exactly via a semi-Lagrangian method, so the only error is the splitting error. As mentioned above, commuting operators will not give a splitting error [73, 13].

5.6 IDC with Splitting Methods

In this section, we expand the results from [67] to examine the same spatial discretization coupled with higher than 2nd order time splitting, using IDC methods to increase the order of low order splitting methods, and in following sections, we examine the effectiveness of our novel construction in solving equations of the form (5.72). Some aspects of our work are similar to constructions in [46, 49, 50]. Note we introduce a special change for the IDC method's error equation for the Vlasov system, but for constant advection, rotation, and drifting rotation problems, the error equation is the same as in previous versions of IDC.

5.6.1 Overview of IDC methods

The basic construction of IDC methods is the same as for SDC methods [25, 18, 15]. Suppose we wish to solve the 1D version of (5.1) (given below by (5.88)) to the final time T. The time interval, $[0, T]$, is discretized into intervals $[t_n, t_{n+1}]$, $n = 0, 1, \ldots, N$, such that

$$0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{5.83}$$

with timestep

$$\Delta t_n = t_{n+1} - t_n. \tag{5.84}$$

Each interval $[t_n, t_{n+1}]$ is discretized again into M uniform subintervals with quadrature nodes denoted by

$$t_{n,0} = t_n, \tag{5.85}$$

$$t_{n,m} = t_{n,0} + m\delta t, \quad m = 0, 1, \ldots, M, \tag{5.86}$$

where $\delta t = \frac{\Delta t_n}{M}$. We apply the IDC method identically on each time interval $[t_n, t_{n+1}]$, so we drop the subscript n so that the notation in (5.86) becomes

$$t_m = t_0 + m\delta t, \quad m = 0, 1, \ldots, M. \tag{5.87}$$

For one time interval $[t_n, t_{n+1}]$, the IDC method calculates a provisional solution and improves that solution through successive correction loops, which reduce the size of the error and improve the order of accuracy of the method. Using a known solution at the initial time, t_0, the IDC algorithm can be summarized as

1. **Prediction** ($j = 0$th loop)

   Solve the IVP over the grid (5.87) via a simple numerical integration method, e.g. 1st or 2nd order splitting in time, to obtain a provisional solution, $\eta^{[0]}$.

2. **jth Correction** (for $j = 1, \ldots, J$ loops)

   (a) Solve an error equation (see Section 5.6.3, below) to approximate the error, $e^{[j-1]}$, between the exact solution and the numerical solution from the previous, $(j-1)$th, loop via a simple numerical integration method, e.g. 1st or 2nd order splitting in time.

   (b) Update the numerical solution as $\eta^{[j]} = \eta^{[j-1]} + e^{[j-1]}$.

Then the solution after the Jth correction loop becomes the initial condition for the next interval, $[t_{n+1}, t_{n+2}]$, in (5.83). Using a first order split prediction and J first order split corrections gives a split IDC method that is $\min(J+1, M+1)$th order accurate in time. Using a second order split prediction and J second order split corrections gives a split IDC method that is $\min(2(J+1), M+1)$th order accurate in time. The details of the prediction and correction loops for equations of the same form as the Vlasov-Poisson system are explained in Sections 5.6.2 and 5.6.3, following.

5.6.2 Prediction Loop

The split IDC prediction loop solves an IVP via a straightforward application of some splitting method as presented in Section 5.5. We clarify a few details as follows. Suppose we want to solve the 1D Vlasov-Poisson system

$$\partial_t f + v\partial_x f + E^f \partial_v f = 0, \tag{5.88}$$

$$\partial_{xx}\phi = 1 - \rho^f, \tag{5.89}$$

$$E^f = -\partial_x \phi, \tag{5.90}$$

$$f(0, x, v) = f_0(x, v), \tag{5.91}$$

where f denotes the exact solution, E^f is the electric field calculated exactly from f, and $\rho^f = \int f dv$.

We will solve the system for $x \in [0, L]$, periodic in x with period L, $v \in \mathcal{R}$, and $t \in [t_0, t_M]$, where M is fixed and the interval is equally subdivided as

$$t_0 < t_1 < \cdots < t_{M-1} < t_M, \quad \delta t = t_{m+1} - t_m. \tag{5.92}$$

Note that periodicity in x is equivalent to

$$\frac{1}{L}\int_0^L \rho^f dx = 1 \tag{5.93}$$

[22].

For a prediction loop of IDC, we could solve (5.88) using the first order splitting method in the same form as (5.73),(5.75) (replacing Δt with δt):

$$\partial_t \tilde{\eta} + v\partial_x \tilde{\eta} = 0, \tag{5.94}$$

$$-\partial_x E^{\tilde{\eta}} = \frac{1}{L}\int_0^L \rho^{\tilde{\eta}} dx - \rho^{\tilde{\eta}}, \tag{5.95}$$

$$\partial_t \eta + E^{\tilde{\eta}}\partial_v \eta = 0, \tag{5.96}$$

where η denotes the preliminary split solution, which is $\mathcal{O}(\delta t) = \mathcal{O}(\Delta t)$ in time, and $\rho^{\tilde{\eta}} = \int \tilde{\eta} dv$. Similarly, one could use the second order splitting given by (5.77), (5.79), and (5.81), calculating the electric field after (5.77) and using it in (5.79).

5.6.3 Splitting for Correction

In the correction loop, we wish to solve for a correction error e defined as

$$e = f - \eta. \tag{5.97}$$

The residual r and the electric field correction E^e are defined as

$$r \doteq \partial_t \eta + v\partial_x \eta + E^f \partial_v \eta \tag{5.98}$$

$$= (\partial_t \eta + v\partial_x \eta + E^\eta \partial_v \eta) + E^e \partial_v \eta \tag{5.99}$$

$$\doteq r^\eta + r^e, \tag{5.100}$$

$$E^e \doteq E^f - E^\eta, \tag{5.101}$$

where r^η depends only on the provisional solution η (and hence is known), and r^e depends on the unknown error e. Since

$$-\partial_x E^f = 1 - \rho^f, \tag{5.102}$$

$$-\partial_x E^\eta = \frac{1}{L}\int_0^L \rho^\eta dx - \rho^\eta, \tag{5.103}$$

$$f = \eta + e, \tag{5.104}$$

then we require

$$-\partial_x E^e = \frac{1}{L}\int_0^L \rho^e dx - \rho^e, \tag{5.105}$$

$$\tag{5.106}$$

where $\rho^e = \int e dv$ and thus

$$\int_0^L \rho^\eta dx + \int_0^L \rho^e dx = \int_0^L \rho^f dx = L. \tag{5.107}$$

Note that a conservative method of calculating η ought to result in

$$\int_0^L \rho^\eta dx = \int_0^L \rho^f dx, \quad \text{i.e.,} \quad \int_0^L \rho^e dx = 0, \tag{5.108}$$

but we should not assume this, since a solution to a system such as (5.88) requires that the integral of the right hand side of (5.89) in x from 0 to L is zero.

Now differentiating e with respect to t, we obtain

$$\partial_t e = \partial_t f - \partial_t \eta \tag{5.109}$$

$$= -v\partial_x f - E^f \partial_v f - (r - v\partial_x \eta - E^f \partial_v \eta) \tag{5.110}$$

$$= -v\partial_x e - E^f \partial_v e - r, \tag{5.111}$$

and rewrite as the *error equation*

$$\partial_t e + v\partial_x e + (E^\eta + E^e)\partial_v e = -r^\eta - r^e. \tag{5.112}$$

Note that r^e is a new term in the error equation and is not found in previous versions of SDC or IDC. It is required for the Vlasov system's error equation, but for constant advection, rotation, and drifting rotation problems, this extra term is zero, giving the same error equation as expected from prior IDC methods.

In Algorithm 4, we present a pseudo-algorithm to solve the error equation (5.112) using first order splitting in time. To solve the error equation (5.112) from t_0 to t_1, we first assume that there is no error at $t = t_0$, so $e|_{t=t_0} = 0$ and $E^e|_{t=t_0} = 0$. At line 12 of the algorithm, we approximate the integral of the residual r^η, rather than the residual itself, using an integration matrix resulting from the integral of a Lagrange interpolant, as described in [25, 18]. To simplify the algorithm, we use $\delta t E^e|_{t=t_m} \partial_v \eta|_{t_m}$ to approximate the integral of r^e. However, for a higher than first order correction loop, a more accurate integration of r^e (e.g., the same integration matrix used to approximate the integral of r^η) is needed. The first step from t_0 to t_1 does not use $(E^\eta + E^e)\partial_v e$ or $v\partial_x e$ because of the quantities that we initialize to zero. Since the error e is quite small, we safeguard the process of

1 Initialize:
2 $e|_{t=t_0} = 0, \quad E^{\eta^{new}}|_{t=t_0} = E^{\eta}|_{t=t_0}$,
3 (note $E^e|_{t=t_0} = 0$, $E^{\eta^{new}} = E^{\eta} + E^e$).
4 **for** $m = 0$ **do**
5 $(t_0$ to $t_1)$
6 solve $\partial_t e = -r^{\eta}$ for $e|_{t=t_1}$
7 Evaluate $E^{\eta^{new}}|_{t=t_1}$ using $(\eta + e)|_{t=t_1}$.
8 **for** $m = 1$ **to** $(M\text{-}1)$ **do**
9 $(t_m$ to $t_{m+1})$
10 solve $\partial_t e + E^{\eta^{new}}|_{t=t_m} \partial_v e = 0$.
11 then solve $\partial_t e + v\partial_x e = 0$.
12 then solve $\partial_t e = -r^{\eta} - r^e$ for $e|_{t=t_{m+1}}$,
13 using $E^e|_{t=t_m} = (E^{\eta^{new}} - E^{\eta})|_{t=t_m}$ in r^e.
14 Evaluate $E^{\eta^{new}}|_{t=t_{m+1}}$ using $(\eta + e)|_{t=t_{m+1}}$.
15 Update η as $\eta^{new} = \eta + e$.

Algorithm 4: One correction loop of IDC for VP using first order time splitting.

finding E^e by solving for the normalized $E^{\eta^{new}} = E^{\eta} + E^e$ and then obtaining $E^e = E^{\eta^{new}} - E^{\eta}$ (see lines 7 and 14 in the algorithm). Using the first order split prediction from Section 5.6.2 and J of the first order split corrections given in Algorithm 4, the split IDC method is $(J+1)$th order accurate in time.

The first order splitting in Algorithm 4 solving (5.112) could be generalized as solving the error equation found from (5.72)

$$e = f - \eta, \tag{5.113}$$

$$r = \partial_t \eta + A\partial_x \eta + B\partial_v \eta, \tag{5.114}$$

$$\partial_t e + A\partial_x e + B\partial_v e = -r, \qquad e(t_0) = 0. \tag{5.115}$$

A first order splitting could be given by:

- Take a full timestep $(t_0 \rightarrow t_0 + \delta t)$ to solve:

$$\partial_t \bar{e} = -r = -\partial_t \Big(\eta - f_0 + \int^t (A\partial_x \eta + B\partial_v \eta) d\tau \Big), \quad \text{IC} = e(t_0), \tag{5.116}$$

- Take a full timestep $(t_0 \rightarrow t_0 + \delta t)$ to solve:

$$\partial_t \bar{\bar{e}} + B\partial_v \bar{\bar{e}} = 0, \quad \text{IC} = \bar{e}, \tag{5.117}$$

- Take a full timestep $(t_0 \rightarrow t_0 + \delta t)$ to solve:

$$\partial_t e_{spl1} + A\partial_x e_{spl1} = 0, \quad \text{IC} = \bar{\bar{e}}. \tag{5.118}$$

Alternatively, a second order splitting of the error equation (5.115) could be given as [29]:

- Take a half timestep $(t_0 \rightarrow t_0 + \delta t/2)$ to solve:

$$\partial_t e_1 + A\partial_x e_1 = 0, \quad \text{IC} = e(t_0), \tag{5.119}$$

- Take a full $(t_0 \rightarrow t_0 + \delta t)$ timestep to solve:

  - Take a half timestep $(t_0 \rightarrow t_0 + \delta t/2)$ to solve:

$$\partial_t e_2 + B\partial_v e_2 = 0, \quad \text{IC} = e_1(t_0 + \delta t/2), \tag{5.120}$$

– Take a full $(t_0 \to t_0 + \delta t)$ timestep to solve:

$$\partial_t e_3 = -r, \quad \text{IC} = e_2(t_0 + \delta t/2), \tag{5.121}$$

– Take a half $(t_0 + \delta t/2 \to t_0 + \delta t)$ timestep to solve:

$$\partial_t e_4 + B\partial_v e_4 = 0, \quad \text{IC} = e_3(t_0 + \delta t), \tag{5.122}$$

- Take a half $(t_0 + \delta t/2 \to t_0 + \delta t)$ timestep to solve:

$$\partial_t e_{spl2} + A\partial_x e_{spl2} = 0, \quad \text{IC} = e_4(t_0 + \delta t). \tag{5.123}$$

5.7 Efficiency Analysis

The following sketch compares the number of split equations that must be solved by high order splitting methods constructed via IDC methods and those methods constructed via Yoshida's methods in [77]. Note that splitting methods in [39] scale similarly to methods in [77]. Although the metric we choose below is an imperfect metric for comparison, it suggests that there is merit in considering high order split IDC methods. In the comparison we assume the timestep Δt of Yoshida is the same as $\delta t = \Delta t/M$ (M = number of substeps in each IDC timestep) of an IDC method; i.e., for Yoshida, we let $\tau = \Delta t$, and for IDC methods, we let $\tau = \delta t$.

Consider a pth order method solving

$$\partial_t u = (A + B)u, \tag{5.124}$$

with exact solution $\exp(\tau(A + B))$, after one timestep. In keeping with Yoshida's notation, a splitting method's solution $S(\tau)$ can be written as a composition of

exponentials

$$S(\tau) = \prod_{i=1}^{k} \exp(c_i \tau A) \exp(d_i \tau B) \tag{5.125}$$

$$= \exp(\tau(A+B) + \mathcal{O}(\tau^{p+1})). \tag{5.126}$$

We introduce the idea of counting the number of split solves as a measure of the efficiency of the method, with the assumption that each split solve is roughly equivalent in cost (understanding that each split solve could be significantly different and thus nullifying our comparison). For example, a first order splitting could have $c_1 = 1$ and $d_1 = 1$, to give

$$S_1(\tau) = e^{\tau A} e^{\tau B}, \tag{5.127}$$

which would require two split solves. Similarly, a second order splitting (equivalent to Strang splitting) could be written with $c_1 = 1/2$, $d_1 = 1$, $c_2 = 1/2$, and $d_2 = 0$, to give

$$S_2(\tau) = e^{\frac{1}{2}\tau A} e^{\tau B} e^{\frac{1}{2}\tau A}, \tag{5.128}$$

which would require three split solves. For each of Yoshida's splitting methods that are higher than 2nd order, one must solve a set of equations (not shown here) to find the coefficients c_i, d_i. To find the exact coefficients, one must solve the coefficient equations analytically. In this situation, if the order of the method is p (and p is even), then the number of nonzero coefficients is $2k - 1$, where $k = 3^{p/2-1} + 1$. These coefficients will be referred to as the analytic coefficients. To find a simpler higher order splitting method, one may solve an approximate set of coefficient equations, giving fewer coefficients, but they are approximate and not exact coefficients.

These coefficients will be referred to as the nonanalytic coefficients. In practice, the nonanalytic coefficients are used since they make the computation much more cost effective (see Table 5.1). The number of nonanalytic coefficients is also given by $2k-1$, but k is different: $k = 2l+2$, where $l = 2^{p/2-1} - 1$, and the (even) order is $p > 4$. Some of these calculations can also be clarified by examining [77, 49, 50]. The number of nonzero coefficients (analytic or nonanalytic) can be thought of as the number of split solves required for one timestep of that method.

order	Yoshida analytic	Yoshida nonanalytic	IDC-S2	IDC-S1
2	3	-	3	3
4	7	-	$7\frac{1}{3}$	9
6	19	15	$12\frac{1}{5}$	15
8	55	31	$17\frac{1}{7}$	21

Table 5.1: Number of split solves required for one step τ of each splitting method. Yoshida's methods are from [77], and $\tau = \Delta t$. IDC-S2 refers to split IDC methods constructed with second order splitting, and IDC-S1 refers to split IDC methods constructed with first order splitting. $\tau = \delta t$ for IDC methods.

Unlike Yoshida's methods, it is not entirely correct to write split IDC methods as compositions of exponentials. However, we can still count the number of split solves in one timestep in a similar manner. Recall we are considering the solution of (5.124), so there will be no partial derivatives in the following analysis. One substep $\tau = \delta t$ of the split IDC prediction loop is simply a basic splitting method, which can be written as (5.127) or (5.128), for example. Thus the cost for one substep of the prediction loop is 2 solves for a first order splitting or 3 solves for a 2nd order prediction. One substep of the correction loop, solved according to the first order

splitting given by (5.116)-(5.118), can be written as

$$C_1(\tau) = e^{\tau A} e^{\tau B} R(\tau), \tag{5.129}$$

where $R(\tau)$ is the solution to (5.116). Thus the cost of this correction substep is 3 solves. However, if we look carefully at Algorithm 4, line 6, we see that the very first substep of an IDC interval, from $t = t_0$ to t_1, only requires the solution of (5.116), but (5.117) and (5.118) are unnecessary. Thus the first substep of the correction loop only requires 1 solve, while the last $M - 1$ substeps require the full 3 solves seen in (5.129).

Similarly, a second order splitting given by (5.119)-(5.123) can be expressed as

$$C_2(\tau) = e^{\frac{1}{2}\tau A} e^{\frac{1}{2}\tau B} R(\tau) e^{\frac{1}{2}\tau B} e^{\frac{1}{2}\tau A}, \tag{5.130}$$

where $R(\tau)$ is the solution to (5.121). Thus the cost of this correction substep is 5 solves. Again, since the initial error at each correction is zero, we may adjust the second order splitting to be more efficient (similarly to Algorithm 4, line 6). Thus we do not need to solve (5.119) or (5.120) over the first substep, but only

$$\widetilde{C}_2(\tau) = R(\tau) e^{\frac{1}{2}\tau B} e^{\frac{1}{2}\tau A}. \tag{5.131}$$

Therefore the first substep of the correction loop only requires 3 solves, while the last $M - 1$ substeps require the full 5 solves seen in (5.130).

To coalesce the number of solves required for an IDC prediction and the number required for a correction into the total number of solves required for one substep δt, we must first consider one step of size Δt. For a pth order IDC method, we subdivide Δt into $M = p - 1$ substeps (see (5.83), (5.86) in Section 5.6.1). We solve for the prediction solution over each of those M substeps. Then we solve for

the error correction J times over each of those M substeps. Letting P denote the number of solves required for a prediction substep, s_0 the number of solves required for the first substep of a correction, and s the number of solves required for the subsequent substeps of a correction, the resulting number of solves over Δt is

$$MP + J(s_0 + (M-1)s). \tag{5.132}$$

Then we divide by M to obtain the number of solves required for one substep $\tau = \delta t$,

$$solves = P + J(s_0 + (M-1)s)/M. \tag{5.133}$$

For example, if we consider a fourth order split IDC method constructed with first order splittings, then one must solve one prediction and 3 correction loops, requiring 9 solves per substep:

$$solves = 2 + 3(1 + 2 * 3)/3 = 9. \tag{5.134}$$

For a fourth order split IDC method constructed with second order splittings, one must solve one prediction and one correction loop, requiring $7\frac{1}{3}$ solves per substep:

$$solves = 3 + 1(3 + 2 * 5)/3 = 7\frac{1}{3}. \tag{5.135}$$

See the Table 5.1 for other higher order results comparing Yoshida's split methods and split IDC methods. For 2nd, 4th, and 6th order methods, Yoshida and IDC methods appear roughly equivalent, whereas for 8th order (and higher, though not shown), IDC methods appear to have an advantage in requiring fewer split solves.

5.8 Conservation of Mass

In this section, we prove that the split IDC methods constructed in this work preserve total mass in the case of periodic boundary conditions; i.e., for f_{ij}^{n+1} the numerical solution to the Vlasov-Poisson system at $t = t_{n+1}$, $x = x_i = x_0 + i\Delta x$, $v = v_j = v_0 + j\Delta v$, we desire the discrete version of

$$\int_X \int_V f(x, v, t_{n+1}) dx\, dv = \int_X \int_V f(x, v, t_n) dx\, dv \tag{5.136}$$

to hold:

$$\sum_{j=0}^{N_v-1} \sum_{i=0}^{N_x-1} f_{ij}^{n+1} \Delta x \Delta v = \sum_{j=0}^{N_v-1} \sum_{i=0}^{N_x-1} f_{ij}^{n} \Delta x \Delta v. \tag{5.137}$$

For simplicity, below we drop $\Delta x \Delta v$ and the limits on the sums.

Theorem 5.8.1. *A split IDC method that is constructed via first order splitting methods in the prediction and correction loops together with conservative semi-Lagrangian WENO methods, and whose spatial and velocity derivatives in the residual are approximated via WENO reconstruction, conserves total mass if periodic boundary conditions are imposed.*

Proof. By Proposition 5.8.3 and repeated application (according to the number of correction loops) of Proposition 5.8.5, both given below. □

Proposition 5.8.2.

$$f_i^{n+1} = f_i^n - \xi_0(\hat{f}_{i+1/2}^n(\xi_0) - \hat{f}_{i-1/2}^n(\xi_0)) \tag{5.138}$$

conserves total mass if periodic boundary conditions are imposed and $\hat{f}_{i-1/2}^n$ *is the numerical flux given by a* $2k+1$ *order conservative semi-Lagrangian WENO*

reconstruction. For example, when interpolation and reconstruction is from the left, the flux is defined as

$$\hat{f}^n_{i-1/2} = (f^n_{i-k-1}, \ldots, f^n_{i+k-1}) \cdot C^L_{2k+1} \cdot (1, \xi, \ldots, \xi^{2k}), \tag{5.139}$$

for C^L_{2k+1} *a matrix of coefficients and* $\xi \in [0, 1/2]$. *Further details are in [67].*

Proof. The sums are over $i = 0, 1, \ldots, N-1$ ($i = N$ is excluded due to periodicity).

$$\begin{aligned}
\sum_i f^{n+1}_i &= \sum_i (f^n_i - \xi_0(\hat{f}^n_{i+1/2}(\xi_0) - \hat{f}^n_{i-1/2}(\xi_0))) \\
&= \sum_i f^n_i - \xi_0 \sum_i (\hat{f}^n_{i+1/2}(\xi_0) - \hat{f}^n_{i-1/2}(\xi_0)) \\
&= \sum_i f^n_i - \xi_0(-\hat{f}^n_{-1/2}(\xi_0) + \hat{f}^n_{N-1/2}(\xi_0)) \\
&= \sum_i f^n_i,
\end{aligned}$$

where the last equality holds from periodicity, which can be seen by writing out the fluxes as defined in the proposition. □

Proposition 5.8.2 represents the solution of only one of the split equations in (5.94). However, conservation also carries through for consecutive splittings.

Proposition 5.8.3. *The first order split prediction of a split IDC method solved via conservative semi-Lagrangian WENO conserves total mass if periodic boundary conditions are imposed.*

Proof. Let f^*_{ij} denote the solution updating f^n_{ij} in the x direction, given by

$$f^*_{ij} = f^n_{ij} - \xi_0(\hat{f}^n_{i+1/2,j}(\xi_0) - \hat{f}^n_{i-1/2,j}(\xi_0)). \tag{5.140}$$

We know by Proposition 5.8.2 that

$$\sum_i f_{ij}^* = \sum_i f_{ij}^n \tag{5.141}$$

for all j. Then let f_{ij}^{n+1} denote the solution updating f_{ij}^* in the v direction, given by

$$f_{ij}^{n+1} = f_{ij}^* - \xi_0(\hat{f}_{i,j+1/2}^*(\xi_0) - \hat{f}_{i,j-1/2}^*(\xi_0)). \tag{5.142}$$

Again, by Proposition 5.8.2, we have

$$\sum_j f_{ij}^{n+1} = \sum_i f_{ij}^* \tag{5.143}$$

for all i. Therefore

$$\sum_j \sum_i f_{ij}^{n+1} = \sum_i \left(\sum_j f_{ij}^{n+1} \right) = \sum_i \left(\sum_j f_{ij}^* \right) \tag{5.144}$$

$$= \sum_j \left(\sum_i f_{ij}^* \right) = \sum_j \left(\sum_i f_{ij}^n \right). \tag{5.145}$$

□

Lemma 5.8.4. *If the derivatives are found via WENO reconstruction, then*

$$\sum_i \sum_j v_j \partial_x \eta_{ij} + E_i^{\eta+e} \partial_v \eta_{ij} = 0. \tag{5.146}$$

Proof. From [67], we see that the WENO flux coefficients can be written as a vector, which for example, when using a $2k+1$th order reconstruction from the left, we

denote as w^L_{2k+1}. Then the numerical flux $\hat{F}^n_{i-1/2}$ can be written

$$\hat{F}^n_{i-1/2} = (f^n_{i-k-1}, \cdots, f^n_{i+k-1}) \cdot w^L_{2k+1}, \tag{5.147}$$

and a derivative can be approximated as

$$\partial_x f^n_i \approx -\frac{1}{\Delta x}(\hat{F}^n_{i+1/2} - \hat{F}^n_{i-1/2}). \tag{5.148}$$

Now consider the sum in the lemma, with the derivatives approximated as in (5.148).

$$\begin{aligned}
&\sum_i \sum_j v_j \partial_x \eta_{ij} + E^{\eta+e}_i \partial_v \eta_{ij} \\
&= \sum_j v_j \sum_i -\frac{1}{\Delta x}(\hat{F}^n_{i+1/2,j} - \hat{F}^n_{i-1/2,j}) \\
&\quad + \sum_i E^{\eta+e}_i \sum_j -\frac{1}{\Delta v}(\hat{F}^n_{i,j+1/2} - \hat{F}^n_{i,j-1/2}) \\
&= -\sum_j \frac{v_j}{\Delta x}(-\hat{F}^n_{-1/2,j} + \hat{F}^n_{N_x-1/2,j}) \\
&\quad - \sum_i \frac{E^{\eta+e}_i}{\Delta v}(-\hat{F}^n_{i,-1/2} + \hat{F}^n_{i,N_v-1/2}) \\
&= 0,
\end{aligned}$$

where the last inequality holds by periodicity, which can be seen by writing out the fluxes as defined above. □

Proposition 5.8.5. *The first order split correction of a split IDC method solved via conservative semi-Lagrangian WENO methods (and whose spatial and velocity derivatives in the residual are approximated via WENO reconstruction) conserves total mass if the prediction conserves total mass and periodic boundary conditions are imposed.*

Proof. Since (5.117) and (5.118) are solved in the same way as the prediction, we only need to prove that the numerical solution for (5.116) conserves mass. Denoting e^{n+1} as the error at the updated time t_{n+1}, we have, as the solution to (5.116),

$$e_{ij}^{n+1} = e_{ij}^{n} - \eta_{ij}^{n+1} + \eta_{ij}^{n} - \int_{t_n}^{t_{n+1}} v_j \partial_x \eta_{ij}(\tau) + E_i^{\eta+e} \partial_v \eta_{ij}(\tau)\, d\tau.$$

Then

$$\begin{aligned}
\sum_i \sum_j e_{ij}^{n+1} &= \sum_i \sum_j e_{ij}^{n} - \sum_i \sum_j \eta_{ij}^{n+1} + \sum_i \sum_j \eta_{ij}^{n} \\
&\quad - \sum_i \sum_j \int_{t_n}^{t_{n+1}} v_j \partial_x \eta_{ij}(\tau) + E_i^{\eta+e} \partial_v \eta_{ij}(\tau)\, d\tau \\
&= \sum_i \sum_j e_{ij}^{n} - \int_{t_n}^{t_{n+1}} \sum_i \sum_j v_j \partial_x \eta_{ij}(\tau) + E_i^{\eta+e} \partial_v \eta_{ij}(\tau)\, d\tau \\
&= \sum_i \sum_j e_{ij}^{n},
\end{aligned}$$

where the second equality holds by the hypothesis, and the last equality holds by Lemma 5.8.4. □

Although the results in this section are for split IDC methods constructed via first order splittings, we anticipate that conservation of mass for split IDC methods constructed via second order splittings can be similarly shown.

5.9 Numerical Results

We apply the split IDC methods to constant advection, rotating problems, and Vlasov-Poisson problems, where we study classic plasma problems, such as the warm two stream instability and Landau damping.

Although some of the following notation used may be obvious, we provide it

here for complete clarity: Δt is the timestep as given by (5.84), N_x, N_v are the number of space and velocity (or second space, where relevant) steps. T_f is the final time to which the solution was calculated. When explaining which part of the operator is used in the splitting, we use the notation of A, B as in Section 5.5. For spatial differentiations and interpolations, we use WENO reconstruction [19] and conservative semi-Lagrangian WENO [67]. For error plots, the error is given as the sum of the l_1 and l_∞ norms ($l_1 + l_\infty$). Comparison to exact, reference, or successive solutions is clarified for each test problem.

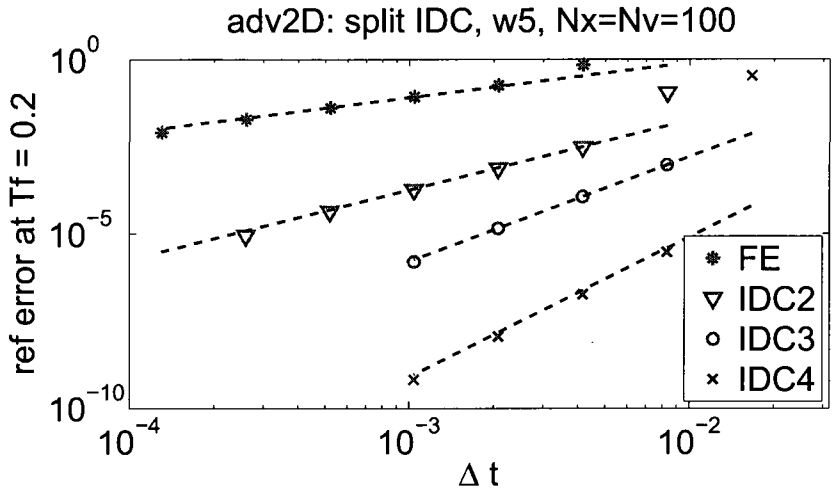
5.9.1 Constant Advection

Here we apply split IDC methods to a constant advection equation,

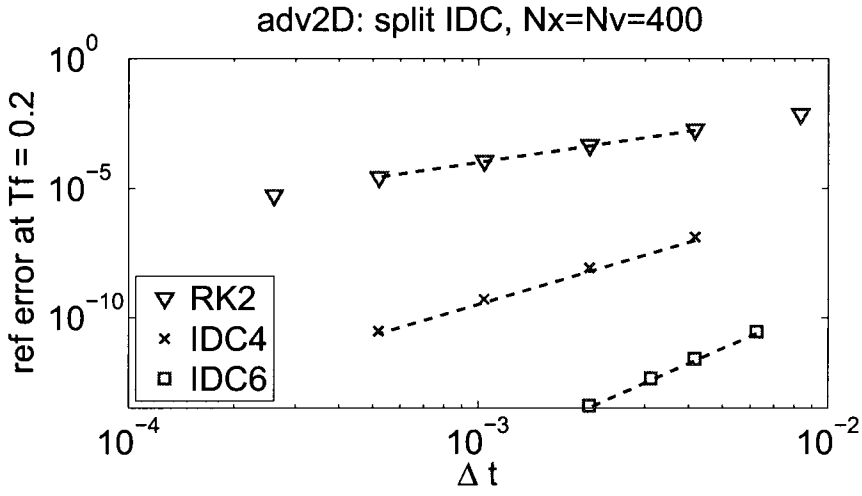
$$\begin{aligned} u_t + u_x + u_v = 0, \quad (x,v) \in [-1,1] \times [-1,1], \quad t > 0, \\ u(0,x,v) = \sin(4\pi(x+v)), \\ u(t,1,v) = u(t,-1,v), \quad u(t,x,1) = u(t,x,-1), \end{aligned} \tag{5.149}$$

using simple numerical integrators in each split step rather than exact (semi-Lagrangian) solves. We verify that the split IDC framework attains the expected increase in order of accuracy with each correction loop. Here we use $A = B = 1$. For the spatial derivatives, we use fifth or ninth order WENO reconstruction.

In Figure 5.1a, we show error plots from solving (5.149) via split IDC that incorporates first order splitting where each split step is solved with a forward Euler (FE) integration. In Figure 5.1b, we show error plots from solving (5.149) via split IDC that incorporates second order splitting where each split step is solved with a second order Runge-Kutta integration. Clearly the order of accuracy increases and the error decreases with each successive correction loop as expected.



(a) 1st order split (0, 1, 2, 3 corrections)



(b) 2nd order split (0, 1, 2 corrections)

Figure 5.1: Convergence study for (5.149) using (5.1a) IDC constructed with 1st order split methods, each split equation is solved via forward Euler, combined with 5th order WENO. The order of accuracy is clear, as reference lines (with slopes of 1, 2, 3, 4) indicate. (5.1b) IDC constructed with 2nd order split methods, each split equation is solved via 2nd order Runge-Kutta, combined with 9th order WENO. The order of accuracy is clear, as reference lines (with slopes of 2, 4, 6) indicate. In both figures, the error is in the norm $l_1 + l_\infty$, compared to a reference solution computed on a finer time mesh.

Although we also used semi-Lagrangian split IDC methods with the conservative WENO construction, the plots are not shown here because the error obtained was effectively machine precision. There is no splitting error for (5.149) since the operators commute, so those results are not unexpected. However, since there is no splitting error, the test of constant advection is not sufficient to verify split IDC methods' increase in order of accuracy as the number of correction loops increases. Thus we consider the rotating problem next.

5.9.2 Rotation

Using split IDC methods with conservative semi-Lagrangian 5th order WENO, we solved the problem of rotating a nonsymmetric Gaussian initial condition

$$\begin{aligned} u_t - vu_x + xu_v = 0, \quad (x,v) \in [-1,1] \times [-1,1], \quad t > 0, \\ u(0,x,v) = \exp\left(-20(x^2 + 5v^2)\right), \\ u(t,1,v) = u(t,-1,v), \quad u(t,x,1) = u(t,x,-1). \end{aligned} \tag{5.150}$$

Here we used $A = -v$, $B = x$ for our splittings.

Time convergence results can be seen in Figure 5.2a for solving (5.150) via split IDC that incorporates a first order splitting in prediction and correction loops. We see that the temporal order of accuracy increases by $\mathcal{O}(\Delta t)$ with each additional correction. Although the figure shows results when $N_x = N_v = 40$, which may raise questions since the Gaussian is sharp enough that more spatial resolution is needed to capture its peak, we found identical results for $N_x = N_v = 400$. Figure 5.2b shows convergence for split IDC that incorporates second order splitting in prediction and correction loops with $N_x = N_v = 400$. A prediction loop only (or simply a standard implementation of Strang splitting) gives second order. For a prediction and one

correction loop, an increase in accuracy by $\mathcal{O}(\Delta t^2)$ is expected, and indeed clean fourth order is acheived. For a prediction and two correction loops, sixth order is expected, but it is hard to verify due to acheiving Matlab precision quickly. However, the magnitude of its error is less than the fourth order error.

Since the split operators used here for (5.150) do not commute, the splitting error will contribute to the error that the IDC methods must correct, and in light of our results, we conclude that IDC methods are able to correct splitting errors to achieve higher order split methods. Unfortunately, however, the IDC corrections appear to introduce a CFL that is not present in the prediction loop. The reason remains to be discovered. Now we move on to more interesting examples.

5.9.3 Drifting Rotation

In this section, we consider a problem that is similar to the rotating problem, except there is a drift in time. We refer to the problem as a drifting rotating problem, but it can also be considered a linear Vlasov equation [69]:

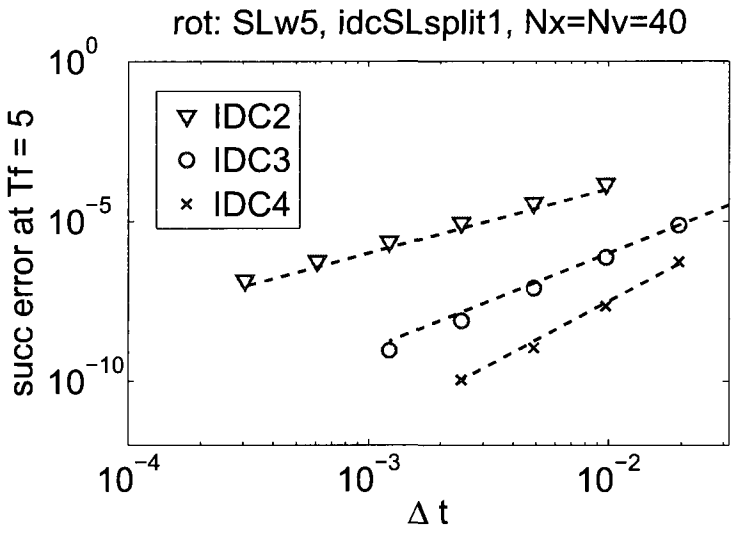
$$\partial_t f + v \cdot \nabla_x f + E(t,x) \cdot \nabla_v f = 0, \quad f(0,x,v) = f_0(x,v), \tag{5.151}$$

where $E(t,x) : [0,\infty) \times \mathcal{R}^N \to \mathcal{R}^N$ is given and smooth. As in [69], we consider x and $v \in \mathcal{R}$ with

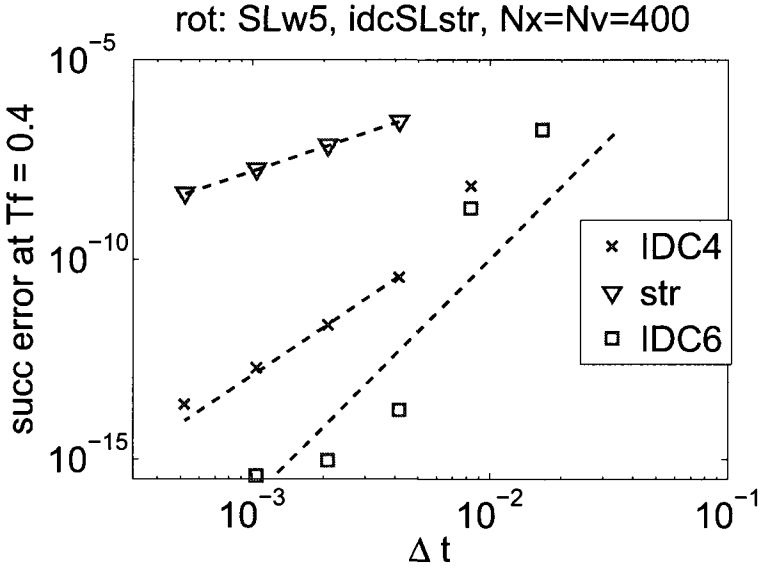
$$E(t,x) = -x - \sin(2t),$$

$\mathcal{F} : \mathcal{R} \to \mathcal{R}$ defined by $\mathcal{F}(0) = 0$ and

$$\mathcal{F}'(s) = \begin{cases} \frac{35\pi}{32}\sin^7(\pi s) & \text{if } 0 < s < 1 \\ 0 & \text{otherwise} \end{cases},$$



(a) 1st order split (1, 2, 3 corrections)



(b) 2nd order split (0, 1, 2 corrections)

Figure 5.2: Convergence study for (5.150). (5.2a) IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO. The order of accuracy is clear, as reference lines (with slopes of 2, 3, 4) indicate. (5.2b) IDC constructed with 2nd order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO. The order of accuracy is clear for 2nd and 4th order, but unclear for 6th order, as reference lines (with slopes of 2, 4, 6) indicate. In both figures, the error is in the norm $l_1 + l_\infty$, comparing successive solutions.

so that if

$$F(0,x,v) = \mathcal{F}'(x\cos 1 + v\sin 1 - 2\sin 1 + \sin 2) \cdot \mathcal{F}'(-x\sin 1 + v\cos 1 - 2\cos 1 + 2\cos 2),$$

then the exact solution [68] is

$$f(t,x,v) = \mathcal{F}'((x-\sin(2t))\cos(t-1) - (v-2\cos(2t))\sin(t-1) + \sin 2) \cdot \mathcal{F}'((x-\sin(2t))\sin(t-1) - (v-2\cos(2t))\cos(t-1) + 2\cos 2),$$

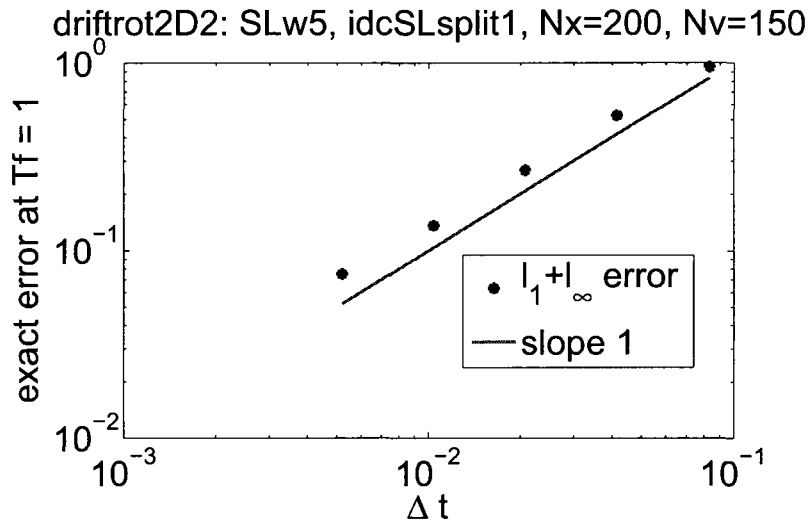
which for $t = 1$ gives

$$f(1,x,v) = \mathcal{F}'(x)\mathcal{F}'(v).$$

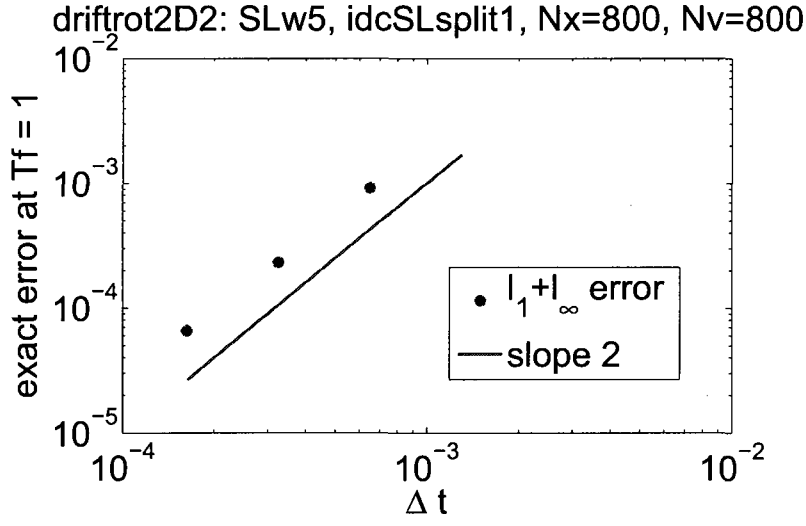
Here we use $A = v$, $B = E$ for our splittings.

Our convergence results can be seen in Figures 5.3a, 5.3b, 5.4a, and 5.4b, where the error is found by comparing the numerical and exact solutions in Figures 5.3a and 5.3b and successive solutions in Figures 5.4a and 5.4b. We see that split IDC that incorporates a first order splitting in prediction and correction loops has the expected convergence, where the order of accuracy increases by $\mathcal{O}(\Delta t)$ with each additional correction loop.

As with the solution of the rotating problem, we observed a CFL limitation for split IDC methods. Thus a future step for us is to determine error bounds that clarified the CFL condition for split IDC methods. However, other higher order split methods also exhibit a CFL limitation. For example, Schaeffer tested the second order splitting method from [13] and his own fourth order splitting method [69] combined with a third order interpolation scheme on (5.151). He tested the

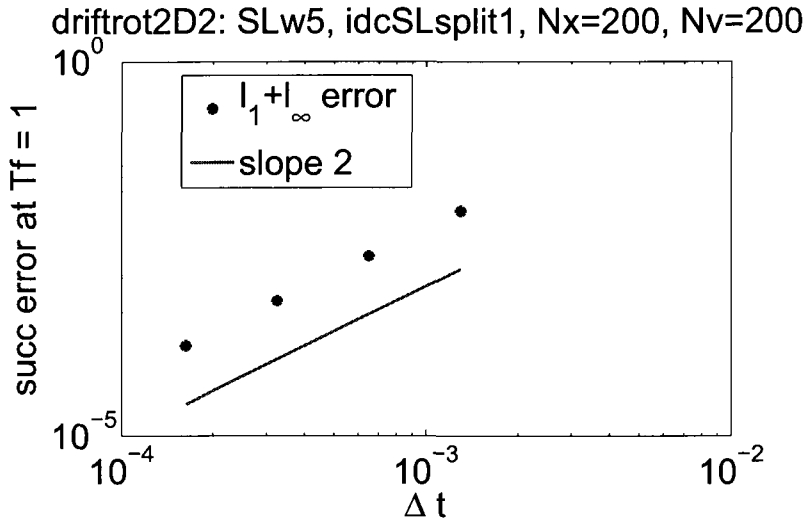


(a) 1 prediction, 0 corrections

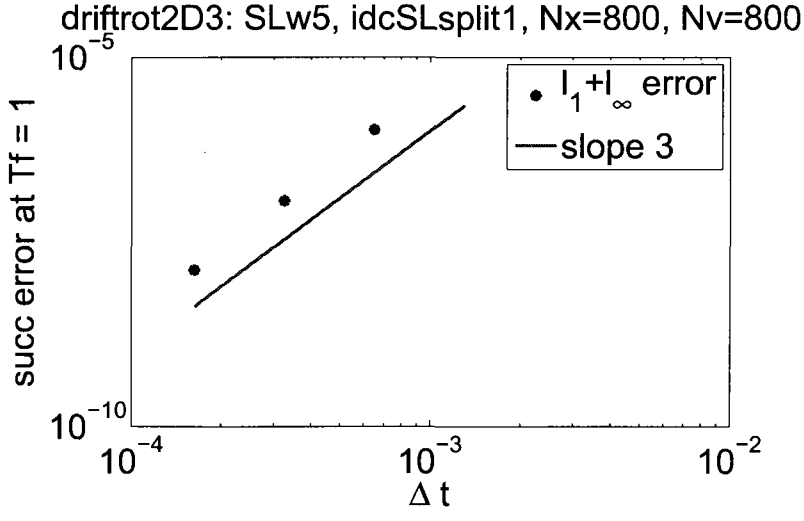


(b) 1 prediction, 1 correction

Figure 5.3: The order of accuracy is clear, as reference lines (with slopes of 1, 2) indicate. In all figures, the error is in the norm $l_1 + l_\infty$.



(a) 1 prediction, 1 correction



(b) 1 prediction, 2 corrections

Figure 5.4: The order of accuracy is clear, as reference lines (with slopes of 2, 3) indicate. In all figures, the error is in the norm $l_1 + l_\infty$.

methods numerically with $\Delta x = \Delta v$ and $\Delta t = c(\Delta x)^{\frac{4}{5}}$ for a fixed constant c for his method and $\Delta t = c(\Delta x)^{\frac{4}{3}}$ for a (possibly different) fixed constant c for [13]'s method. The values of Δt were chosen based on a CFL condition found through a rigorous error bound, and so that the error bounds in each case should be optimized [69]. Although Schaeffer's fourth order method showed an improvement over Cheng and Knorr's method, we note that Schaeffer's method was only implemented for a linear problem. He displays no results for the nonlinear Vlasov system, and it is not clear how his method extends to the nonlinear problem. In the following section, we show that we may apply our split IDC methods to the nonlinear Vlasov-Poisson system.

5.9.4 Vlasov–Poisson

Now we consider the implications of using split IDC methods with conservative semi-Lagrangian WENO to solve the Vlasov-Poisson system (5.88) with initial conditions of a two stream instability and Landau damping. In addition to temporal convergence results, we also include plots of solutions and physical quantities that should be conserved theoretically. Since the spatial resolution and methods are largely what influence the accurate numerical representation of the physical quantities, we do not expect that high temporal order via split IDC methods will improve them; however, we hope to assert that the physical properties are not unduly degraded by the split IDC methods' error correction process.

For both the two stream instability and Landau damping, we impose periodic boundary conditions in the x-direction and Neumann boundary conditions in the v-direction. Periodicity in space allows the use of a fast Fourier transform (FFT) to solve the 1D Poisson equation. The density $\rho(x,t)$ is computed by the rectangular rule, $\rho(x,t) = \int f(x,v,t)dv \approx \sum_{j=1}^{N_v} f(x,v_j,t)\delta v$ and then is normalized as

described in Section 5.6.3. For the splittings, we use $A = v$, $B = E$. Convergence studies are performed using $N_x = N_v = 400$. Classical theoretical results for the Vlasov-Poisson system include conservation of the following quantities (i.e., the time derivative is zero):

- The L^p norm, for $1 \leq p < \infty$,

$$\int_V \int_X |f(x,v,t)|^p dx\, dv, \tag{5.152}$$

- Entropy,

$$-\int_V \int_X f(x,v,t) \ln(f(x,v,t)) dx\, dv, \tag{5.153}$$

- Total energy,

$$\frac{1}{2}\int_V \int_X f(x,v,t) v^2 dx\, dv + \frac{1}{2}\int_X E(x,t)^2 dx. \tag{5.154}$$

In our numerical experiments, we checked the time evolution of the discrete versions of these theoretically preserved quantities. Entropy, energy, and the L^1 and L^2 norms are approximated by the rectangular rule.

Two Stream Instability

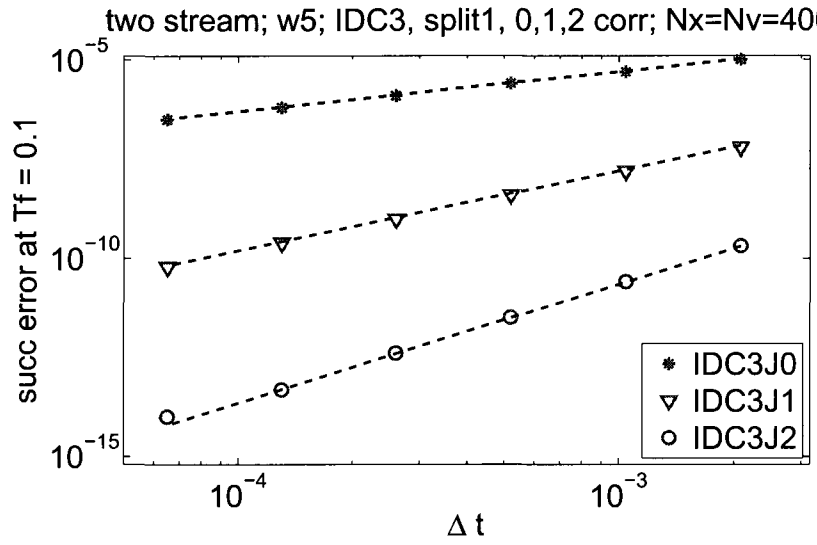
For the two stream instability, we use (5.88) with the initial condition [26]

$$f(0,x,v) = \frac{2}{7\sqrt{\pi}}(1+5v^2)\,(1+\alpha((\cos(2kx)+\cos(3kx))/1.2+\cos(kx))) \cdot \exp(-v^2/2), \tag{5.155}$$

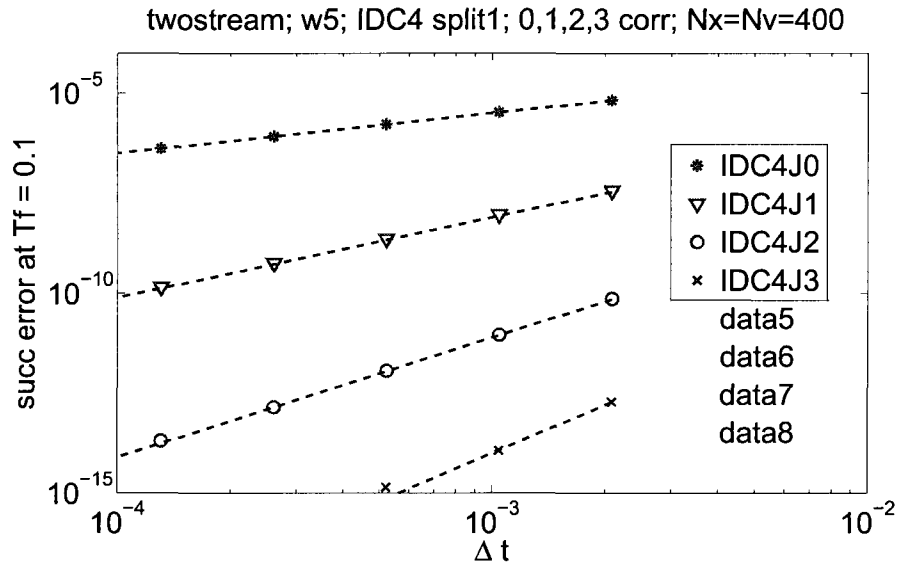
with $\alpha = 0.1$, $k = 0.5$, $x \in [0, 2\pi/k]$, $v \in [-\pi/k, \pi/k]$, periodic BCs in the x direction, and Neumann BCs in the v direction. Figures 5.5a and 5.5b show time convergence for split IDC methods with three and four, respectively, nodes in each subinterval, and in both figures, the error is shown for the prediction loop alone, followed by the prediction plus an increasing number of correction loops. As with the constant advection, rotation, and drifting rotation examples, we see a decreased error and very clear improved order of accuracy with each additional IDC correction loop.

A first order split solution and a fourth order split IDC solution are shown in Figures 5.6a and 5.6b, respectively. In both plots, $N_x = N_v = 400$, $\Delta t = 1/300$, and the final time is $T_f = 25$.

The physical quantities are shown in Figures 5.7a, 5.7b, 5.7c, 5.7d, and 5.7e. In all plots, $N_x = 64$, $N_v = 128$, $\Delta t = 1/40$, and $M + 1 = 4$. The energy, entropy, and L^2 norms are virtually indistinguishable among the numerical methods, so split IDC methods conserve these quantities equally as well as the first order splitting method. Although the L_1 norm (Figure 5.7a) has one peak for both the first order splitting (lower peak) and all the split IDC methods shown (higher peak), note that the peak is very small ($< \mathcal{O}(10^{-5})$, though the figure shows only $\leq \mathcal{O}(10^{-4})$). We note that it should not be surprising that split IDC methods do not preserve positivity because the structure of IDC methods closely resembles (and when RK methods are used in the IDC construction, actually is identical to, see [17]) the structure of RK methods, which are known to be non-positivity preserving for orders greater than one without special assumptions on the type of problem solved or the size of the timestep used [7, 61]. The integral $\int \int f(x, v, t) dx\, dv$ is clearly conserved (Figure 5.7b), so these split IDC methods maintain conservation of mass, consistent with the proof in Section 5.8.

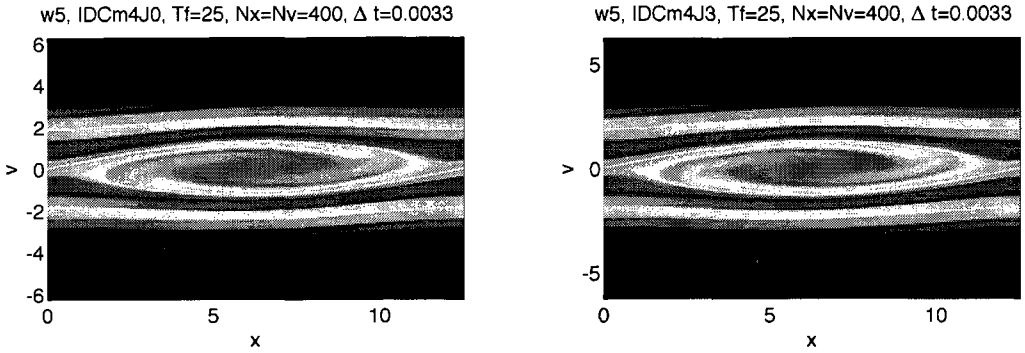


(a) 3 IDC nodes



(b) 4 IDC nodes

Figure 5.5: Convergence study for (5.88) with ICs as in (5.155). IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO. (5.5a) Here three IDC nodes are used, with 0, 1, and 2 correction loops. The order of accuracy is clear, as reference lines (with slopes of 1, 2, 3) indicate. (5.5b) Here four IDC nodes are used, with 0, 1, 2, and 3 correction loops. The order of accuracy is clear, as reference lines (with slopes of 1, 2, 3, 4) indicate. In both figures, the error is in the norm $l_1 + l_\infty$, comparing successive solutions.



(a) 1st order

(b) 4th order

Figure 5.6: Solution for (5.88) with ICs as in (5.155). IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = N_v = 400$, $\Delta t = 1/300$.

Landau Damping

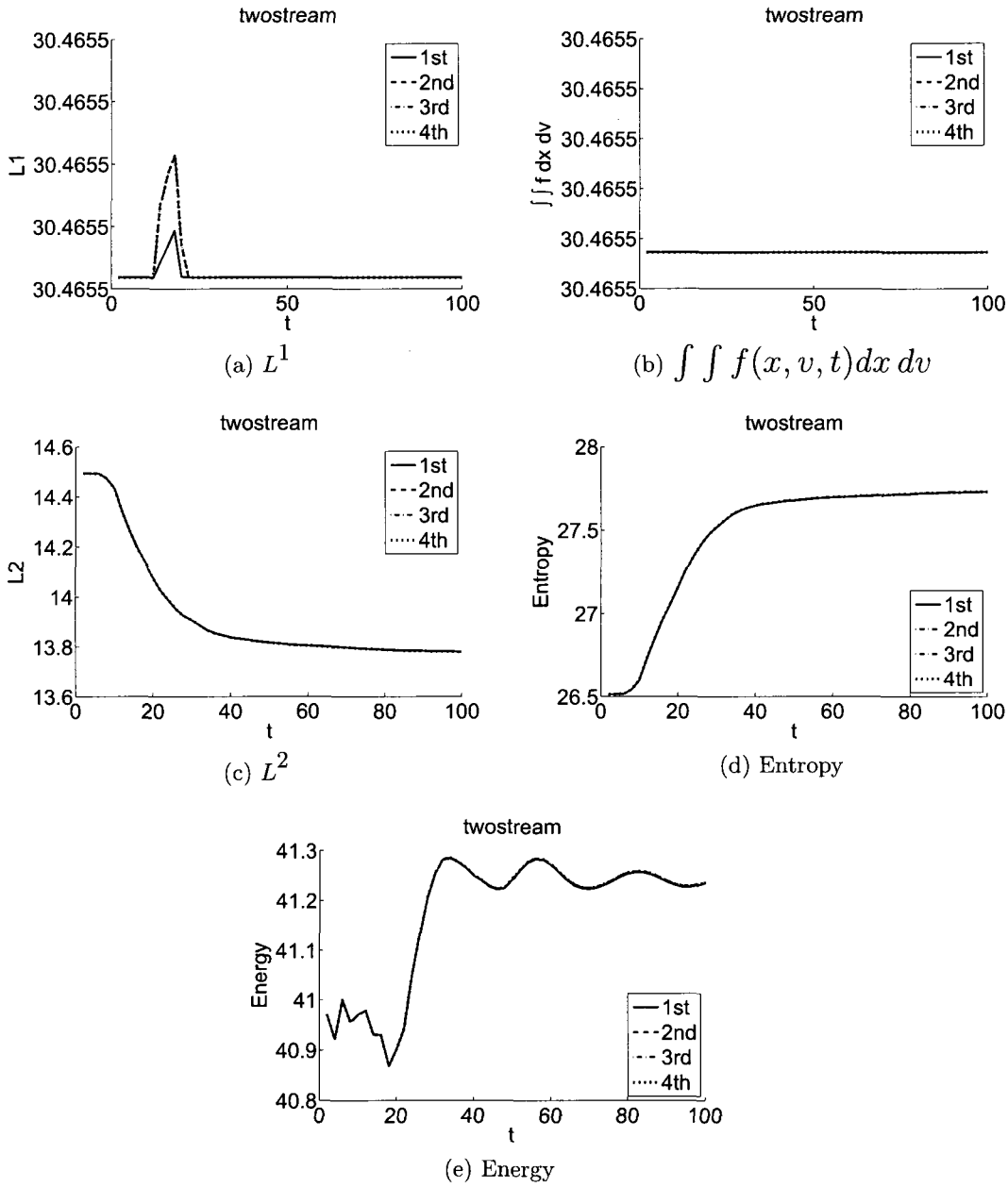
Landau damping is given by (5.88) with the initial condition [26]

$$f(0, x, v) = (1 + \alpha \cos(kx)) \exp(-0.5v^2)/\sqrt{2\pi}, \tag{5.156}$$

with $k = 1$, $x \in [0, 2\pi]$, $v \in [-2\pi, 2\pi]$, periodic BCs in the x direction, and Neumann BCs in the v direction, $\alpha = 0.5$ for strong Landau damping, and $\alpha = 0.01$ for weak Landau damping.

Figures 5.8a and 5.8b show time convergence for split IDC methods applied to weak and strong Landau damping problems, respectively, and in both figures, the error is shown for the prediction loop alone, followed by the prediction plus an increasing number of correction loops. As with the previous examples, we see a decreased error and very clear improved order of accuracy with each additional IDC correction loop.

For weak damping, a first order split solution and a fourth order split IDC solution are shown in Figures 5.9a and 5.9b, respectively. In both plots, $N_x =$



(a) L^1

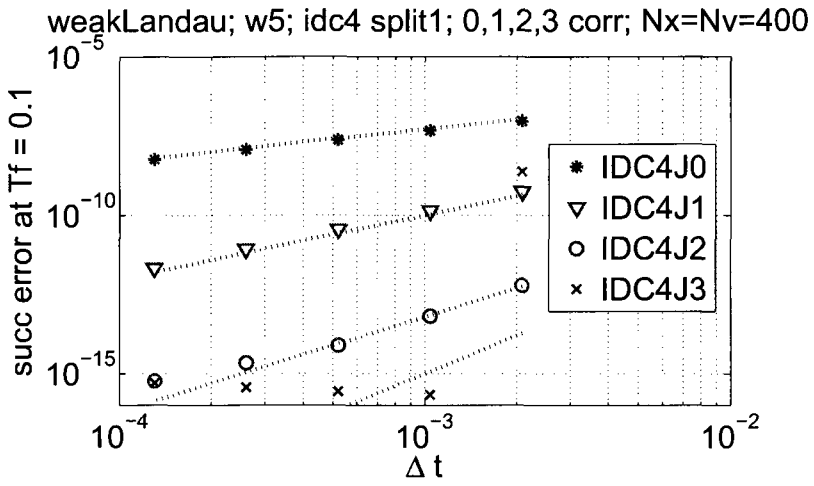
(b) $\int \int f(x, v, t) dx\, dv$

(c) L^2

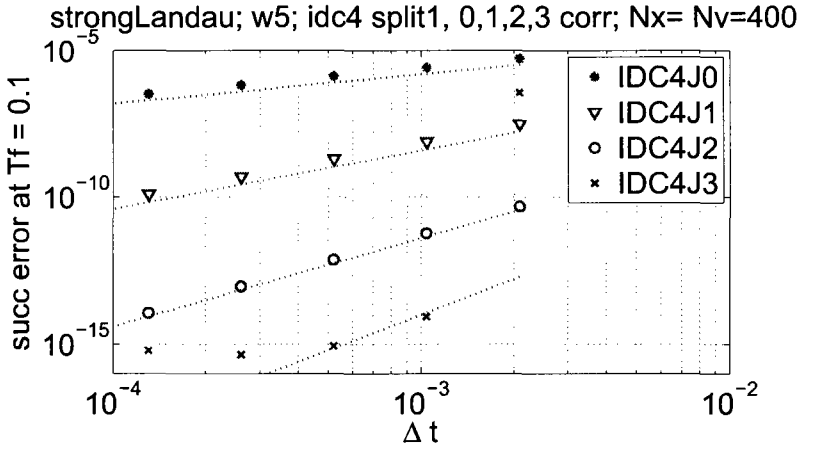
(d) Entropy

(e) Energy

Figure 5.7: Physical quantities for (5.88) with ICs as in (5.155). IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = 64$, $N_v = 128$, $\Delta t = 1/40$, and $M + 1 = 4$. The labels 1st, 2nd, 3rd, and 4th in the legends denote first order splitting (prediction only), 2nd order split IDC (prediction, 1 correction), 3rd order split IDC (prediction, 2 corrections), and 4th order split IDC (prediction, 3 corrections) methods, resp.

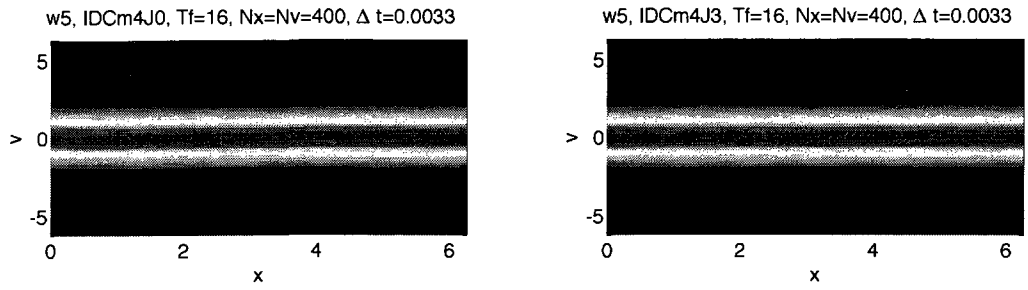


(a) weak Landau



(b) strong Landau

Figure 5.8: Convergence study for (5.88) with ICs as in (5.156). In Figure 5.8a, $\alpha = 0.01$, and in Figure 5.8b, $\alpha = 0.5$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO. Here four IDC nodes are used, with 0, 1, 2, and 3 correction loops. The order of accuracy is clear, as reference lines (with slopes of 1, 2, 3, 4) indicate. The error is in the norm $l_1 + l_\infty$, comparing successive solutions.



(a) 1st order (b) 4th order

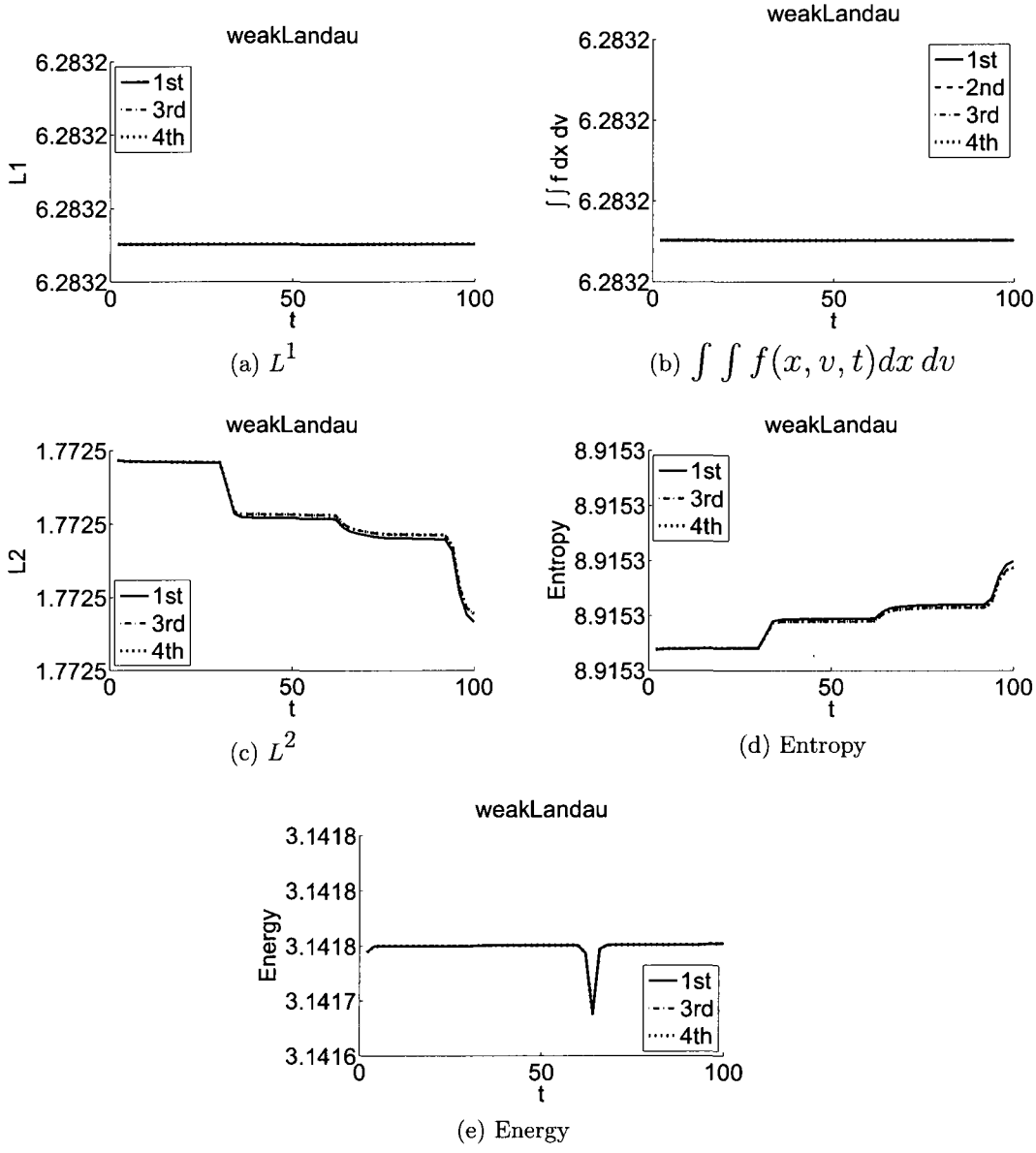
Figure 5.9: Solution for (5.88) with ICs as in (5.156), $\alpha = 0.01$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = N_v = 400$. (5.9a) $\Delta t = 1/300$. (5.9b) $\Delta t = 1/300$.

$N_v = 400$ and $\Delta t = 1/300$, and the final time is $T_f = 16$.

The physical quantities for weak damping are shown in Figures 5.10a, 5.10c, 5.10d, and 5.10e. We can see that the quantities are essentially conserved for all methods shown, and IDC methods may have a slight improvement over the first order splitting alone. Second order split IDC methods were not included in most graphs since there appear to be stability issues due to the choice of time step. The expected damping of the electric field is seen qualitatively in Figure 5.11.

For strong Landau damping, a first order split solution and a fourth order split IDC solution are shown in Figures 5.12a ($\Delta t = 1/80$) and 5.12b ($\Delta t = 1/80$), respectively. In both plots, $N_x = N_v = 400$, and the final time is $T_f = 16$.

The physical quantities for strong damping are shown in Figures 5.13a, 5.13c, 5.13d, and 5.13e. We can see that the quantities are essentially conserved for all methods shown, and IDC methods produce nearly identical results to first order splitting alone. Second order split IDC methods were not included in most graphs since there appear to be stability issues due to the choice of time step. The expected damping of the electric field is seen here qualitatively in Figures 5.14.



(a) L^1

(b) $\int \int f(x, v, t) dx\, dv$

(c) L^2

(d) Entropy

(e) Energy

Figure 5.10: Physical quantities for (5.88) with ICs as in (5.156), $\alpha = 0.01$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = 64$, $N_v = 128$, and $M + 1 = 4$. The labels 1st, 2nd, 3rd, and 4th in the legends denote first order splitting (prediction only, $\Delta t = 1/40$), 2nd order split IDC (prediction, 1 correction, $\Delta t = 1/40$), 3rd order split IDC (prediction, 2 corrections, $\Delta t = 1/80$), and 4th order split IDC (prediction, 3 corrections, $\Delta t = 1/80$) methods, resp.

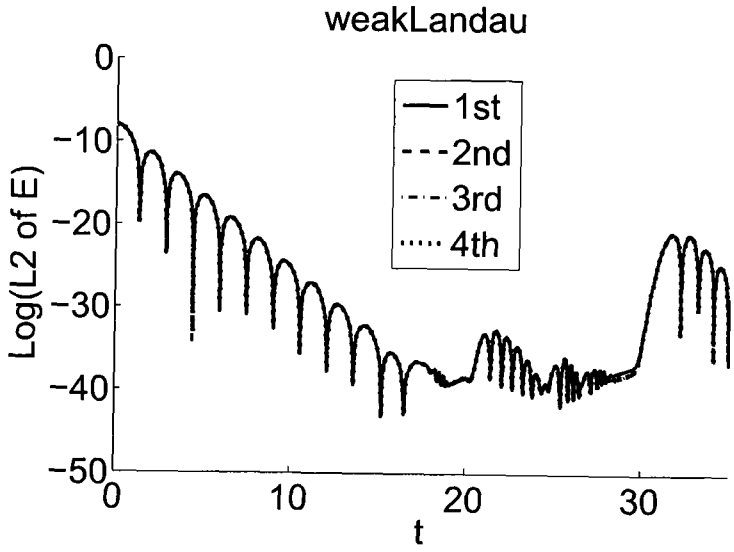
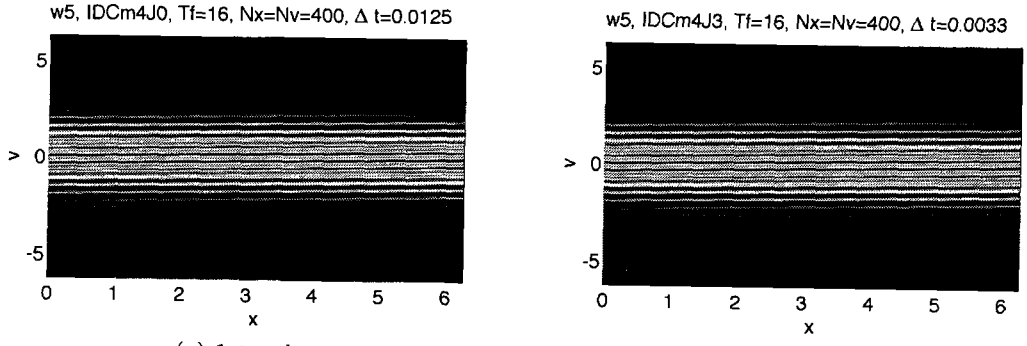


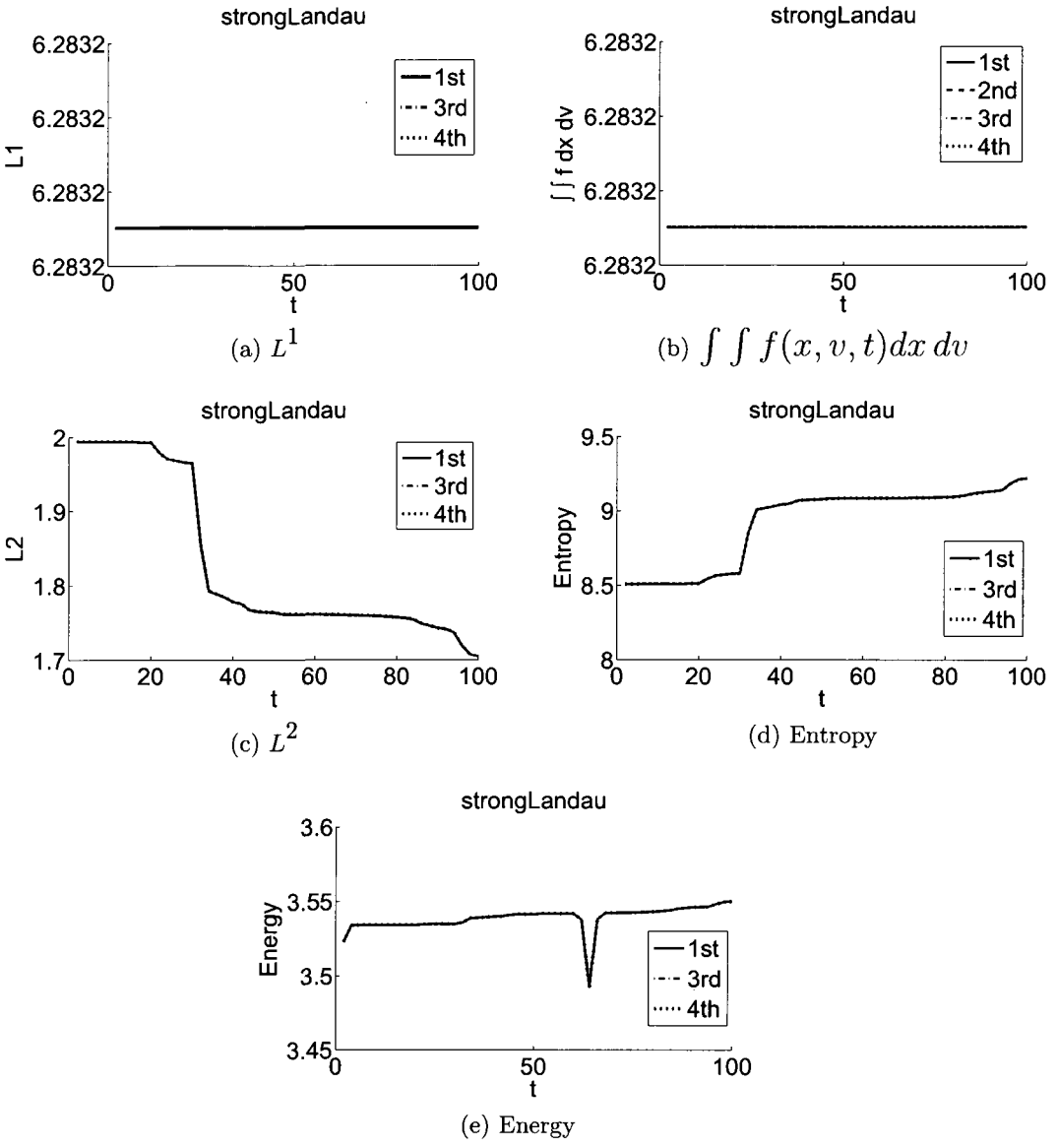
Figure 5.11: Electric field for (5.88) with ICs as in (5.156) and $\alpha = 0.01$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = 64$, $N_v = 128$, $M + 1 = 4$. For 1st order: $\Delta t = 1/80$, 2nd order: $\Delta t = 1/80$, 3rd order: $\Delta t = 1/80$, 4th order: $\Delta t = 1/80$.



(a) 1st order

(b) 4th order

Figure 5.12: Solution for (5.88) with ICs as in (5.156), $\alpha = 0.5$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = N_v = 400$, $M + 1 = 4$. (5.12a) $\Delta t = 1/80$. (5.12b) $\Delta t = 1/300$.



(a) L^1

(b) $\int \int f(x, v, t) dx\, dv$

(c) L^2

(d) Entropy

(e) Energy

Figure 5.13: Physical quantities for (5.88) with ICs as in (5.156), $\alpha = 0.5$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = 64$, $N_v = 128$, and $M + 1 = 4$. The labels 1st, 2nd, 3rd, and 4th in the legends denote first order splitting (prediction only, $\Delta t = 1/40$), 2nd order split IDC (prediction, 1 correction, $\Delta t = 1/40$), 3rd order split IDC (prediction, 2 corrections, $\Delta t = 1/80$), and 4th order split IDC (prediction, 3 corrections, $\Delta t = 1/80$) methods, resp.

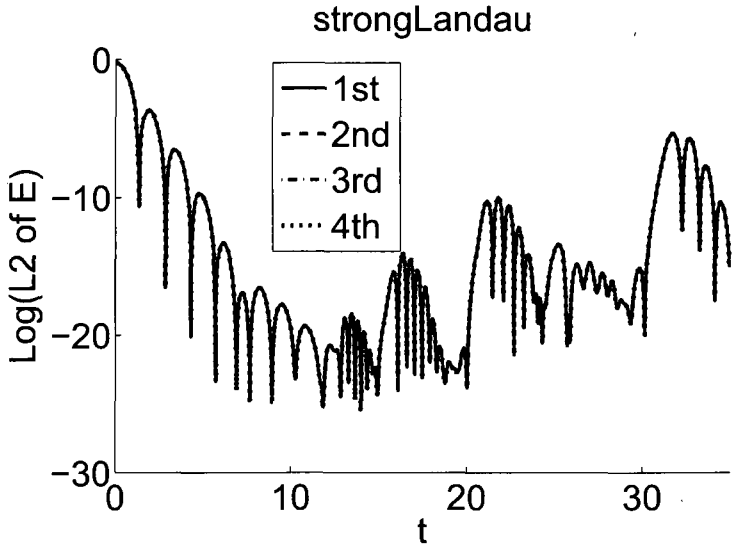


Figure 5.14: Electric field for (5.88) with ICs as in (5.156), $\alpha = 0.5$. IDC constructed with 1st order split methods, each split equation is solved in a semi-Lagrangian setting with conservative 5th order WENO, $N_x = 64$, $N_v = 128$, $M + 1 = 4$. For 1st order: $\Delta t = 1/80$, 2nd order: $\Delta t = 1/80$, 3rd order: $\Delta t = 1/80$, 4th order: $\Delta t = 1/80$.

5.10 Conclusions

Here we have investigated high order splitting methods using IDC methods constructed with low order splitting methods. We have found that these split IDC methods have potential for improved efficiency over other general order splitting methods, requiring a number of solves that increases linearly rather than exponentially with the order of the method. We also proved conservation of mass, and applied some split IDC methods to constant advection, rotating, and Vlasov-Poisson equations to verify the order of accuracy of split IDC methods. We found that split IDC methods maintain conservation properties at least as well as low order splitting methods. Ongoing work is to determine why a CFL restriction is introduced in the IDC correction step, whether that restriction may be mitigated, and to investigate high order splitting for other PDEs. The principles for split IDC methods could

be extended easily from the Vlasov–Poisson operators to other operators suitable for splitting methods, such as the Vlasov–Maxwell, BGK–Poisson, or BGK–Maxwell equations. Future work should also consider how well the high order split IDC methods handle Vlasov equations with certain collision operators (such as BGK), and how boundary conditions may be handled in such a way that high order accuracy is still maintained.

Chapter 6

Future Work

6.1 Asymptotic Preserving Methods

As mentioned in Chapter 3, semi-implicit IDC methods seem ideal to investigate higher order asymptotic preserving (AP) methods by utilizing popular low order AP ARK methods. An AP method is a numerical scheme designed for specific models containing a small scale ϵ (such as shallow water equations with strong diffusive terms), such that numerical solutions behave similarly to analytic solutions in the asymptotic limit, and the timestep is independent of ϵ [43, 44, 64, 63]. Both the numerical method and the form of the model are interdependent and must be chosen carefully to achieve AP results. However, AP schemes are not yet higher than third order in time, and it is the existing first and second order schemes that work best thus far [64, 34]. Using IDC methods, we expect to be able to extend these methods to higher order in time. We anticipate that incorporating AP schemes into the IDC framework will make it possible to maintain stability and mitigate order reduction that occurs in many semi-implicit methods as ϵ shrinks. Additionally, since AP ARK methods may also be described as splitting methods which solve the stiff part of the system first [64, 63], we expect some relationship between the split IDC methods

of Chapter 5 and IDC methods that are constructed via AP ARK methods. In the subsequent sections, we outline details of our current plans for investigating whether IDC methods may possess AP properties.

6.2 IDC with AP-ARK Theory

6.2.1 Moving Towards an AP IDC Proof

The algorithm for IDC methods constructed with AP ARK integrators is identical to the algorithm given in Chapter 3 for IDC-ARK methods, being careful to note that the $c_i^N \neq c_i^S$ for AP ARK integrators. A main concern is whether the resulting IDC-ARK method will preserve the AP property of its constituent ARK method. Applying the ideas in [64, 63], we present a sketch that indicates that, under certain assumptions, and provided we may answer some questions, it is likely that we may obtain IDC-ARK methods that are AP methods. From [64, 63], we wish solve

$$\partial_t U = -\partial_x F(U) + \frac{1}{\epsilon} R(U), \tag{6.1}$$

$$\tag{6.2}$$

where $U \in \mathcal{R}^D$, $F : \mathcal{R}^D \times \mathcal{R}^D$, and $R : \mathcal{R}^D \times \mathcal{R}^D$. If R is a relaxation operator, then $R(U) = 0$ in the limit as $\epsilon \to 0$, and there exists a constant $d \times D$ matrix W with $\text{rank}(W) = d < D$ and

$$WR(U) = 0 \quad \forall U \in \mathcal{R}^D. \tag{6.3}$$

Then also there are d conserved quantities u such that $u = WU$ and $R(U) = 0$ can be solved uniquely for U as a function of u, that is, $U = \mathcal{E}(u)$ and $R(\mathcal{E}(u)) = 0$. Then we may obtain a system of d conservation laws by applying W and (6.3) to

(6.1)

$$\partial_t(WU) + \partial_x(WF(U)) = 0, \tag{6.4}$$

which is holds for every solution U of (6.1). Since $\epsilon \to 0$ gives $R(U) = 0$, then the equilibrium system

$$\partial_t(u) + \partial_x \mathcal{G}(u) = 0, \tag{6.5}$$

where $\mathcal{G}(u) = WF(\mathcal{E}(u))$, is a good approximation to (6.1).

- **Prediction**

Consider the following prediction step to solve (6.1) via an AP ARK method for the predicted solution η_1 at the next timestep

$$\begin{aligned}
U^{(i)} &= U_0 \\
&\quad + \Delta t \left(\sum_{j=1}^{i-1} a_{ij}^N (-\partial_x F(U^{(j)})|_{t=t_0+c_j^N \Delta t}) + \sum_{j=1}^{\nu^S} a_{ij}^S \frac{1}{\epsilon} R(U^{(j)})|_{t=t_0+c_j^S \Delta t} \right), \\
&\quad i = 1, \ldots, \nu^S, \qquad (6.6) \\
\eta_1 &= U_0 + \Delta t \\
&\quad \cdot \left(\sum_{i=1}^{\nu^N} a_{\nu^N i}^N (-\partial_x F(U^{(i)})|_{t=t_0+c_i^N \Delta t}) + \sum_{i=1}^{\nu^S} a_{\nu^S i}^S \frac{1}{\epsilon} R(U^{(i)})|_{t=t_0+c_i^S \Delta t} \right).
\end{aligned}$$

When $\epsilon \to 0$,

$$\sum_{j=1}^{\nu^S} a_{\nu^S i}^S a_{ij}^S \frac{1}{\epsilon} R(U^{(j)}) = 0, \quad i = 1, 2, \ldots, \nu^S.$$

From a lemma in [63], if A^S (the matrix from the Butcher table coefficients for a_{ij}^S) is nonsingular, then $R(U^{(j)}) = 0$. If the implicit part of the ARK method has stiff decay, i.e., if $a_{\nu^S j} = b_j^S$, then $\eta_1 = U^{(\nu^S)}$, and $R(\eta_1) = 0$ also. By a theorem in [63], then the AP ARK scheme in (6.6) becomes the RK scheme characterized by the explicit part of the ARK scheme applied to the equilibrium limit system (6.5). The proof for the prediction step only requires that all $R(U^{(j)}) = 0$, but the proof for the correction step likely requires that $R(\eta_1) = 0$ also. However, if $R(\eta_1) = 0$ also, then the AP ARK scheme is also asymptotically accurate for the prediction step; i.e., the order of accuracy of not only the d conserved quantities but also of the $D - d$ nonconserved quantities should be preserved.

- **Correction**

Now we consider the error equation in order to analyze a correction loop. The error equation is given by

$$\begin{aligned}\partial_t e &= \partial_t U - \partial_t \eta \\ &= -\partial_x F(U) + \frac{1}{\epsilon} R(U) + \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) - r,\end{aligned}$$

where r is the residual,

$$r = \partial_t \eta + \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta),$$

$$\int_{t_0}^{t} r(\tau) d\tau = \eta(t) - U_0 + \int_{t_0}^{t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau.$$

Then the error equation becomes

$$\begin{aligned}\partial_t (e + \int r) = -\partial_x F \left(\eta + (e + \int r) - \int r \right) + \frac{1}{\epsilon} R \left(\eta + (e + \int r) - \int r \right) \\ + \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta),\end{aligned}$$

or, letting $Q = e + \int r$,

$$\partial_t Q = -\partial_x F\left(\eta + Q - \int r\right) + \frac{1}{\epsilon} R\left(\eta + Q - \int r\right) + \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta),$$

or

$$\begin{aligned}
\partial_t Q = &-\partial_x F\left(Q + U_0 - \int_{t_0}^{t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau\right) \\
&+ \frac{1}{\epsilon} R\left(Q + U_0 - \int_{t_0}^{t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau\right) \\
&+ \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta), \\
Q_0 = e_0 + &\int_0^0 r = 0.
\end{aligned} \tag{6.7}$$

Now we solve equation (6.7) via an AP ARK method whose implicit part has stiff decay, so $Q^{(\nu^S)}$ is also the approximate solution to (6.7) at the next timestep.

$$\begin{aligned}
Q^{(i)} = Q_0 + \Delta t \sum_{j=1}^{\nu^N} a_{ij}^N \Bigg(&-\partial_x F\left(Q^{(j)} + U_0 - \int_{t_0}^{t_0 + c_j^N \Delta t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau\right) \\
&+ \partial_x F(\eta(t_0 + c_j^N \Delta t)) \Bigg) \\
+ \Delta t \sum_{j=1}^{\nu^S} a_{ij}^S \frac{1}{\epsilon} \Bigg(&R\left(Q^{(j)} + U_0 - \int_{t_0}^{t_0 + c_j^S \Delta t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau\right) \\
&- R(\eta(t_0 + c_j^S \Delta t)) \Bigg), \qquad i = 1, \ldots, \nu^S.
\end{aligned}$$

Then multiply by ϵ to obtain

$$\epsilon Q^{(i)} = \epsilon Q_0 + \Delta t \sum_{j=1}^{\nu^N} a_{ij}^N \left(-\epsilon \partial_x F \left(Q^{(j)} + U_0 - \int_{t_0}^{t_0 + c_j^N \Delta t} \partial_x F(\eta) - \frac{1}{\epsilon} R(\eta) d\tau \right) \right.$$
$$\left. + \epsilon \partial_x F(\eta(t_0 + c_j^N \Delta t)) \right)$$
$$+ \Delta t \sum_{j=1}^{\nu^S} a_{ij}^S \left(R \left(Q^{(j)} + U_0 - \int_{t_0}^{t_0 + c_j^S \Delta t} \partial_x F(\eta) - R(\eta) d\tau \right) \right.$$
$$\left. - R(\eta(t_0 + c_j^S \Delta t)) \right), \tag{6.8}$$

and consider the limit as ϵ approaches zero. The left hand side of (6.8) goes to zero. The first term on the right hand side also goes to zero. We wish to prove that the entire explicit sum and the second part of the implicit sum go to zero, which would leave us with

$$R \left(Q^{(j)} + U_0 - \int_{t_0}^{t_0 + c_j^S \Delta t} \partial_x F(\eta) - R(\eta) d\tau \right) = 0, \quad j = 1, \ldots, \nu^S, \tag{6.9}$$

since A^S is nonsingular. If (6.9) holds, then, by a similar process as in the rest of the proof of the theorem in [63], the correction step possesses the AP property. It seems likely that the explicit sum goes to zero, but it is important to take care with the fact that the quantity inside F in the first part of the explicit sum contains a $1/\epsilon$ term. It also seems likely that the second part of the implicit sum goes to zero, but the fact that the values in the second part of the implicit sum are interpolated values may affect that conclusion. However, we know that $R(\eta(t_0))$ goes to zero, and if the AP ARK scheme has stiff decay, then we know that $R(\eta_1)$ go to zero as well. By the same reasoning as for the prediction, the correction step is also asymptotically accurate if the AP ARK method's implicit part has stiff decay.

6.3 AP IDC Methods for P_1 Equations

Here we outline a plan for understanding how to implemen AP IC methods applied to the P_1 equations in [34].

6.3.1 The P_1 Problem and the Splitting

We begin iwth a presentation of the P_1 system and its splitting that is used in AP methods. The original P_1 system that we desire to solve is

$$\begin{aligned} \partial_t \rho + a\partial_x m &= 0, \\ \partial_t m + \frac{b}{\epsilon^2}\partial_x \rho &= -\frac{\sigma}{\epsilon^2} m, \\ \rho(x, t_0) &= \rho_0(x), \\ m(x, t_0) &= m_0(x), \end{aligned} \tag{6.10}$$

where a, b are constants, $\sigma = \sigma(x)$, and $t \in [t_0, t_0 + \Delta t]$.

The first order splitting of (6.10) is given by

$$\begin{aligned} \partial_t \rho^{(1)} &= 0, \\ \partial_t m^{(1)} + \frac{b}{\epsilon^2}(1 - \epsilon^2)\partial_x \rho^{(1)} &= -\frac{\sigma}{\epsilon^2} m^{(1)}, \\ \rho^{(1)}(x, t_0) &= \rho_0(x), \\ m^{(1)}(x, t_0) &= m_0(x), \end{aligned} \tag{6.11}$$

which solves in the stiff direction first, followed by

$$\begin{aligned} \partial_t \rho^{(2)} + a\partial_x m^{(2)} &= 0, \\ \partial_t m^{(2)} + b\partial_x \rho^{(2)} &= 0, \\ \rho^{(2)}(x, t_0) &= \rho^{(1)}(x, t_0 + \Delta t), \\ m^{(2)}(x, t_0) &= m^{(1)}(x, t_0 + \Delta t), \end{aligned} \tag{6.12}$$

the nonstiff direction.

Then rewriting the original system and the splitting with matrices and vectors, we obtain the original system,

$$\partial_t \begin{pmatrix} \rho \\ m \end{pmatrix} = \begin{bmatrix} 0 & -a\partial_x \\ -\frac{b}{\epsilon^2}\partial_x & -\frac{\sigma}{\epsilon^2} \end{bmatrix} \begin{pmatrix} \rho \\ m \end{pmatrix},$$

and the splitting,

$$\partial_t \begin{pmatrix} \rho^{(1)} \\ m^{(1)} \end{pmatrix} = \begin{bmatrix} 0 & 0 \\ -\frac{b(1-\epsilon^2)}{\epsilon^2}\partial_x & -\frac{\sigma}{\epsilon^2} \end{bmatrix} \begin{pmatrix} \rho^{(1)} \\ m^{(1)} \end{pmatrix},$$

$$\partial_t \begin{pmatrix} \rho^{(2)} \\ m^{(2)} \end{pmatrix} = \begin{bmatrix} 0 & -a\partial_x \\ -b\partial_x & 0 \end{bmatrix} \begin{pmatrix} \rho^{(2)} \\ m^{(2)} \end{pmatrix},$$

and we choose the following matrix and vector designations:

$$u = \begin{pmatrix} \rho^{(1)} \\ m^{(1)} \end{pmatrix}, \quad u = \begin{pmatrix} \rho^{(1)} \\ m^{(1)} \end{pmatrix}, \quad u^{(2)} = \begin{pmatrix} \rho^{(2)} \\ m^{(2)} \end{pmatrix}, \tag{6.13}$$

$$F = \begin{bmatrix} 0 & 0 \\ -b(1-\epsilon^2)\partial_x & -\sigma \end{bmatrix}, \quad G = \begin{bmatrix} 0 & -a\partial_x \\ -b\partial_x & 0 \end{bmatrix}.$$

Then we obtain the original system,

$$\partial_t u = \left(\frac{1}{\epsilon^2}F + G\right) u, \tag{6.14}$$
$$u(x, t_0) = u_0(x),$$

and the splitting,

$$\partial_t u^{(1)} = \frac{1}{\epsilon^2} F u^{(1)}, \tag{6.15}$$
$$u^{(1)}(x, t_0) = u_0(x);$$

$$\partial_t u^{(2)} = G u^{(2)}, \tag{6.16}$$
$$u^{(2)}(x, t_0) = u^{(1)}(x, t_0 + \Delta t),$$

in vector form. This notation is for ease of the calculations in the following sections.

6.3.2 Asymptotic Limit

In the limit as $\epsilon \to 0$, the system (6.10) becomes

$$\partial_t \rho + a\partial_x \left(\frac{b}{\sigma}\partial_x \rho\right) = 0. \tag{6.17}$$

For an asymptotic preserving method, the numerical solution of (6.10) should match the numerical solution of (6.17) in the $\epsilon \to 0$ limit, for $\Delta t > \epsilon^2$.

6.3.3 Splitting Error

Now we analyze the splitting error by Taylor expanding the exact solution u of (6.14),

$$u(x, t_0 + \Delta t) = (I + \Delta t \partial_t + \frac{\Delta t^2}{2} \partial_{tt} + \dots) u_0(x),$$

where I is the identity matrix. We replace the t derivatives with their equivalent expressions from (6.14),

$$\begin{aligned} \partial_t &= \frac{1}{\epsilon^2} F + G, \\ \partial_{tt} &= \left(\frac{1}{\epsilon^2} F + G\right)\left(\frac{1}{\epsilon^2} F + G\right) \\ &= \frac{1}{\epsilon^4} F^2 + \frac{1}{\epsilon^2}(FG + GF) + G^2, \end{aligned}$$

and obtain

$$\begin{aligned} u(x, t_0 + \Delta t) = \Bigg(I + \Delta t \left(\frac{1}{\epsilon^2} F + G\right) \\ + \frac{\Delta t^2}{2}\left(\frac{1}{\epsilon^4} F^2 + \frac{1}{\epsilon^2}(FG + GF) + G^2\right) + \dots\Bigg) u_0(x). \end{aligned} \tag{6.18}$$

Similarly, we expand the solutions to (6.15) and (6.16) to obtain

$$
\begin{aligned}
u^{(1)}(x,t_0+\Delta t) &= \left(I+\Delta t\frac{1}{\epsilon^2}F+\frac{\Delta t^2}{2}\frac{1}{\epsilon^4}F^2+\dots\right)u^{(1)}(x,t_0), \qquad (6.19)\\
&= \left(I+\Delta t\frac{1}{\epsilon^2}F+\frac{\Delta t^2}{2}\frac{1}{\epsilon^4}F^2+\dots\right)u_0(x),\\
u^{(2)}(x,t_0+\Delta t) &= \left(I+\Delta tG+\frac{\Delta t^2}{2}G^2+\dots\right)u^{(2)}(x,t_0)\\
&= \left(I+\Delta tG+\frac{\Delta t^2}{2}G^2+\dots\right)u^{(1)}(x,t_0+\Delta t)\\
&= \left(I+\Delta t\left(\frac{1}{\epsilon^2}F+G\right)+\frac{\Delta t^2}{2}\left(\frac{1}{\epsilon^4}F^2+\frac{2}{\epsilon^2}GF+G^2\right)+\dots\right)u_0(x).
\end{aligned}
$$

Since F and G do not commute, then the exact (6.18) and split (6.19) solutions differ in the Δt^2 term, giving a first order splitting.

Alternatively, the solution to the split equations is given by

$$u^{(2)}(x,t_0+\Delta t)=\exp\left((t_0+\Delta_t)\frac{1}{\epsilon^2}F\right)\exp\left((t_0+\Delta_t)G\right)u_0(x),$$

which indicates that, if F is negative semi-definite, then it seems more likely that the method will approximate (6.17) as $\epsilon\to 0$. Also, for the split solution not to blow up as $\epsilon\to 0$, we require

$$Fu=\mathcal{O}(\epsilon^2),\quad u=\bar{u}+\epsilon^2\tilde{u},\quad \bar{u}\in\mathcal{N}(F).$$

6.3.4 Split Scheme with one IDC Correction

Now we present the form that the error equation and the split error equation should take in one correction loop of an IDC method. Let u (as defined in (6.13)) be the exact solution to (6.14), η be the split solution after solving (6.15) and (6.16) exactly,

i.e. $\eta = u^{(2)}$, and define the error, e, and residual, r, as

$$\begin{aligned} e &= u - \eta, \\ r &= \partial_t \eta - \left(\frac{1}{\epsilon^2} F + G\right) \eta. \end{aligned}$$

Differentiating e with respect to t, we obtain the error equation,

$$\begin{aligned} \partial_t e &= \partial_t u - \partial_t \eta \\ &= \left(\frac{1}{\epsilon^2} F + G\right) u - \left(\left(\frac{1}{\epsilon^2} F + G\right) \eta + r\right) \\ &= \left(\frac{1}{\epsilon^2} F + G\right) e - r, \\ e(x, t_0) &= 0. \end{aligned} \tag{6.20}$$

Then we approximate the solution to the error equation (6.20) by the three-part first order splitting

$$\begin{aligned} \partial_t e^{(1)} &= \frac{1}{\epsilon^2} F e^{(1)}, \\ e^{(1)}(x, t_0) &= e_0(x); \end{aligned} \tag{6.21}$$

$$\begin{aligned} \partial_t e^{(2)} &= G e^{(2)}, \\ e^{(2)}(x, t_0) &= e^{(1)}(x, t_0 + \Delta t); \end{aligned} \tag{6.22}$$

$$\begin{aligned} \partial_t e^{(3)} &= -r = -\partial_t \left(\eta - u_0 - \int_{t_0}^{t} \left(\frac{1}{\epsilon^2} F + G\right) \eta \, d\tau\right), \\ e^{(3)}(x, t_0) &= e^{(2)}(x, t_0 + \Delta t). \end{aligned} \tag{6.23}$$

Now for comparison with the common form of (6.10), we change from the vector notation back to the component-wise notation. The error equation (6.20) becomes

$$\partial_t e_\rho + a\partial_x e_m = -\partial_t \left(\eta_\rho - \rho_0 + \int_{t_0}^{t} a\partial_x \eta_m \, d\tau \right),$$

$$\partial_t e_m + \frac{b}{\epsilon^2}\partial_x e_\rho = -\frac{\sigma}{\epsilon^2} e_m - \partial_t \left(\eta_m - m_0 + \int_{t_0}^{t} (b + \frac{b(1-\epsilon^2)}{\epsilon^2})\partial_x \eta_\rho + \frac{\sigma}{\epsilon^2}\eta_m \, d\tau \right),$$

$$e_\rho(x, t_0) = e_{\rho,0}(x),$$

$$e_m(x, t_0) = e_{m,0}(x),$$

and the split form (6.21), (6.22), (6.23) of the error equation becomes

$$\partial_t e_\rho^{(1)} = 0, \tag{6.24}$$

$$\partial_t e_m^{(1)} + \frac{b}{\epsilon^2}(1-\epsilon^2)\partial_x e_\rho^{(1)} = -\frac{\sigma}{\epsilon^2} e_m^{(1)},$$

$$e_\rho^{(1)}(x, t_0) = e_{\rho,0}(x),$$

$$e_m^{(1)}(x, t_0) = e_{m,0}(x);$$

$$\partial_t e_\rho^{(2)} + a\partial_x e_m^{(2)} = 0, \tag{6.25}$$

$$\partial_t e_m^{(2)} + b\partial_x e_\rho^{(2)} = 0, \tag{6.26}$$

$$e_\rho^{(2)}(x, t_0) = e_\rho^{(1)}(x, t_0 + \Delta t), \tag{6.27}$$

$$e_m^{(2)}(x, t_0) = e_m^{(1)}(x, t_0 + \Delta t); \tag{6.28}$$

$$\partial_t e_\rho^{(3)} = -\partial_t \left(\eta_\rho - \rho_0 + \int_{t_0}^{t} a \partial_x \eta_m \, d\tau \right), \tag{6.29}$$

$$\partial_t e_m^{(3)} = -\partial_t \left(\eta_m - m_0 + \int_{t_0}^{t} (b + \frac{b(1-\epsilon^2)}{\epsilon^2}) \partial_x \eta_\rho + \frac{\sigma}{\epsilon^2} \eta_m \, d\tau \right), \tag{6.30}$$

$$e_\rho^{(3)}(x, t_0) = e_\rho^{(2)}(x, t_0 + \Delta t), \tag{6.31}$$

$$e_m^{(3)}(x, t_0) = e_m^{(2)}(x, t_0 + \Delta t). \tag{6.32}$$

6.3.5 Analytic Solutions of Splitting Parts

Here we give the analytic solutions to the split equations (6.11), (6.12), and the split error equations (6.24), (6.25), and (6.29). To simplify the presentation, let $t = t_0 + \Delta t$. The solution to (6.11) is exactly

$$\begin{aligned} \rho^{(1)}(t) &= \rho^{(1)}(t_0), \\ m^{(1)}(t) &= -\exp\left(-\frac{\sigma}{\epsilon^2} t\right) \int_{t_0}^{t} \frac{b}{\epsilon^2} \exp\left(\frac{\sigma}{\epsilon^2} \tau\right) \partial_x \rho^{(1)} \, d\tau \\ &= -\frac{b}{\sigma} \partial_x \rho^{(1)} \left(1 - \exp\left(\frac{\sigma}{\epsilon^2}(t - t_0)\right)\right), \end{aligned} \tag{6.33}$$

where the second equality for $m^{(1)}$ holds if there is enough smoothness such that $\rho^{(1)}$ constant in t gives $\partial_x \rho^{(1)}$ also constant in t.

Now we solve (6.12). Left multiplying (6.12) by

$$V_L = \begin{bmatrix} \frac{1}{2a} & \frac{1}{2\sqrt{ab}} \\ -\frac{1}{2a} & \frac{1}{2\sqrt{ab}} \end{bmatrix},$$

one obtains

$$\begin{aligned}
&\partial_t \Psi^+ + \sqrt{ab}\,\partial_x \Psi^+ = 0, \\
&\partial_t \Psi^- - \sqrt{ab}\,\partial_x \Psi^- = 0, \\
&\Psi^+ = \frac{\rho^{(2)}}{2a} + \frac{m^{(2)}}{2\sqrt{ab}}, \quad \Psi^- = -\frac{\rho^{(2)}}{2a} + \frac{m^{(2)}}{2\sqrt{ab}}.
\end{aligned} \tag{6.34}$$

Then the solution for (6.34) is given by

$$\begin{aligned}
\Psi^+(x,t) &= \Psi^+(x - \sqrt{ab}\,t, t_0), \\
\Psi^-(x,t) &= \Psi^-(x + \sqrt{ab}\,t, t_0).
\end{aligned}$$

Then some simple calculations give the solution to (6.12) as

$$\begin{aligned}
\rho^{(2)}(x,t) &= \frac{1}{2}\left(\rho^{(2)}(x - \sqrt{ab}\,t, t_0) + \rho^{(2)}(x + \sqrt{ab}\,t, t_0)\right) \\
&\quad + \frac{\sqrt{a}}{2\sqrt{b}}\left(m^{(2)}(x - \sqrt{ab}\,t, t_0) - m^{(2)}(x + \sqrt{ab}\,t, t_0)\right), \\
m^{(2)}(x,t) &= \frac{\sqrt{a}}{2\sqrt{b}}\left(\rho^{(2)}(x - \sqrt{ab}\,t, t_0) - \rho^{(2)}(x + \sqrt{ab}\,t, t_0)\right) \\
&\quad + \frac{1}{2}\left(m^{(2)}(x - \sqrt{ab}\,t, t_0) + m^{(2)}(x + \sqrt{ab}\,t, t_0)\right).
\end{aligned} \tag{6.35}$$

The solutions for the split error equations (6.24) and (6.25) are analogous to the solutions (6.33) and (6.35) of (6.11) and (6.12), respectively, and the solution for the part of the error splitting (6.29) is given by

$$e_\rho^{(3)}(x,t) = e_\rho^{(3)}(x,t_0) - \left(\eta_\rho(x,t) - \rho_0(x) + \int_{t_0}^{t} a\partial_x \eta_m(x,\tau)\,d\tau\right), \tag{6.36}$$

$$\begin{aligned}
e_m^{(3)}(x,t) &= e_m^{(3)}(x,t_0) \\
&\quad - \left(\eta_m(x,t) - m_0(x) + \int_{t_0}^{t} (b + \frac{b(1-\epsilon^2)}{\epsilon^2})\partial_x \eta_\rho(x,\tau) + \frac{\sigma}{\epsilon^2}\eta_m(x,\tau)\,d\tau\right).
\end{aligned}$$

6.3.6 Anticipated Numerical Tests

Here we discuss the plans for numerical tests to determine the effectiveness of IDC in the AP setting, in particular, to determine whether the IDC framework is capable of maintaining the AP property. To start with, we plan to use periodic boundary conditions and smooth initial conditions. We will solve each splitting analytically, which is essentially a semi-Lagrangian method. It may be necessary to use WENO reconstruction for some places where ∂_x appears, but in other places, setting $\Delta x = \Delta t$ may mean WENO reconstruction is unnecessary. Note that within the WENO method, the Lax-Friedrichs flux may cause problems for these P_1 equations. Perhaps instead a cubic spline may be used, especially if the nonoscillatory features of WENO are not required. We intend to test various values of ϵ, for $\Delta t > \epsilon^2$. Note, however, that the CFL limitation found numerically for the split IDC methods in Chapter 5.6 will need to be understood much better before these tests can be performed reasonably.

BIBLIOGRAPHY

[1] A. L. Araújo, A. Murua, and J. M. Sanz-Serna. Symplectic methods based on decompositions. *SIAM J. Numer. Anal.*, 34(5):1926–1947, 1997.

[2] Uri M. Ascher and Linda R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[3] 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(2-3):151–167, 1997. Special issue on time integration (Amsterdam, 1996).

[4] Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.*, 32(3):797–823, 1995.

[5] W. Auzinger, H. Hofstätter, W. Kreuzer, and E. Weinmüller. Modified defect correction algorithms for ODEs. I. General theory. *Numer. Algorithms*, 36(2):135–155, 2004.

[6] W. Auzinger, H. Hofstätter, W. Kreuzer, and E. Weinmüller. Modified defect correction algorithms for ODEs. II. Stiff initial value problems. *Numer. Algorithms*, 40(3):285–303, 2005.

[7] C. Bolley and M. Crouzeix. Conservation de la positivité lors de la discrétisation des problemes dévolution paraboliques. *RAIRO Anal. Numér*, 12(3):237–245, 1978.

[8] Sebastiano Boscarino. On an accurate third order implicit-explicit Runge–Kutta method for stiff problems. *Appl. Numer. Math.*, 59(7):1515–1528, 2009.

[9] Anne Bourlioux, Anita T. Layton, and Michael L. Minion. High-order multi-implicit spectral deferred correction methods for problems of reactive flow. *J. Comput. Phys.*, 189(2):651–675, 2003.

[10] T. J. M. Boyd and J. J. Sanderson. *The physics of plasmas.* Cambridge University Press, Cambridge, 2003.

[11] M. P. Calvo, J. de Frutos, and J. Novo. Linearly implicit Runge–Kutta methods for advection-reaction-diffusion equations. *Appl. Numer. Math.*, 37(4):535–549, 2001.

[12] J. A. Carrillo and F. Vecil. Nonoscillatory interpolation methods applied to Vlasov-based models. *SIAM J. Sci. Comput.*, 29(3):1179–1206 (electronic), 2007.

[13] C.Z. Cheng and G. Knorr. The integration of the Vlasov equation in configuration space. *J. Comput. Phys.*, 22(3):330–351, 1976.

[14] A. Christlieb, R. Krasny, and B. Ong. Particle simulation of Weak Solutions of the Vlasov–Poisson equations. In preparation.

[15] A. Christlieb, M. Morton, B. Ong, and J.-M. Qiu. Semi-implicit integral deferred correction constructed with high order additive Runge–Kutta methods. Submitted.

[16] Andrew Christlieb, Colin Macdonald, and Benjamin Ong. Parallel high-order integrators. *SIAM J. Sci. Comput.*, 32(2):818–835, 2010.

[17] Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu. Comments on high order integrators embedded within integral deferred correction methods. *Comm. Appl. Math. Comput. Sci.*, 4(1):27–56, 2009.

[18] Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu. Integral deferred correction methods constructed with high order runge–kutta integrators. *Math. Comput.*, 79:761–783, 2010.

[19] B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor. Advanced numerical approximation of nonlinear hyperbolic equations. 1697:vi+433, 1998. Papers from the C.I.M.E. Summer School held in Cetraro, June 23–28, 1997, Edited by Alfio Quarteroni, Fondazione C.I.M.E.. [C.I.M.E. Foundation].

[20] G. J. Cooper and A. Sayfy. Additive methods for the numerical solution of ordinary differential equations. *Math. Comp.*, 35(152):1159–1172, 1980.

[21] G. J. Cooper and A. Sayfy. Additive Runge–Kutta methods for stiff ordinary differential equations. *Math. Comp.*, 40(161):207–218, 1983.

[22] G.-H. Cottet and P.-A. Raviart. Particle methods for the one-dimensional Vlasov-Poisson equations. *SIAM J. Numer. Anal.*, 21(1):52–76, 1984.

[23] N. Crouseilles, M. Mehrenberger, and E. Sonnendrücker. Conservative semi-Lagrangian schemes for Vlasov equations. *J. Comput. Phys.*, 229(6):1927 – 1953, 2010.

[24] P. Csomós and I. Faragó. Error analysis of the numerical solution of split differential equations. *Math. Comput. Modelling*, 48(7-8):1090–1106, 2008.

[25] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT*, 40(2):241–266, 2000.

[26] F. Filbet and E. Sonnendrücker. Comparison of Eulerian Vlasov solvers. *Comput. Phys. Comm.*, 150(3):247–266, 2003.

[27] R. Frank and C. W. Ueberhuber. Iterated defect correction for differential equations. I. Theoretical results. *Computing*, 20(3):207–228, 1978.

[28] Reinhard Frank and Christoph W. Ueberhuber. Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. *Nordisk Tidskr. Informationsbehandling (BIT)*, 17(2):146–159, 1977.

[29] D. Gottlieb. Strang-type difference schemes for multidimensional problems. *SIAM J. Numer. Anal.*, 9:650–661, 1972.

[30] Sigal Gottlieb, David I. Ketcheson, and Chi-Wang Shu. High order strong stability preserving time discretizations. *J. Sci. Comput.*, 38(3):251–289, 2009.

[31] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43(1):89–112 (electronic), 2001.

[32] L. Greengard. Spectral integration and two-point boundary value problems. *SIAM J. Numer. Anal.*, 28(4):1071–1080, 1991.

[33] Kjell Gustafsson. Control-theoretic techniques for stepsize selection in implicit Runge–Kutta methods. *ACM Trans. Math. Software*, 20(4):496–517, 1994.

[34] J.R. Haack and C.D. Hauck. Oscillatory behavior of asymptotic-preserving splitting methods for a linear model of diffusive relaxation. *Los Alamos Report LA-UR-08-0571.*

[35] Wolfgang Hackbusch. Bemerkungen zur iterierten Defektkorrektur und zu ihrer Kombination mit Mehrgitterverfahren. *Rev. Roumaine Math. Pures Appl.*, 26(10):1319–1329, 1981.

[36] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1987. Nonstiff problems.

[37] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. Nonstiff problems.

[38] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006. Structure-preserving algorithms for ordinary differential equations.

[39] E. Hansen and A. Ostermann. High order splitting methods for analytic semigroups exist. *BIT Numerical Mathematics*, 49(3):527–542, 2009.

[40] P. W. Hemker. Mixed defect correction iteration for the solution of a singular perturbation problem. In *Defect correction methods (Oberwolfach, 1983)*, volume 5 of *Comput. Suppl.*, pages 123–145. Springer, Vienna, 1984.

[41] Jingfang Huang, Jun Jia, and Michael Minion. Accelerating the convergence of spectral deferred correction methods. *J. Comput. Phys.*, 214(2):633–656, 2006.

[42] Jingfang Huang, Jun Jia, and Michael Minion. Arbitrary order Krylov deferred correction methods for differential algebraic equations. *J. Comput. Phys.*, 221(2):739–760, 2007.

[43] Shi Jin. Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations. *SIAM J. Sci. Comput.*, 21(2):441–454 (electronic), 1999.

[44] Shi Jin, Lorenzo Pareschi, and Giuseppe Toscani. Uniformly accurate diffusive relaxation schemes for multiscale transport equations. *SIAM J. Numer. Anal.*, 38(3):913–936 (electronic), 2000.

[45] Christopher A. Kennedy and Mark H. Carpenter. Additive Runge–Kutta schemes for convection-diffusion-reaction equations. *Appl. Numer. Math.*, 44(1-2):139–181, 2003.

[46] Wendy Kress and Bertil Gustafsson. Deferred correction methods for initial boundary value problems. In *Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala)*, volume 17, pages 241–251, 2002.

[47] Anita T. Layton. On the choice of correctors for semi-implicit Picard deferred correction methods. *Appl. Numer. Math.*, 58(6):845–858, 2008.

[48] Anita T. Layton and Michael L. Minion. Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods. *Commun. Appl. Math. Comput. Sci.*, 2:1–34 (electronic), 2007.

[49] Jongwoo Lee and Bengt Fornberg. A split step approach for the 3-D Maxwell's equations. *J. Comput. Appl. Math.*, 158(2):485–505, 2003.

[50] Jongwoo Lee and Bengt Fornberg. Some unconditionally stable time stepping methods for the 3D Maxwell's equations. *J. Comput. Appl. Math.*, 166(2):497–523, 2004.

[51] Randall J. LeVeque. *Numerical methods for conservation laws.* Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, second edition, 1992.

[52] R.J. LeVeque. Intermediate boundary conditions for LOD, ADI and approximate factorization methods. 1985.

[53] R.J. LeVeque. Intermediate boundary conditions for time-split methods applied to hyperbolic partial differential equations. *Math. Comp.*, 47(175):37–54, 1986.

[54] M.A. Lieberman and A.J. Lichtenberg. *Principles of plasma discharges and materials processing.* Wiley-Blackwell, 2005.

[55] Hongyu Liu. *Personal correspondence*, 2008.

[56] Hongyu Liu and Jun Zou. Some new additive Runge–Kutta methods and their applications. *J. Comput. Appl. Math.*, 190(1-2):74–98, 2006.

[57] Yuan Liu, Chi-Wang Shu, and Mengping Zhang. Strong stability preserving property of the deferred correction time discretization. *J. Comput. Math.*, 26(5):633–656, 2008.

[58] G. I. Marčuk. Some application of splitting-up methods to the solution of mathematical physics problems. *Apl. Mat.*, 13:103–132, 1968.

[59] Francesca Mazzia and Cecilia Magherini. Test set for initial value problem solvers, release 2.4. Technical Report 4, Department of Mathematics, University of Bari, Italy, February 2008. Available at `http://pitagora.dm.uniba.it/~testset`.

[60] Michael L. Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Commun. Math. Sci.*, 1(3):471–500, 2003.

[61] A. Ostermann and M. van Daele. Positivity of exponential Runge–Kutta methods. *BIT Numerical Mathematics*, 47(2):419–426, 2007.

[62] Lorenzo Pareschi and Giovanni Russo. Implicit-explicit Runge–Kutta schemes for stiff systems of differential equations. In *Recent trends in numerical analysis*, volume 3 of *Adv. Theory Comput. Math.*, pages 269–288. Nova Sci. Publ., Huntington, NY, 2001.

[63] Lorenzo Pareschi and Giovanni Russo. High order asymptotically strong-stability-preserving methods for hyperbolic systems with stiff relaxation. In *Hyperbolic problems: theory, numerics, applications*, pages 241–251. Springer, Berlin, 2003.

[64] Lorenzo Pareschi and Giovanni Russo. Implicit-Explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation. *J. Sci. Comput.*, 25(1-2):129–155, 2005.

[65] Victor Pereyra. Iterated deferred corrections for nonlinear operator equations. *Numer. Math.*, 10:316–323, 1967.

[66] J.-M. Qiu. *Lecture, IPAM, UCLA*, 2009.

[67] J.-M. Qiu and A. Christlieb. A conservative high order semi-Lagrangian WENO method for the Vlasov equation. *J. Comput. Phys.*, 229(4):1130–1149, 2010.

[68] J. Schaeffer. *Personal correspondence*, 2010.

[69] Jack Schaeffer. Higher order time splitting for the linear Vlasov equation. *SIAM J. Numer. Anal.*, 47(3):2203–2223, 2009.

[70] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations (Cetraro, 1997)*, volume 1697 of *Lecture Notes in Math.*, pages 325–432. Springer, Berlin, 1998.

[71] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. The semi-Lagrangian method for the numerical resolution of the Vlasov equation. *J. Comput. Phys.*, 149(2):201–220, 1999.

[72] Hans J. Stetter. The defect correction principle and discretization methods. *Numer. Math.*, 29(4):425–443, 1977/78.

[73] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5:506–517, 1968.

[74] J. van Dijk, G.M.W. Kroesen, and A. Bogaerts. Plasma modelling and numerical simulation. *J. Phys. D: Appl. Phys*, 42(19).

[75] J.P. Verboncoeur. Particle simulation of plasmas: review and advances. *Plasma physics and controlled fusion*, 47:A231–A260, 2005.

[76] Yinhua Xia, Yan Xu, and Chi-Wang Shu. Efficient time discretization for local discontinuous Galerkin methods. *Discrete Contin. Dyn. Syst. Ser. B*, 8(3):677–693, 2007.

[77] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5-7):262–268, 1990.

[78] Pedro E. Zadunaisky. On the estimation of errors propagated in the numerical integration of ordinary differential equations. *Numer. Math.*, 27(1):21–39, 1976/77.