Efficient Runge-Kutta Based Local Time-Stepping Methods

by

Alex Ashbourne

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2016

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The method of lines approach to the numerical solution of transient hyperbolic partial differential equations (PDEs) allows us to write the PDE as a system of ordinary differential equations (ODEs) in time. Solving this system of ODEs explicitly requires choosing a stable time step satisfying the Courant-Friedrichs-Lewy (CFL) condition. When a uniform mesh is used, the global CFL number is used to choose the time step and the system is advanced uniformly in time. The need for local time-stepping, i.e., advancing elements by their maximum locally defined time step, occurs when the elements in the mesh differ greatly in size. When global time-stepping is used, the global CFL number and the globally defined time step are defined by the smallest element in the mesh. This leads to inefficiencies as a few small elements impose a restrictive time step on the entire mesh. Local time-stepping mitigates these inefficiencies by advancing elements by their locally defined time step and, hence, reduces the number of function evaluations.

In this thesis, we present two local time-stepping algorithms based on a third order Runge-Kutta method and the classical fourth order Runge-Kutta method. We prove these methods keep the order of accuracy of the underlying Runge-Kutta methods in the context of a system of ODEs. We then show how they can be used with the method of lines approach to the numerical solution of PDEs, specifically with the discontinuous Galerkin (DG) spatial discretization. Numerical simulations show we obtain the theoretical $p+1$ rate of convergence of the DG method in both the L^2 and maximum norms. We provide evidence that these algorithms are stable through a number of linear and nonlinear examples.

Acknowledgements

First, I would like to thank my advisor, Lilia Krivodonova. Her expertise and experience have allowed me to learn a tremendous amount through the development of this thesis. I would like to thank my committee members, Justin Wan and Sander Rhebergen. I would also like to extend my gratitude to my fellow students Andrew Giuliani and Noel Chalmers for their helpful discussions on theory and implementation. Finally, I would like to thank my family and friends for all their support and enthusiasm during the preparation of this thesis.

Dedication

To my beautiful girlfriend, Vanessa. Without your support this would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hyperbolic partial differential equations appear in many fields of science and engineering, especially where wave phenomenon or advective transport are present. Among other applications, they are used to model electromagnetic or acoustic wave propagation, gas dynamics problems, and elastic waves in solids. Solutions of nonlinear hyperbolic partial differential equations can develop fine structures that need to be resolved, solutions can also steepen into shocks. This can occur even with smooth initial data [23]. In order to capture these features accurately, we require small mesh elements. The modern approach is to use an adaptive mesh rather than a uniform mesh so that we can allocate computational resources to regions where they are needed most. Adaptive meshes are also useful to accurately represent small features of the geometry.

When using an adaptive mesh, element sizes can vary greatly throughout the computational domain. By the Courant-Friedrichs-Lewy (CFL) condition, the time step on each element must be proportional to the element's size for the method to be stable. If a uniform time step is used over the whole mesh, as is often the practice with the method of lines and explicit time integration methods, it is determined by the smallest element (the global CFL condition). This can be very inefficient as a few small elements impose a restrictive time step for all elements. A possible solution would be to use an implicit, unconditionally stable time integration scheme as this would eliminate the CFL condition. However, this has a disadvantage of requiring the solution of a large sparse linear system at each time step.

Locally implicit methods allow explicit time-integrators to be used on large elements but implicit time-integrators on smaller elements, where the time step would have been drastically reduced. In [20], a fourth order implicit-explicit Runge-Kutta method is applied

with the nodal discontinuous Galerkin discretization to problems in fluid flow. Using an implicit time-integrator on the small elements creates a nonlinear system which needs to be solved at each time step.

Explicit local time-stepping methods overcome the effects of local refinement by using smaller time steps in the area of refined elements while keeping the time-integrator explicit throughout the entire computational domain. This increases efficiency as larger elements are not required to take as many time steps. The difficulty with explicit local time-stepping is in the communication between elements of different sizes when they are at different time levels. In [15], a second order local time stepping procedure is combined with discontinous Galerkin discretization for solving symmetric hyperbolic systems. The method is fully explicit but requires the solution of a linear system at each time step to resolve the interface between large and small elements.

A local time-stepping approach is presented in [3, 2] in the context of adaptive mesh refinement (AMR) with Cartesian girds. The AMR algorithm creates a set of finer and finer subgrids which are allowed to overlap and nest in areas where finer resolution is required. The time step on a fine grid is chosen in proportion to the time step of the largest grid based on its level of refinement. The grids are updated in time from largest to smallest with information being passed between grid levels through interpolation.

An alternative approach, based on the arbitrarily high-order derivatives discontinuous Galerkin approach [13, 25], gives an explicit local time-stepping method for elastic wave equations and the time-dependent Maxwell's equations. Later, [16] combined the time-stepping ideas in [13] to develop a predictor-corrector type Runge-Kutta based local time-stepping method.

Recently, [17] presented arbitrarily high-order Runge-Kutta based local time-stepping methods for wave phenomena. This method separates the computational mesh into two groups, coarse elements and fine elements. Each group is then advanced with its stable time step. Intermediate values needed at the coarse-fine interface are computed through a combination of interpolation and Taylor expansion.

In [22], a second order Runge-Kutta based local time-stepping method is developed. Each element is advanced with its maximum stable time-step and the interface between large and small elements is resolved using an accurate quadratic polynomial. This polynomial is used as a boundary condition for the small elements in a way which preserves the order of the underlying Runge-Kutta method. This method supports meshes with arbitrary levels of refinement.

Here, we follow a similar approach taken in [22] to develop local time-stepping methods based on a third order Runge-Kutta method and the classical fourth order Runge-Kutta

method. We store past information on the interface elements and use them to approximate the inner stages of the small elements to advance the large interface elements. Next we will create a polynomial approximation to the large interface elements to be used as a boundary condition to advance the small interface elements.

This thesis is organized as follows. In Chapter 2, we will give a brief introduction to explicit Runge-Kutta methods for solving first order ordinary differential equations. Next, in Chapter 3, we will introduce the discontinuous Galerkin finite element method for the spatial discretization of nonlinear hyperbolic conservation laws. We will show that this spatial discretization leads to a first order system of ordinary differential equations and discuss global stability restrictions. In Chapter 4, we will develop local time-stepping schemes for ordinary differential equations and prove that they preserve the accuracy of their Runge-Kutta counterparts. From here, we will discuss how these local time-stepping schemes can be utilized for the time integration of discontinuous Galerkin spatial discretization to alleviate the restrictive globally stable time-step. Finally, in Chapter 5, numerical experiments will illustrate the expected rate of convergence of our local time-stepping schemes from Chapter 4. We will present examples illustrating the efficiency of local time-stepping versus global time-stepping. Concluding remarks and areas for future work on this topic will be given in Chapter 6.

Chapter 2

Numerical Solution of Ordinary Differential Equations

2.1 Introduction

Ordinary differential equations have been studied since the development of calculus in the seventeenth century by Newton and Liebniz. In general, ordinary differential equations do not have closed form solutions, thus, numerical methods to approximate the solution are necessary.

Numerical methods for solving ordinary differential equations fall into two broad categories: "single-step" methods, also known as Runge-Kutta methods which use only the current approximation to advance the solution in time, and "multi-step" methods which use several previous values to advance the solution. In this chapter we briefly discuss Runge-Kutta methods. For a more detailed exposition of Runge-Kutta methods and other numerical techniques for solving ordinary differential equations, see [4, 18].

2.2 Runge-Kutta Methods

Consider the initial value problem

$$\begin{aligned} \frac{d}{dt}\mathbf{y} &= \mathbf{f}(\mathbf{y}, t), \quad t > t_0, \\ \mathbf{y}(t_0) &= \mathbf{y}_0, \end{aligned} \tag{2.1}$$

for a vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$ and $\mathbf{f} : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$.

To numerically approximate the solution of the system of ordinary differential equations (2.1) on the interval $[t_0, t_f]$ we first define our step size to be $h_{n+1} = t_{n+1} - t_n$. The general s-stage Runge-Kutta method for the ordinary differential equation system (2.1) can be written as

$$\mathbf{Y}_i = \mathbf{y}_n + h_{n+1} \sum_{j=1}^{s} a_{ij} \mathbf{f}(\mathbf{Y}_j, t_n + c_j h_{n+1}), \quad 1 \leq i \leq s, \tag{2.2}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_{n+1} \sum_{i=1}^{s} b_i \mathbf{f}(\mathbf{Y}_i, t_n + c_i h_{n+1}), \tag{2.3}$$

where the $\mathbf{Y}_i$'s are the intermediate approximations, or stages, to the solution at time $t_n + c_i h_{n+1}$. The method can be represented using the shorthand notation, called a Butcher Tableau,

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} .$$

Table 2.1: Butcher tableau for a general Runge-Kutta method.

The coefficients c_i must satisfy

$$c_i = \sum_{j=1}^{s} a_{ij}, \quad i = 1, \ldots, s, \tag{2.4}$$

a necessary condition for high-order methods [18].

The Runge-Kutta method is explicit if and only if $a_{ij} = 0$ for $j \geq i$ because then the stages $\mathbf{Y}_i$ in (2.2) are given in terms of previously computed stages. Methods with some $a_{ij} \neq 0, j \geq i$ are implicit. We will only consider explicit Runge-Kutta methods. Some examples of explicit Runge-Kutta methods [1] are given below.

The only one parameter family of second-order methods:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \alpha & \alpha & 0 \\ \hline & 1-\frac{1}{2\alpha} & \frac{1}{2\alpha} \end{array}.$$

Table 2.2: One parameter family of Runge-Kutta 2 methods

Popular Runge-Kutta 2 methods are the midpoint method given by $\alpha = 1/2$ and Heun's method given by $\alpha = 1$.

An example of a one-parameter family of a three-stage, third-order Runge-Kutta method is

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 & 0 \\ \frac{2}{3} & \frac{2}{3}-\frac{1}{4\alpha} & \frac{1}{4\alpha} & 0 \\ \hline & \frac{1}{4} & \frac{3}{4}-\alpha & \alpha \end{array}.$$

Table 2.3: A one parameter family of Runge-Kutta 3 methods.

where α is a parameter.

Finally, the classical fourth-order Runge-Kutta scheme is written as

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}.$$

Table 2.4: The classical fourth order Runge-Kutta method.

2.3 Error Estimation and Order Conditions

2.3.1 Consistency and Order Conditions

Suppose our goal is to create a Runge-Kutta method which has global error satisfying

$$\mathbf{y}(t_f) - \mathbf{y}_N = \mathcal{O}(h^p), \tag{2.5}$$

where t_f is our final time, $\mathbf{y}_N$ is the numerical solution at t_f and h is the maximum time-step size used in the computation. We will state necessary conditions on the coefficients of the Butcher tableau, Table 2.1, to achieve this level of accuracy.

From (2.3), we can see that any explicit Runge-Kutta method can be written as a single-step method

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h_{n+1}\boldsymbol{\psi}(\mathbf{y}_n, t; h_{n+1}), \quad h_{n+1} = t_{n+1} - t_n, \tag{2.6}$$

where $\boldsymbol{\psi}$ is given in terms of the right-hand side of the differential equation $\mathbf{f}$. We begin by defining consistency and order of consistency of a single-step method [21].

Definition 1. *For each $(\mathbf{y}, t) \in G \subset \mathbb{R}^m \times \mathbb{R}$ denote by $\boldsymbol{\eta} = \boldsymbol{\eta}(\xi)$ the unique solution to the initial value problem*

$$\boldsymbol{\eta}' = \mathbf{f}(\boldsymbol{\eta}, \xi), \quad \boldsymbol{\eta}(t) = \mathbf{y},$$

with initial data $(\mathbf{y}, t)$. Then

$$\Delta(\mathbf{y}, t; h) = \frac{1}{h}\left(\boldsymbol{\eta}(t+h) - \boldsymbol{\eta}(t)\right) - \boldsymbol{\psi}(\mathbf{y}, t; h)$$

is called the local discretization error. The single-step (Runge-Kutta) method is called consistent (with the initial value problem) if

$$\lim_{h \to 0} \Delta(\mathbf{y}, t; h) = \mathbf{0}$$

uniformly for all $(\mathbf{y}, t) \in G$, and it is said to have consistency order p if

$$\|\Delta(\mathbf{y}, t; h)\| \leq K h^p$$

for all $(\mathbf{y}, t) \in G$, all $h > 0$, and some constant K.

Finding the order of consistency of a given Runge-Kutta method is as simple as taking the Taylor expansion of the method (2.3) and subtracting the Taylor expansion of the exact solution. The first power of h with a nonzero coefficient gives us the order of consistency, provided the derivatives exist and are bounded. Determining the order of consistency for high-order Runge-Kutta methods becomes troublesome as the number of terms in the derivatives grows quickly. In a long series of papers by J.C. Butcher, starting in the 1960's, an elegant theory for determining this order was given. The theory involves using rooted trees to enumerate the components in the Taylor series. This is beyond the scope of this section but the details can be found in [4, 18].

In [1], necessary conditions on the coefficients of the Runge-Kutta method to be consistent of order p were derived, they are outlined below.

Let A be the $s \times s$ matrix whose entries are the coefficients of the tableau that defines the Runge-Kutta method. Let $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_s)^T$ be the vector of intermediate stages, $\mathbf{b} = (b_1, b_2, \ldots, b_s)^T$ be the vector of weights, $C = \text{diag}(c_1, c_2, \ldots, c_s)$ be the diagonal matrix with the c_j coefficients on its diagonal, and $\mathbf{1} = (1, 1, \ldots, 1)^T$. Note that when the method is explicit, A is lower triangular with zeros along the diagonal and so, $A^j = 0$ for all $j \geq s$.

For consistency of order p we require

$$\mathbf{b}^T A^k C^{l-1} \mathbf{1} = \frac{(l-1)!}{(l+k)!}, \quad 1 \leq l + k \leq p. \tag{2.7}$$

In particular, with $l = 1$, we have

$$\mathbf{b}^T A^{k-1} \mathbf{1} = \frac{1}{k!}, \quad k = 1, 2, \ldots, p. \tag{2.8}$$

In [4] the order conditions are used to prove that if an explicit s-stage Runge-Kutta method has order p, then $s \geq p$. It is also shown that if an explicit s-stage Runge-Kutta method has order $p \geq 5$, then $s > p$. Thus, the only explicit Runge-Kutta methods with $s = p$ have order $p = 1, 2, 3, 4$. In the table below, we give the highest order of accuracy for a method with a given number of stages [1],

Number of stages	1	2	3	4	5	6	7	8	9	10
Highest order	1	2	3	4	4	5	6	6	7	7

Table 2.5: Highest attainable order per number of stages.

2.3.2 Local Error

An important measure of error for a Runge-Kutta method is the local error. The local error of a method is the error made over one time step. We define this quantity as the difference between the numerical solution $\mathbf{y}_{n+1}$ and the exact solution $\tilde{\mathbf{y}}(t_{n+1})$ when solving the initial value problem

$$\frac{d}{dt}\tilde{\mathbf{y}} = \mathbf{f}(t, \tilde{\mathbf{y}}), \tag{2.9}$$

$$\tilde{\mathbf{y}}(t_n) = \mathbf{y}_n. \tag{2.10}$$

Then, the local error at time t_{n+1} is

$$\mathbf{l}_{n+1} = \tilde{\mathbf{y}}(t_{n+1}) - \mathbf{y}_{n+1}. \tag{2.11}$$

A rigorous local error bound for a Runge-Kutta method of order p is given in [18], and we see that if the method is order p, and $\mathbf{f} \in C^p$, then

$$\|\mathbf{l}_{n+1}\| = \|\tilde{\mathbf{y}}(t_{n+1}) - \mathbf{y}_{n+1}\| \leq Ch^{p+1}, \tag{2.12}$$

for some constant C.

It is interesting to not only analyze the local error of the Runge-Kutta method, but also to analyze the local error of the intermediate stages. We can find the local error by computing the Taylor expansions of the intermediate stages and comparing it to the Taylor expansion of the exact solution at that time. This illustrates how Runge-Kutta methods use the inner stages as low-order approximations to the solution and then cancel the low-order errors to create a high-order approximation to the exact solution.

As an example to show the canceling of low order errors, we analyze the local error of the Runge-Kutta method with the Butcher tableau given by Table 2.3 with $\alpha = 3/8$ for the autonomous scalar problem

$$\frac{d}{dt}y = f(y), \tag{2.13}$$

where f is a smooth function. This method has two intermediate stages at time $t = t_n + 2h/3$. The exact solution at this time has the Taylor expansion about $t = t_n$

$$\begin{aligned} y(t_{n+2/3}) &= y(t_n) + \frac{2}{3}hy' + \frac{2}{9}h^2y'' + \frac{4}{81}h^3y''' + \mathcal{O}(h^4), \\ &= y(t_n) + \frac{2}{3}hf + \frac{2}{9}h^2f_yf + \frac{4}{81}h^3(f_{yy}f^2 + f_y^2f) + \mathcal{O}(h^4), \end{aligned} \tag{2.14}$$

and the exact solution at time $t = t_{n+1}$ has the Taylor expansion about $t = t_n$

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + hy' + \frac{h^2}{2}y'' + \frac{h^3}{6}y''' + \frac{h^4}{24}y^{(4)} + \mathcal{O}(h^5), \\ &= y(t_n) + hf + \frac{h^2}{2}f_yf + \frac{h^3}{6}(f_{yy}f^2 + f_y^2f) \\ &\quad + \frac{h^4}{24}(f_{yyy}f^3 + 4f_{yy}f_yf^2 + f_y^3f) + \mathcal{O}(h^5), \end{aligned} \tag{2.15}$$

where the derivatives of y are evaluated at $t = t_n$ and the derivatives of f are evaluated at $y(t_n)$.

The first stage is

$$Y^{(1)} = y_n + \frac{2}{3}hf(y_n). \tag{2.16}$$

If we assume that the numerical solution is exact at time t_n, that is $y_n = y(t_n)$, then

$$y(t_{n+2/3}) - Y^{(1)} = \frac{2}{9}h^2 f_y f + \mathcal{O}(h^3). \tag{2.17}$$

Comparing this with (2.14), we see that the local error of the first stage is $\mathcal{O}(h^2)$. Next, we expand $f(Y^{(1)})$ around the point y_n. The second stage has the Taylor expansion

$$\begin{aligned} Y^{(2)} &= y_n + \frac{2}{3}hf(Y^{(1)}) \\ &= y_n + \frac{2}{3}hf\left(y_n + \frac{2}{3}hf(y_n)\right) \\ &= y_n + \frac{2}{3}hf + \frac{4}{9}h^2 f_y f + \frac{4}{27}h^3 f_{yy} f^2 + \mathcal{O}(h^4) \end{aligned} \tag{2.18}$$

Again, assuming $y_n = y(t_n)$,

$$y(t_{n+2/3}) - Y^{(2)} = -\frac{2}{9}h^2 f_y f + \mathcal{O}(h^3). \tag{2.19}$$

Comparing this with (2.14), we see the local error of the second stage is also $\mathcal{O}(h^2)$.

To investigate the local error of the numerical solution y_{n+1}, we expand each of the following expressions in a Taylor series around y_n

$$f(Y^{(1)}) = f + \frac{2}{3}hf_y f + \frac{2}{9}h^2 f_{yy} f^2 + \frac{4}{81}h^3 f_{yyy} f^3 + \frac{1}{243}h^4 f_{yyyy} f^4 + \mathcal{O}(h^5), \tag{2.20}$$

$$\begin{aligned} f(Y^{(2)}) = f + \frac{2}{3}hf_y f + \frac{2}{9}h^2\left(f_{yy}f^2 + 2f_y^2 f\right) + \frac{4}{81}h^3\left(f_{yyy}f^3 + 9f_{yy}f_y f^2\right) \\ + \frac{2}{243}h^4\left(f_{yyyy}f^4 + 16f_{yyy}f_y f^3 + 13f_{yy}f_y^2 f^2 + 12f_{yy}^2 f^3\right) + \mathcal{O}(h^5). \end{aligned} \tag{2.21}$$

The solution is updated in time using

$$y_{n+1} = y_n + \frac{h}{4}\left(f + \frac{3}{2}f(Y^{(1)}) + \frac{3}{2}f(Y^{(2)})\right). \tag{2.22}$$

Using (2.15), (2.20), and (2.21), we have

$$y(t_{n+1}) - y_{n+1} = \frac{h^4}{24}\left(\frac{3}{27}f_{yyy}f^3 + f_y^3 f\right) + \mathcal{O}(h^5). \tag{2.23}$$

Thus, the local error of this Runge-Kutta 3 method is $\mathcal{O}(h^4)$, while the inner stages are only $\mathcal{O}(h^2)$ accurate.

2.3.3 Global Error and Convergence

Any practical numerical method needs a guarantee that the numerical solution will converge to the unique solution. The global error of the method is the error of the numerical solution after several time steps

$$\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n. \tag{2.24}$$

We are particularly interested in estimating the global error at the final time

$$\mathbf{e}_N = \mathbf{y}(t_f) - \mathbf{y}_N. \tag{2.25}$$

We state that under certain conditions consistency is equivalent to convergence. From [21], we have the following theorem.

Theorem 1. *Assume that the function $\boldsymbol{\psi}$ describing the single-step (Runge-Kutta) method is continuous in all variables and satisfies a Lipschitz condition in the first variable; i.e.,*

$$\|\boldsymbol{\psi}(\mathbf{y}, t; h) - \boldsymbol{\psi}(\mathbf{w}, t; h)\| \leq K\|\mathbf{y} - \mathbf{w}\|$$

for all $(\mathbf{y}, t), (\mathbf{w}, t) \in G \subset \mathbb{R}^m \times \mathbb{R}$, all (sufficiently small) h, and a Lipschitz constant K. Then the single-step (Runge-Kutta) method is convergent if and only if it is consistent.

We see in Theorem 1 the need for a "sufficiently small" step size h; this is the topic of discussion in Section 2.4.

To get the exact value of the global error requires knowledge of the exact solution of the differential equation. For most problems, a closed form solution is not available; thus, there is no way of computing the exact global error. To estimate the global error, we notice that local errors are propagated along each time step through out the computation. The following theorem, from [18], allows us to bound the global error at time t_f based on the local error estimation. We restate the theorem below.

Theorem 2. *Let U be a neighborhood of $\{(\mathbf{y}(t), t) : t_0 \leq t \leq t_f\}$ where $\mathbf{y}(t)$ is the exact solution of (2.1). Suppose that in U*

$$\left\| \frac{\partial f}{\partial y} \right\| \leq L, \tag{2.26}$$

and that the local error estimates $\|\mathbf{l}_n\| \leq C h_{n-1}^{p+1}$ are valid in U. Then the global error (2.25) can be estimated by

$$\|\mathbf{e}_N\| \leq h^p \frac{C'}{L} \left(\exp(L(t_f - t_0)) - 1\right) \tag{2.27}$$

where $h = \max h_i$,

$$C' = \begin{cases} C & L \geq 0 \\ C\exp(-Lh) & L < 0, \end{cases}$$

and h *is small enough for the numerical solution to remain in* U.

It is useful to note that for the Runge-Kutta methods discussed above

$$h_{n+1}\|\Delta(\mathbf{y}, t_n; h_{n+1})\| = \|\mathbf{1}_{n+1}\|(1 + \mathcal{O}(h_{n+1})), \tag{2.28}$$

i.e., the local error is one order higher than the local discretization error. Thus, if we have consistency of order p, we have convergence of order p.

2.4 Region of Absolute Stability

To use a Runge-Kutta method to advance our solution in time we need a way to choose a suitable time step $h_{n+1} = t_{n+1} - t_n$. We will see below that the time step cannot be chosen arbitrarily and the size of our time step depends on the method we use. To find conditions on our time step, we analyze how the numerical solution behaves when applied to the test equation

$$y' = \lambda y. \tag{2.29}$$

When $\mathrm{Re}(\lambda) > 0$ the modulus of the exact solution will grow exponentially with time, $|y(t_{n+1})| > |y(t_n)|$. Hence, the exact solution is unstable and we will not be able to to determine if the numerical solution is stable. When $\mathrm{Re}(\lambda) = 0$ the exact solution will oscillate for all time. Finally, when $\mathrm{Re}(\lambda) < 0$ the modulus of the exact solution will decay exponentially in time $|y(t_{n+1})| < |y(t_n)|$, so we require the time step be chosen so that the numerical solution does not grow with time, i.e., $|y_{n+1}| \leq |y_n|$.

The region of absolute stability for a Runge-Kutta method is the region in the complex plane such that applying the method to the test equation (2.29) with $z = h\lambda$ from within this region, gives a numerical solution satisfying $|y_{n+1}| \leq |y_n|$. The region is taken in the complex plane since in general, $\lambda \in \mathbb{C}$.

To determine the region of absolute stability, we begin by rewriting the inner stages (2.2) for the test equation (2.29) as

$$\mathbf{Y} = y_n\mathbf{1} + h\lambda A\mathbf{Y}, \tag{2.30}$$

$$\mathbf{Y} = y_n(I - zA)^{-1}\mathbf{1}. \tag{2.31}$$

Next, we can rewrite (2.3) as

$$y_{n+1} = y_n + h\lambda \mathbf{b}^T \mathbf{Y}, \tag{2.32}$$
$$= y_n + y_n z \mathbf{b}^T (I - zA)^{-1} \mathbf{1}, \tag{2.33}$$
$$= y_n \left(1 + z\mathbf{b}^T (I - zA)^{-1} \mathbf{1}\right), \tag{2.34}$$
$$= y_n \left(1 + z\mathbf{b}^T \left(I + \sum_{i=1}^{\infty} z^i A^i\right) \mathbf{1}\right). \tag{2.35}$$

Substituting (2.8) into (2.35), and using the fact $A^j = 0$ for all $j \geq s$ for explicit Runge-Kutta methods, we have

$$y_{n+1} = \left(1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \mathbf{b}^T A^{j-1} \mathbf{1}\right) y_n. \tag{2.36}$$

To satisfy $|y_{n+1}| \leq |y_n|$, we require

$$\left|1 + z + \frac{z^2}{2} + \cdots + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \mathbf{b}^T A^{j-1} \mathbf{1}\right| \leq 1 \tag{2.37}$$

This defines a polynomial in the complex variable z. Solving (2.37), we can find a region in the complex plane so that if $z = h\lambda$ lies in this region, the Runge-Kutta method will be stable. In Figure 2.1, we plot the regions of absolute stability for the Runge-Kutta methods of orders $p = 1, 2, 3, 4$.

In the inequality (2.37), we observe that Runge-Kutta methods with $s = p$ will have the same region of absolute stability, but when $s \geq p$ the region is determined by the methods coefficients. We see that we can create different regions of absolute stability, possibly increasing their size, by adding more stages to the method and altering the coefficients of the Butcher tableau.

Consider now the linear system of ordinary differential equations

$$\frac{d}{dt}\mathbf{y} = A\mathbf{y}, \tag{2.38}$$

where A is a constant, diagonalizable, $m \times m$ matrix, and $\mathbf{y} = (y_1, y_2, \ldots, y_m)^T$.

Let $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ be the diagonal matrix composed of the eigenvalues λ_j of A. Since A is diagonalizable, let Q be the matrix of eigenvectors of A so that

$$Q^{-1} A Q = \Lambda. \tag{2.39}$$

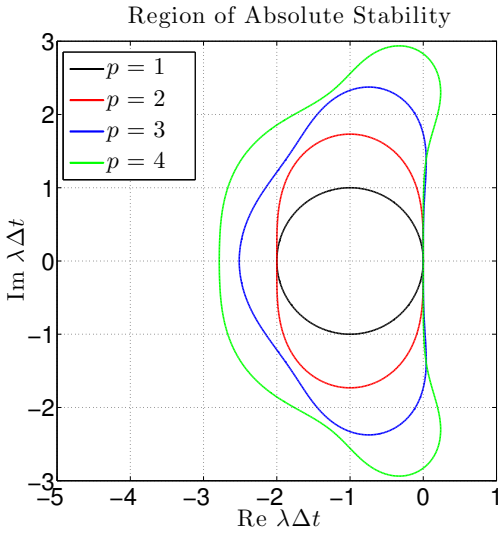


Figure 2.1: Boundaries of Regions of Absolute Stability

Define $\mathbf{w} = Q^{-1}\mathbf{y}$. Multiplying (2.38) on the left by Q^{-1} and noticing that $Q\mathbf{w} = \mathbf{y}$, we get the system of differential equations

$$\frac{d}{dt}\mathbf{w} = \Lambda\mathbf{w}, \tag{2.40}$$

where the system has now been written as m independent scalar differential equations. To ensure stability of the Runge-Kutta method, we have to choose the time step so that each $h\lambda_j, j = 1, 2, \ldots, m$ is within the region of absolute stability. When we choose our time step this way, we have

$$|\mathbf{w}_n| \leq |\mathbf{w}_{n-1}| \leq \cdots \leq |\mathbf{w}_0|. \tag{2.41}$$

Transforming back into the variable $\mathbf{y}$, we get

$$|\mathbf{y}_n| \leq \|Q\||\mathbf{w}_n| \leq \cdots \leq \|Q\||\mathbf{w}_0| \leq \|Q\|\|Q^{-1}\||\mathbf{y}_0|, \tag{2.42}$$

where $\|\cdot\|$ is the matrix norm induced by the vector norm $|\cdot|$. We notice that $\|Q\|\|Q^{-1}\|$ is the condition number of the matrix Q. From here, we see that if $\|Q\|\|Q^{-1}\|$ is not large, or if $\|Q\|\|Q^{-1}\| \leq 1$, we can use the eigenvalues of A to determine a stable time step.

Chapter 3

The Discontinuous Galerkin Finite Element Method

3.1 Introduction

The discontinuous Galerkin methods are a class of numerical methods used for solving differential equations. It was originally introduced by Reed and Hill in 1973 for solving the neutron transport equation [24], then developed by Cockburn and Shu in a series of papers [9, 8, 7, 6, 10] in the context of nonlinear hyperbolic conservation laws. Later, it was extended to parabolic and elliptic partial differential equations. The discontinuous Galerkin method has seen applications in gas and fluid dynamics, acoustics, electromagnetics, geophysics, and many more topics. A more detailed history of the method is given in [19].

Below, we introduce the discontinuous Galerkin method for nonlinear hyperbolic conservation laws, as developed by Cockburn and Shu.

3.2 One-Dimensional Scalar Equations

In this section, we present the discontinuous Galerkin method for one-dimensional scalar hyperbolic conservation laws. Consider the scalar hyperbolic conservation law

$$u_t + f(u)_x = 0, \quad a < x < b, \quad t > 0, \tag{3.1}$$

$$u(x,0) = u_0(x), \quad a \leq x \leq b, \tag{3.2}$$

with appropriate boundary conditions.

Begin by subdividing the domain into non-overlapping elements $I_j = [x_j, x_{j+1}]$ with element size $h_j = x_{j+1} - x_j$, $j = 1, \ldots, N$, such that

$$[a, b] = \bigcup_{j=1}^{N} I_j.$$

To construct the discontinuous Galerkin formulation of [8] on element j, we multiply (3.1) by a test function from the Sobolev space $v \in \mathcal{H}^1(x_j, x_{j+1})$ and integrate over the interval $[x_j, x_{j+1}]$

$$\int_{x_j}^{x_{j+1}} u_t v dx + \int_{x_j}^{x_{j+1}} f(u)_x v dx = 0. \tag{3.3}$$

Integrating by parts yields

$$\int_{x_j}^{x_{j+1}} u_t v dx + f(u)v\Big|_{x_j}^{x_{j+1}} - \int_{x_j}^{x_{j+1}} f(u)v' dx = 0. \tag{3.4}$$

Next, we approximate u on I_j by $U_j \in \mathcal{S}$, where $\mathcal{S}$ is a finite-dimensional subspace of $\mathcal{H}^1(x_j, x_{j+1})$. We choose to approximate our test functions v on I_j by $V \in \mathcal{S}$, where $\mathcal{S}$ is the same finite dimensional subspace as our solution. This is known as the Galerkin formulation. We obtain

$$\int_{x_j}^{x_{j+1}} \frac{d}{dt}(U_j) V dx + f(U_j)V\Big|_{x_j}^{x_{j+1}} - \int_{x_j}^{x_{j+1}} f(U_j)V' dx = 0. \tag{3.5}$$

To choose a basis for $\mathcal{S}$, we map the physical element $I_j = [x_j, x_{j+1}]$ to the canonical element $I_0 = [-1, 1]$ using the mapping

$$x(\xi) = \frac{1-\xi}{2} x_j + \frac{1+\xi}{2} x_{j+1}, \text{ and } \frac{dx}{d\xi} = \frac{x_{j+1} - x_j}{2} = \frac{h_j}{2}. \tag{3.6}$$

Changing variables, (3.5) becomes

$$\frac{h_j}{2} \int_{-1}^{1} \frac{d}{dt} U_j(t, x(\xi)) V(x(\xi)) d\xi + f(U_j(t, x(\xi)))V(x(\xi))\Big|_{-1}^{1} - \int_{-1}^{1} f(U_j(t, x(\xi))) \frac{d}{d\xi} V(x(\xi)) d\xi = 0, \tag{3.7}$$

or, more compactly written

$$\frac{h_j}{2} \int_{-1}^{1} \frac{d}{dt}(U_j) V d\xi + f(U_j)V\Big|_{-1}^{1} - \int_{-1}^{1} f(U_j)V' d\xi = 0, \tag{3.8}$$

where the derivative in V' is understood to be taken with respect to ξ.

Next, we are tasked with choosing an appropriate finite dimensional subspace. We choose $\mathcal{S} = \mathcal{S}^p$, the space of polynomials of degree less than or equal to p. One choice of basis functions for this subspace is $\Phi = \{1, x, x^2, \ldots, x^p\}$. Although this is a valid choice for a basis, it is known to be ill-conditioned for large p. Instead, we choose the set of Legendre polynomials $P_k, k = 0, 1, \ldots, p$. The Legendre polynomials are defined using the recursion relation

$$P_0(\xi) = 1, \tag{3.9a}$$

$$P_1(\xi) = \xi, \tag{3.9b}$$

$$P_n(\xi) = \frac{2n-1}{n}\xi P_{n-1}(\xi) - \frac{n-1}{n}P_{n-2}(\xi), \quad n \geq 2. \tag{3.9c}$$

The Legendre polynomials form an orthogonal basis on $[-1, 1]$,

$$\int_{-1}^{1} P_k P_l d\xi = \frac{2}{2k+1}\delta_{kl}, \tag{3.10}$$

where δ_{kl} is the Kronecker delta. This basis does not form an orthonormal set because the normalization of (3.9) was chosen to satisfy

$$P_k(1) = 1. \tag{3.11}$$

We note that with (3.11), we have at the other endpoint $P_k(-1) = (-1)^k$. In Figure 3.1, we show plots of the Legendre polynomials $P_i(\xi), i = 1, 2, \ldots, 5$.

We write the numerical solution as a linear combination of basis functions

$$U_j = \sum_{i=0}^{p} c_{ij} P_i, \tag{3.12}$$

where the coefficients c_{ij} are functions of time t.

At each point x_j the global numerical solution $U(x_j)$ is given by the left element I_{j-1} and the right element I_j. The global numerical solution U is not well defined at these points since we do not enforce continuity between elements and so, we are not guaranteed that $U_{j-1}(x_j) = U_j(x_j)$. To resolve this problem, we find the value of $U(x_j)$ by solving the local Riemann problem. Its solution U_j^* is called a Riemann state and can be determined using a Riemann solver. Having a value U_j^*, we can directly evaluate $f(U_j^*)$ in (3.8). Exact Riemann solvers for many problems are known but are costly to compute [26].

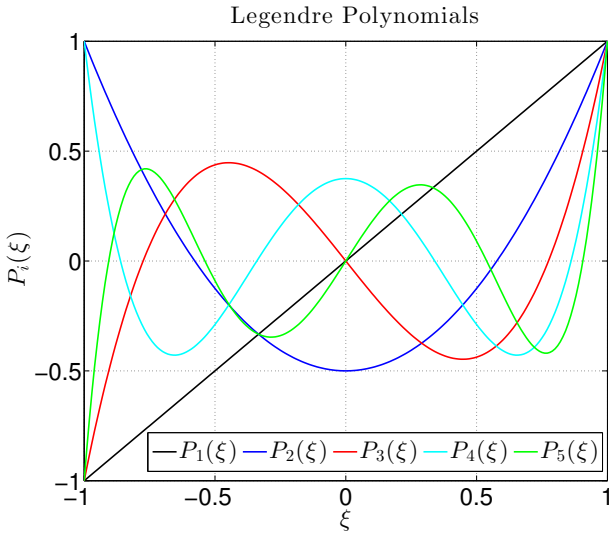


Figure 3.1: Legendre polynomials $P_i(\xi), i = 1, 2, \ldots, 5$

Alternatively, an approximate Riemann solver can be used to approximate $f(U_j^*)$ at each element interface. This is the approach we will use. To do so, we use the local Lax-Friedrich flux

$$f(U_j^*) = F_n(U_{j-1}, U_j) = \frac{1}{2}\left(f(U_j(x_j)) + f(U_{j-1}(x_j))\right) - \frac{\lambda_j}{2}(U_j(x_j) - U_{j-1}(x_j)), \tag{3.13}$$

where $\lambda_j = \max\{|f'(U_j(x_j))|, |f'(U_{j-1}(x_j))|\}$.

Choosing our test functions $V = P_k, k = 0, 1, \ldots, p$, substituting (3.12) into (3.8) and using (3.10), we obtain an ordinary differential equation for each coefficient on element I_j

$$\frac{d}{dt}c_{kj} = -\frac{2k+1}{h_j}\left[F_n(U_j, U_{j+1}) - (-1)^k F_n(U_{j-1}, U_j)\right] + \frac{2k+1}{h_j}\int_{-1}^{1} f(U_j)P_k' d\xi, \tag{3.14}$$

$k = 0, 1, \ldots, p$. Combining all coefficients over all elements, we obtain a vector of solution coefficients $\mathbf{c}$. Thus, the discontinuous Galerkin method can be written as a system of ordinary differential equations

$$\frac{d}{dt}\mathbf{c} = \mathbf{f}(\mathbf{c}), \tag{3.15}$$

where $\mathbf{f}$ is a vector function defined by the right hand side of (3.14).

3.3 One-Dimensional Systems of Equations

The derivation of the discontinuous Galerkin method for one-dimensional hyperbolic systems of conservation laws is analogous to that of the scalar case. Consider the hyperbolic system

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = 0, \quad a < x < b, \quad t > 0, \tag{3.16}$$

$$\mathbf{u}(x,0) = \mathbf{u}_0(x), \quad a \leq x \leq b, \tag{3.17}$$

with appropriate boundary conditions, where $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ is a vector of n variables. Following the same procedure as in the previous section, we multiply (3.16) by a test function v and integrate by parts. Mapping element I_j to the canonical element I_0 and approximating the exact solution by a vector of polynomials, we arrive at

$$\frac{h_j}{2}\int_{-1}^{1}\frac{d}{dt}(\mathbf{U}_j)V d\xi + \mathbf{f}(\mathbf{U}_j)V\Big|_{-1}^{1} - \int_{-1}^{1}\mathbf{f}(\mathbf{U}_j)V' d\xi = 0, \tag{3.18}$$

where the derivative in V' is understood to be taken with respect to ξ. Again, choosing our polynomial basis to be the set of Legendre polynomials of degree up to p, and choosing our test functions to be these basis functions $V = P_k, k = 0, 1, \ldots, p$ we arrive at

$$\frac{d}{dt}\mathbf{c}_{kj} = -\frac{2k+1}{h_j}\left[\mathbf{F}_n(\mathbf{U}_j, \mathbf{U}_{j+1}) - (-1)^k\mathbf{F}_n(\mathbf{U}_{j-1}, \mathbf{U}_j)\right] + \frac{2k+1}{h_j}\int_{-1}^{1}\mathbf{f}(\mathbf{U}_j)P_k' d\xi, \tag{3.19}$$

$k = 0, 1, \ldots, p$. For one-dimensional systems, we choose our numerical flux to be the local Lax-Friedrichs flux

$$\mathbf{F}_n(\mathbf{U}_{j-1}, \mathbf{U}_j) = \frac{1}{2}\left(\mathbf{f}(\mathbf{U}_j(x_j)) + \mathbf{f}(\mathbf{U}_{j-1}(x_j))\right) - \frac{\lambda_j}{2}(\mathbf{U}_j(x_j) - \mathbf{U}_{j-1}(x_j)), \tag{3.20}$$

where λ_j is the maximum of the spectral radius of $\mathbf{f}(\mathbf{U}_j(x_j))_{\mathbf{u}}$ and $\mathbf{f}(\mathbf{U}_{j-1}(x_j))_{\mathbf{u}}$. Since the system (3.16) is hyperbolic, the eigenvalues of the Jacobian matrix $\mathbf{f}(\mathbf{u})_{\mathbf{u}}$ are always real [23].

Combining (3.19) with the numerical flux (3.20), the discontinuous Galerkin method for one-dimensional systems can written as a system of ordinary differential equations

$$\frac{d}{dt}\mathbf{c} = \mathbf{f}(\mathbf{c}), \tag{3.21}$$

where $\mathbf{c}$ is a vector of solution coefficients and $\mathbf{f}$ is a vector function defined by the right hand side of (3.19).

3.4 Two-Dimensional Systems of Equations

We now consider the two-dimensional system of hyperbolic conservation laws

$$\mathbf{u}_t + \nabla \cdot \mathbf{F}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{x} \in \Omega, \quad t > 0, \tag{3.22}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{3.23}$$

with suitable boundary conditions. Here, $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ is a vector of n variables in the computational domain $\Omega \subset \mathbb{R}^2$ and $\mathbf{F} = [\mathbf{F}_1, \mathbf{F}_2]$ where $\mathbf{F}_1$ and $\mathbf{F}_2$ are fluxes in the x and y direction, respectively.

We begin by partitioning our domain Ω into a mesh of non-overlapping triangles Ω_j, such that

$$\Omega = \bigcup_{j=1}^{N} \Omega_j.$$

Proceeding as in the one-dimensional case, we multiply (3.22) by a test function $v \in \mathcal{H}^1(\Omega_j)$ and integrate over Ω_j

$$\iint_{\Omega_j} \mathbf{u}_t v d\mathbf{x} + \iint_{\Omega_j} \nabla \cdot \mathbf{F}(\mathbf{u}) v d\mathbf{x} = \mathbf{0}. \tag{3.24}$$

We would like to apply the Divergence Theorem to the second integral. To do so, we use the identity

$$\nabla \cdot (\mathbf{F}(\mathbf{u})v) = \nabla v \cdot \mathbf{F}(\mathbf{u}) + v \nabla \cdot \mathbf{F}(\mathbf{u}). \tag{3.25}$$

Substituting (3.25) into (3.24) we have

$$\iint_{\Omega_j} \mathbf{u}_t v d\mathbf{x} + \iint_{\Omega_j} \nabla \cdot (\mathbf{F}(\mathbf{u})v) d\mathbf{x} - \iint_{\Omega_j} \nabla v \cdot \mathbf{F}(\mathbf{u}) d\mathbf{x} = \mathbf{0}. \tag{3.26}$$

Applying the Divergence Theorem to the second integral in (3.26) we get

$$\iint_{\Omega_j} \mathbf{u}_t v d\mathbf{x} + \int_{\partial\Omega_j} (\mathbf{F}(\mathbf{u})v) \cdot \mathbf{n}_j d\tau - \iint_{\Omega_j} \nabla v \cdot \mathbf{F}(\mathbf{u}) d\mathbf{x} = \mathbf{0}, \tag{3.27}$$

where $\mathbf{n}_j$ is the outward facing normal unit vector along each of element j's edges.

Next, we approximate the solution $\mathbf{u}$ on Ω_j by $\mathbf{U}_j \in \mathcal{S}$, where $\mathcal{S}$ is a finite-dimensional subspace of $\mathcal{H}^1(\Omega_j)$. Similarly, we choose to approximate the test functions from the same subspace. As in one-dimension, the global solution $\mathbf{U}$ is not well defined along the element boundaries and hence, along the element boundaries we solve the local Riemann problem.

Using an approximate Riemann solver, we approximate the value $\mathbf{F}(\mathbf{U}_j^*)$. Let $\mathbf{U}_{j+}$ denote the value of the numerical solution on the outside of Ω_j along its boundary $\partial\Omega_j$. Then, our approximate Riemann value, given by the local Lax-Friedrichs flux, is computed as

$$\mathbf{n}\cdot\mathbf{F}(\mathbf{U}_j^*) = \mathbf{F}_n(\mathbf{U}_j,\mathbf{U}_{j+}) = \frac{1}{2}\mathbf{n}\cdot(\mathbf{F}(\mathbf{U}_j)+\mathbf{F}(\mathbf{U}_{j+})) - \frac{\lambda_j}{2}(\mathbf{U}_{j+}-\mathbf{U}_j), \tag{3.28}$$

where λ_j is the the maximum of the spectral radius of the matrices

$$n_x(\mathbf{F}_1(\mathbf{U}_j))_{\mathbf{u}} + n_y(\mathbf{F}_2(\mathbf{U}_j))_{\mathbf{u}}, \text{ and } n_x(\mathbf{F}_1(\mathbf{U}_{j+}))_{\mathbf{u}} + n_y(\mathbf{F}_2(\mathbf{U}_{j+}))_{\mathbf{u}}$$

computed at each integration point along the element edge, see Section 3.5.

To simplify computations, we will map each element Ω_j to the canonical element Ω_0, which is the triangle with coordinates $(0,0),(1,0),(0,1)$. This mapping shown in Figure 3.2, maps each vertex $(x,y)\in\Omega_j$ to a new vertex $(\xi,\eta)\in\Omega_0$. The mapping is given by

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \xi \\ \eta \\ 1-\xi-\eta \end{pmatrix}, \tag{3.29}$$

where the coordinates $(x_i,y_i), i=1,2,3$ are the vertices of the given element. The Jacobian of this mapping is a constant matrix and is given by

$$J_j = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} = \begin{pmatrix} x_2-x_1 & x_3-x_1 \\ y_2-y_1 & y_3-y_1 \end{pmatrix}. \tag{3.30}$$

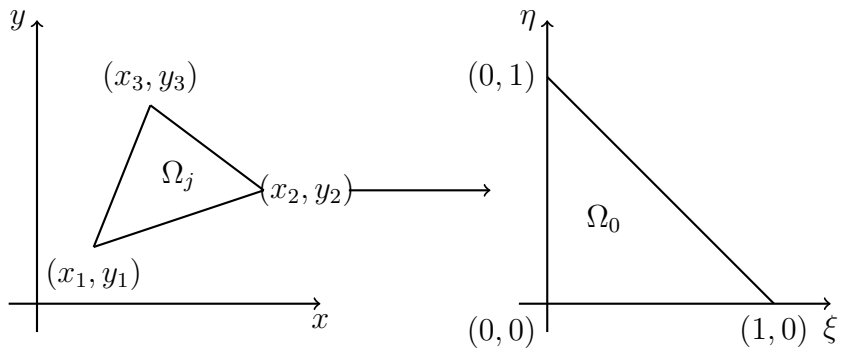


Figure 3.2: Mapping of element Ω_j to the canonical element Ω_0.

Applying the change of variables (3.29) to the first integral in (3.27) with numerical solution $\mathbf{U}_j$, we get

$$\frac{d}{dt}\iint_{\Omega_j}\mathbf{U}_j(x,y)V(x,y)dxdy=\frac{d}{dt}\iint_{\Omega_0}\mathbf{U}_j(x(\xi,\eta),y(\xi,\eta))V(x(\xi,\eta),y(\xi,\eta))|\det J_j|d\xi d\eta. \tag{3.31}$$

Next we apply the change of variables (3.29) to the volume integral

$$\iint_{\Omega_j}\nabla V\cdot\mathbf{F}(\mathbf{U}_j)dxdy. \tag{3.32}$$

Under this transformation we have

$$\begin{pmatrix}V_x\\V_y\end{pmatrix}=\begin{pmatrix}\xi_x&\eta_x\\\xi_y&\eta_y\end{pmatrix}\begin{pmatrix}V_\xi\\V_\eta\end{pmatrix}, \tag{3.33}$$

or in shorthand

$$\nabla_{xy}V=J_j^{-1}\nabla_{\xi\eta}V. \tag{3.34}$$

Therefore, under the change of variables

$$\iint_{\Omega_j}\nabla_{xy}V\cdot\mathbf{F}(\mathbf{U}_j)dxdy=\iint_{\Omega_0}J_j^{-1}\nabla_{\xi\eta}V\cdot\mathbf{F}(\mathbf{U}_j)|\det J_j|d\xi d\eta, \tag{3.35}$$

where on the left side of the equality V and $\mathbf{U}_j$ are functions of (x,y), and on the right side V and $\mathbf{U}_j$ are understood to be functions of $(x(\xi,\eta),y(\xi,\eta))$.

Finally, for the last integral in (3.27), known as the surface integral, we will map each side of element Ω_j to the canonical interval $I_0=[-1,1]$. This mapping is given by

$$\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}x_1&x_2\\y_1&y_2\end{pmatrix}\begin{pmatrix}(1-\xi)/2\\(1+\xi)/2\end{pmatrix}, \tag{3.36}$$

where (x_i,y_i) are the endpoints of the edge and $\xi\in I_0$. The chain rule gives, for each surface integral

$$d\tau=\frac{1}{2}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}d\xi, \tag{3.37}$$

which is simply half the length of each edge. Thus, for each edge of element Ω_j, we define the constant

$$l_{j,q}=\frac{1}{2}\sqrt{(x_2-x_1)^2+(y_2-y_1)^2},\quad q=1,2,3. \tag{3.38}$$

Applying these transformations to (3.27), our numerical solution must satisfy

$$|\det J_j| \iint_{\Omega_0} \frac{d}{dt}(\mathbf{U}_j) V d\xi d\eta - |\det J_j| \iint_{\Omega_0} J_j^{-1} \nabla V \cdot \mathbf{F}(\mathbf{U}_j) d\xi d\eta + \sum_{q=1}^{3} \int_{-1}^{1} (\mathbf{F}(\mathbf{U}_j^*) V) \cdot \mathbf{n}_{j,q} l_{j,q} d\xi = \mathbf{0}, \tag{3.39}$$

where $\mathbf{U}_j$ and V are understood to be functions of ξ, η and ∇ is an operator in (ξ, η)-space.

For our finite dimensional subspace $\mathcal{S}$, we choose $\mathcal{S}^p = \text{span}\{\xi^i \eta^j | 0 \leq i + j \leq p\}$, the space of polynomials degree p or less. This subspace has

$$N_p = \frac{(p+1)(p+2)}{2} \tag{3.40}$$

basis functions. We choose an orthogonal basis [12] on Ω_0, given by

$$\psi_i^k(\xi, \eta) = \sqrt{(2i+1)(2k+2)} P_{k-i}^{0,2i+1}(1-2\xi) P_i \left(1 - \frac{2\eta}{1-\xi}\right) (1-\xi)^i, \tag{3.41}$$

for $k = 0, 1, \ldots, p, i = 0, 1, \ldots, k$. Here, $P_{k-i}^{(0,2i+1)}$ are Jacobi polynomials of degree $k-i$ and P_i are Legendre polynomials of degree i. These basis functions satisfy

$$\iint_{\Omega_0} \psi_i^k \psi_m^l d\xi d\eta = \delta_{kl} \delta_{im}, \tag{3.42}$$

where δ_{kl} is the Kronecker delta function.

The numerical solution $\mathbf{U}_j$ can be written in terms of this basis as

$$\mathbf{U}_j = \sum_{m=0}^{p} \sum_{l=0}^{m} \mathbf{c}_{jml} \psi_l^m(\xi, \eta). \tag{3.43}$$

Choosing $V = \psi_i^k$ for $k = 0, 1, \ldots, p$ and $i = 0, 1, \ldots, k$, substituting (3.43) into (3.39) and using the orthogonality condition (3.42), we obtain N_p equations for the solution coefficients on element j

$$\frac{d}{dt}\mathbf{c}_{jki} = \iint_{\Omega_0} J_j^{-1} \nabla \psi_i^k \cdot \mathbf{F}(\mathbf{U}_j) d\xi d\eta - \frac{1}{|\det J_j|} \sum_{q=1}^{3} \int_{-1}^{1} (\mathbf{F}(\mathbf{U}_j^*) \psi_i^k) \cdot \mathbf{n}_{j,q} l_{j,q} d\xi. \tag{3.44}$$

Writing $\mathbf{c}$ as a vector of the coefficients over the entire domain, the discontinuous Galerkin method can be written as a system of ordinary differential equations

$$\frac{d}{dt}\mathbf{c} = \mathbf{f}(\mathbf{c}), \tag{3.45}$$

where $\mathbf{f}$ is a vector function defined by the right hand side of (3.44).

3.5 Time-Stepping and the CFL Condition

To fully discretize our partial differential equation in space, we need a rule for approximating the volume integral of (3.14) and (3.19), and the volume and surface integrals of (3.44).

Suppose we approximate our solution using up to degree p polynomials. Then, for the one-dimensional volume integral of (3.14) we have that P_k' is at most degree $p-1$ and U_j is at most of degree p. If we have a linear flux, i.e., $f(u) = au$, we will need to integrate a polynomial of at most degree $2p-1$. We would like the numerical quadrature to be as accurate as possible, so we choose a quadrature rule to integrate polynomials of degree $2p-1$ exactly. In [8], it was stated that for nonlinear flux we increase the order of the quadrature rule by one, so we must integrate with a quadrature rule of order $2p$. For the quadrature rule, we use the Gauss-Legendre quadrature with $n = p+1$ integration points [8].

Similarly in two-dimensions, if we approximate our solution using up to degree p polynomials, the surface integral contribution of (3.44) needs a one-dimensional quadrature rule to integrate up to degree $2p+1$ polynomials exactly. We again use the Gauss-Legendre quadrature with $n = p+1$ points. For the volume integral contribution, we choose an integration rule to integrate polynomials of degree up to $2p$ exactly. There are many choices for integration rules for triangles, we use rules listed in [14]. In Table (3.1), we show the number N of integration points on the triangle for a quadrature of degree q.

q	0	1	2	3	4	5	6	7	8	9	10
N	0	1	3	4	6	7	12	13	16	19	25

Table 3.1: Number N of integration points for quadrature of degree q on the canonical triangle.

Combining (3.14), (3.19) and (3.44) with their appropriate quadrature rules, we have a full discretization in space. We then need to advance the system of ordinary differential equations

$$\frac{d}{dt}\mathbf{c} = \mathbf{f}(\mathbf{c}), \tag{3.46}$$

in time, where $\mathbf{f}(\mathbf{c})$ is given by the right hand side of (3.14), (3.19) or (3.44). Since we are concerned with solving hyperbolic conservation laws, we will use an explicit time integration scheme. The most popular methods for solving (3.46), which we use in this thesis, are explicit Runge-Kutta methods [19] as outlined in Chapter 2.

With any time-stepping scheme, we need a way to choose a suitable Δt to advance the solution from t_n to t_{n+1}. In one-dimension, when we pair our spatial discretization of degree p with an explicit Runge-Kutta scheme of order $p+1$, we require

$$\Delta t \leq \frac{\min_j h_j}{\max_j \lambda_j} \cdot \frac{1}{2p+1}, \tag{3.47}$$

where h_j is the size of element j and λ_j is the fastest wave speed on element j, here $\lambda_j = |f_u(u_j)|$ [8].,

Similarly, in two-dimensions, the maximum stable time step on each element is dependent on some measure of element size and the speed and direction of flow through that element. The standard global CFL condition for two dimensions [6] when pairing a degree p spatial discretization with an order $p+1$ Runge-Kutta method is given by

$$\Delta t \leq \frac{\min_j r_j}{\max_j |\lambda_j|} \cdot \frac{1}{2p+1}, \tag{3.48}$$

where r_j is the radius of the inscribed circle in element j and λ_j is the maximum wave speed on element j.

From (3.47), and (3.48), we see that the size of the global time step is restricted by the smallest element computational mesh. When a computational mesh has elements of greatly varying sizes, this leads to inefficiencies. Large elements must be advanced in time with the same time step as the smallest element. To overcome these inefficiencies, it would be ideal that each element is advanced in time with its maximum stable time step. This is the topic of the next chapter.

Chapter 4

Local Time-Stepping

4.1 Introduction

The time integration of the discontinuous Galerkin method requires knowing the solution values of the neighboring elements at each time level to compute the numerical flux of (3.15) in one-dimension, or the surface integral component of (3.44) in two-dimensions. As an example, consider a one-dimensional uniform mesh. If we integrate in time using a three stage Runge-Kutta method the stencil shown in Figure 4.1 is needed to advance the solution on element j from t_n to t_{n+1}. Each stage is represented by an arrow. Higher order Runge-Kutta methods need a larger stencil due to the higher number of stages.

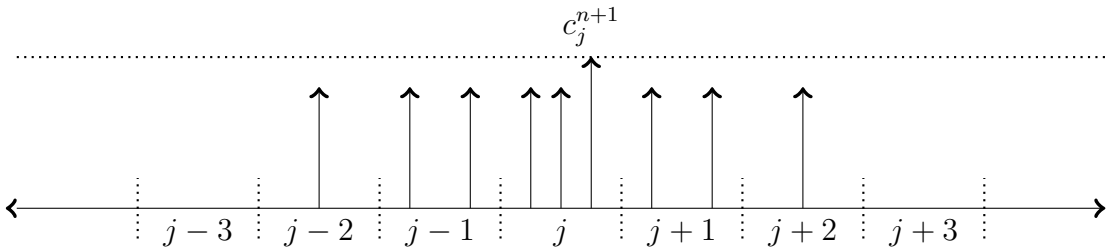


Figure 4.1: Stencil used to advance element j with a three stage Runge-Kutta method on a uniform mesh.

Consider now a nonuniform mesh where the elements are of size Δx for elements $i \leq j$ and size $\Delta x/2$ for elements $i > j$. We would like to advance the elements with their locally stable time step, say Δt for elements $i \leq j$ and $\Delta t/2$ for elements $i > j$. To advance the

solution on element j we would need the stencil given in Figure 4.2. We see that for a three stage Runge-Kutta method, we need to extend two elements into the region of small elements in the mesh. This could be problematic as we may be taking an unstable time step on these elements.

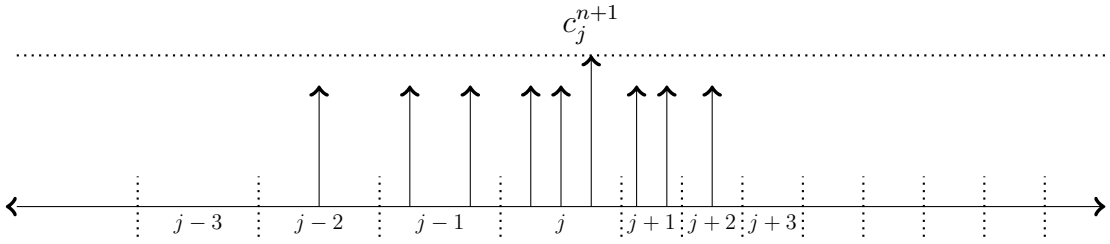


Figure 4.2: Stencil for advancing element j by the local time step size Δt using a three stage Runge-Kutta method on a nonuniform mesh.

Advancing the solution on element $j+1$ from t_n to t_{n+1} requires us to advance from t_n to $t_{n+1/2}$, then from $t_{n+1/2}$ to t_{n+1}. To compute $c_{j+1}^{n+1/2}$ only requires the information from large elements j and $j-1$ to compute the inner stages. Then, to compute the inner stages on elements j and $j-1$ to advance $c_{j+1}^{n+1/2}$ to c_{j+1}^{n+1} we need to have the solution solution values at $t_{n+1/2}$ on the large elements $j-2, j-1$ and j. This requires us to extend the stencil outward in space to element $j-4$, shown in Figure 4.3. Again, higher order Runge-Kutta methods will require a larger stencil.

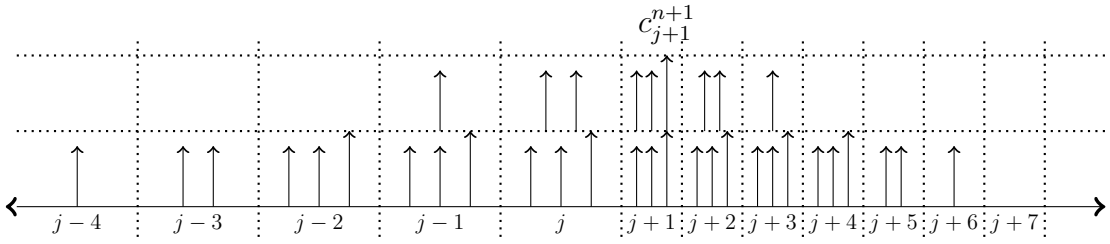


Figure 4.3: Stencil for advancing element $j+1$ by the local time step size $\Delta t/2$ to t_{n+1} using a three stage Runge-Kutta method on a nonuniform mesh.

The example presented above illustrates how extensive the stencil can be for advancing elements with their local time steps without modifying the time-stepping method. Our goal is to develop time stepping methods that do not require us to extend the stencil outward in space through the refined or coarsened regions. On elements where the neighboring element requires a different sized time step, we will not compute stages higher than 1.

Instead, we will approximate the stages using data from previous time steps in such a way that we keep the accuracy of the underlying Runge-Kutta method.

Although our goal is to develop local time-stepping methods for the numerical solution of partial differential equations, we begin this section by developing local time-stepping methods for a system of two ordinary differential equations using a Runge-Kutta 3 scheme and the classical Runge-Kutta 4 scheme. This will be an easier exposition of the methods and allow us to arrive at some theoretical results. Finally, we will present how they can be used with the discontinuous Galerkin method for solving problems on computational meshes with elements of drastically different sizes.

4.2 Local Time-Stepping for Ordinary Differential Equations

Suppose we would like to numerically solve the system of ordinary differential equations

$$x' = f(x, y) \tag{4.1a}$$
$$y' = g(x, y), \tag{4.1b}$$

where $x = x(t), y = y(t)$. Suppose x can be advanced with time step $h_{n+1} = t_{n+1} - t_n$ and y has to be advanced with time step $h_{n+1}/K, K \in \mathbb{N}$.

We follow the same approach presented in [22] to develop local time-stepping schemes for the system (4.1). First, we advance the x in time. In order to do so, we must approximate the inner stages of y. These approximations are done by using data from the current and previous time levels to create an approximation that is at least as accurate as the local error of the actual Runge-Kutta stages.

Once x has been advanced one time step, we will create an interpolating polynomial $\mathcal{X}(t)$ which approximates $x(t)$ on the interval $[t_n, t_{n+1}]$. This interpolating polynomial will be at least as accurate as the Runge-Kutta scheme being used. The interpolating polynomial $\mathcal{X}(t)$, along with its derivatives, will be used in approximating the inner stages of x to advance y through the K sub time steps needed to advance from t_n to t_{n+1}. Note that we cannot simply use the value of the interpolating polynomial at the inner stage times due to the fact that we need to commit low order errors to match the low order errors of the stages in y.

4.2.1 Runge-Kutta 3 Local Time-Stepping

The following local time-stepping method is based on the family of Runge-Kutta 3 methods given by Table 2.3 with $\alpha = 3/8$. It is shown below in Table 4.1.

0	0	0	0
2/3	2/3	0	0
2/3	0	2/3	0
	1/4	3/8	3/8

Table 4.1: Butcher Tableau for the Runge-Kutta 3 method used to develop the local time-stepping method.

Assume that at time t_n both solutions x_n and y_n are known. We also assume we have stored some data from the previous time level, namely $f(x_{n-1}, y_{n-1})$, $g(x_{n-1}, y_{n-1})$ and the size of the time step h_n.

We begin by advancing x from t_n to t_{n+1}. To do this, we must approximate the intermediate stages of y. The scheme for advancing x is given by

$$X^{(1)} = x_n + \frac{2}{3}h_{n+1}f(x_n, y_n), \tag{4.2a}$$

$$Y^{(1)} = y_n + \frac{2}{3}h_{n+1}g(x_n, y_n), \tag{4.2b}$$

$$X^{(2)} = x_n + \frac{2}{3}h_{n+1}f(X^{(1)}, Y^{(1)}), \tag{4.2c}$$

$$Y^{(2)} = y_n + \frac{2}{3}h_{n+1}g(x_n, y_n) + \frac{4}{9}h_{n+1}^2\left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n}\right), \tag{4.2d}$$

$$x_{n+1} = x_n + \frac{h_{n+1}}{4}\left(f(x_n, y_n) + \frac{3}{2}f(X^{(1)}, Y^{(1)}) + \frac{3}{2}f(X^{(2)}, Y^{(2)})\right). \tag{4.2e}$$

In (4.2) all steps are standard Runge-Kutta 3 steps applied to system (4.1) except for (4.2d). Expanding the actual second Runge-Kutta stage of y in a Taylor series about t_n gives

$$\begin{aligned} Y_{rk}^{(2)} &= y_n + \frac{2}{3}h_{n+1}f\left(X_{rk}^{(1)}, Y_{rk}^{(1)}\right) \\ &= y_n + \frac{2}{3}h_{n+1}f\left(x_n + \frac{2}{3}h_{n+1}f(x_n, y_n), y_n + \frac{2}{3}h_{n+1}g(x_n, y_n)\right) \\ &= y_n + \frac{2}{3}h_{n+1}g(x_n, y_n) + \frac{4}{9}h_{n+1}^2(g_x f + g_y g)(x_n, y_n) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{4.3}$$

Since $g(x_n, y_n)$ is known, we only need to approximate $(g_x f + g_y g)(x_n, y_n)$. We notice that the chain rule gives

$$\frac{d}{dt} g(x(t), y(t)) = (g_x f + g_y g)(x(t), y(t)), \tag{4.4}$$

and so we use a backward difference method to approximate (4.4)

$$\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} = \frac{d}{dt} g(x(t), y(t)) + \mathcal{O}(h_n). \tag{4.5}$$

Assuming that $h_n = c h_{n+1}$ for some constant c, we substitute (4.5) into (4.3) to arrive at (4.2d).

Having advanced x_n to x_{n+1}, we seek to construct a polynomial of degree 3, $\mathcal{X}(t)$ which interpolates the numerical solution x along the segment $[t_n, t_{n+1}]$. We require $\mathcal{X}(t_n) = x_n, \mathcal{X}(t_{n+1}) = x_{n+1}$ and $\mathcal{X}'(t_n) = f(x_n, y_n)$. To have a cubic interpolating polynomial, we require one more point. Choosing $\mathcal{X}'(t_n + 2h_{n+1}/3) = (f(X^{(1)}, Y^{(1)}) + f(X^{(2)}, Y^{(2)}))/2$, we get the family of interpolating polynomials

$$\begin{aligned} \mathcal{X}(t) = x_n + (t - t_n) f(x_n, y_n) + (t - t_n)^2 \left(\frac{x_{n+1} - x_n - h_{n+1} f(x_n, y_n)}{h_{n+1}^2} - h_{n+1} \beta \right) \\ + (t - t_n)^3 \beta, \quad t_n \le t \le t_{n+1}, \end{aligned} \tag{4.6a}$$

where β is a parameter. Under the assumption that $x_n = x(t_n)$, we choose β so that $|x(t) - \mathcal{X}(t)| = \mathcal{O}(h_{n+1}^4)$ for all $t_n \le t \le t_{n+1}$. Expanding the interpolating polynomial and the exact solution in a Taylor series about t_n reveals that this is satisfied when $\beta = x'''(t_n) + \mathcal{O}(h_{n+1})$. Using finite differences we have

$$\beta = \frac{1}{2h_{n+1} + 3h_n} \left(2 \frac{x_{n+1} - x_n - h_{n+1} f(x_n, y_n)}{h_{n+1}^2} - \frac{f(x_n, y_n) - f(x_{n-1}, y_{n-1})}{h_n} \right). \tag{4.6b}$$

We are now tasked with advancing y through the fractional time steps $t_{n,k} = t_n + k h_{n+1}/K, k = 0, 1, \dots K$. Denote the computed value of y at time $t_{n,k}$ by $y_{n,k}$, where $y_{n,0} = y_n$ and $y_{n,K} = y_{n+1}$. Additionally let $X_k^{(i)}, Y_k^{(i)}, i = 1, 2$, denote the inner Runge-Kutta stages at the fractional step k. Since the interpolating polynomial (4.6) is locally fourth order accurate, we also have $x'(t) = \mathcal{X}'(t) + \mathcal{O}(h_{n+1}^3)$ and $x''(t) = \mathcal{X}''(t) + \mathcal{O}(h_{n+1}^2)$ for all $t_n \le t \le t_{n+1}$. Using these approximations, we have the following scheme to advance

$y_{n,k}$ to $y_{n,k+1}$

$$x_{n,k} = \mathcal{X}(t_{n,k}), \tag{4.7a}$$

$$X_k^{(1)} = x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}), \tag{4.7b}$$

$$Y_k^{(1)} = y_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) g(x_{n,k}, y_{n,k}), \tag{4.7c}$$

$$X_k^{(2)} = x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2 \mathcal{X}''(t_{n,k}), \tag{4.7d}$$

$$Y_k^{(2)} = y_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) g(X_k^{(1)}, Y_k^{(1)}), \tag{4.7e}$$

$$y_{n,k+1} = y_{n,k} + \left(\frac{h_{n+1}}{K}\right)\left(\frac{1}{4}\right)\left(g(x_{n,k}, y_{n,k}) + \frac{3}{2}g(X_k^{(1)}, Y_k^{(1)}) + \frac{3}{2}g(X_k^{(2)}, Y_k^{(2)})\right). \tag{4.7f}$$

In (4.7) all steps are standard Runge-Kutta 3 steps applied to (4.1) except for (4.7b) and (4.7d). In (4.7b), we use the fact that $f(x(t_{n,k}), y(t_{n,k})) = x'(t_{n,k})$ and $x'(t_{n,k}) = \mathcal{X}'(t_{n,k}) + \mathcal{O}(h_{n+1}^3)$. Additionally, expanding the actual second Runge-Kutta stage of x in a Taylor series about $t_{n,k}$ gives

$$\begin{aligned} X_{rk}^{(2)} &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f\left(X_{rk}^{(1)}, Y_{rk}^{(1)}\right) \\ &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f\left(x_{n,k} + \frac{2}{3}\left(\frac{h}{K}\right) f(x_{n,k}, y_{n,k}), y_{n,k} + \frac{2}{3}\left(\frac{h}{K}\right) g(x_{n,k}, y_{n,k})\right) \\ &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f(x_{n,k}, y_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2 (f_x f + f_y g)(x_{n,k}, y_{n,k}) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{4.8}$$

To accurately approximate (4.8), we need an approximation to $f(x_{n,k}, y_{n,k})$ and $(f_x f + f_y g)(x_{n,k}, y_{n,k})$. Observing that $f(x(t_{n,k}), y(t_{n,k})) = x'(t_{n,k})$ and $(f_x f + f_y g)(x(t_{n,k}), y(t_{n,k})) = x''(t_{n,k})$, we use $x'(t_{n,k}) = \mathcal{X}'(t_{n,k}) + \mathcal{O}(h_{n+1}^3)$ and $x''(t_{n,k}) = \mathcal{X}''(t_{n,k}) + \mathcal{O}(h_{n+1}^2)$ to arrive at (4.7d) See Appendix A for a detailed derivation of (4.2)-(4.7).

Next, we will show that (4.2)-(4.7) is a third order scheme.

Lemma 3. *If f, g are C^4, then*

(*i*) *the local error of (4.2) is $\mathcal{O}(h^4)$,*

(*ii*) *the interpolating polynomial* $\mathcal{X}(t)$ *is a locally fourth order approximation of* $x(t)$ *on the interval* $[t_n, t_{n+1}]$ *provided* $x_n = x(t_n)$,

(*iii*) *the local error of (4.7) is* $\mathcal{O}(h^4)$,

(*iv*) *the global error of (4.2)-(4.7) is* $\mathcal{O}(h^3)$.

Proof. Assume $h_n = h_{n+1} = h$. First, we analyze the local error in x

$$l^x_{n+1} \equiv x(t_{n+1}) - x_{n+1}. \tag{4.9}$$

Assuming no error is committed prior to step n, i.e., $x_j = x(t_j), y_j = y(t_j), j \le n$, we expand (4.2e) and $x(t_{n+1})$ in a Taylor series about t_n, obtaining

$$\begin{aligned} x(t_{n+1}) - x_{n+1} &= x(t_{n+1}) \\ &\quad - \left(x_n + \frac{h_{n+1}}{4}\left(f(x_n, y_n) + \frac{3}{2} f(X^{(1)}, Y^{(1)}) + \frac{3}{2} f(X^{(2)}, Y^{(2)}) \right) \right) \\ &= \left(x(t_n) + hx' + \frac{h^2}{2}x'' + \frac{h^3}{6}x''' + \frac{h^4}{24}x^{(4)}(\xi_1) \right) - \Bigg(x_n + hf \\ &\quad + \frac{h^2}{2}(f_x f + f_y g) + \frac{h^3}{6}\Big(f_{xx}f^2 + 2f_{xy}fg + f_x^2 f + f_x f_y g + f_{yy}g^2 \\ &\quad + f_y g_x f + f_y g_y g \Big) + h^4 \mathcal{R}_x(\xi_2) \Bigg), \end{aligned}$$

where ξ_1 is a point in the interval $[t_n, t_{n+1}]$, ξ_2 is a point on the line connecting (x_n, y_n) to $(x_n + hf, y_n + hg)$, and $\mathcal{R}_x$ is the remainder term in the Taylor expansion of (4.2e). The terms of $x^{(4)}(t)$ and $\mathcal{R}_x(t)$ are given by (C.1) and (C.2), in Appendix C. Noting that

$$\begin{aligned} x' = f, \quad x'' = f_x f + f_y g, \\ x''' = f_{xx}f^2 + 2f_{xy}fg + f_x^2 f + f_x f_y g + f_{yy}g^2 + f_y g_x f + f_y g_y g, \end{aligned}$$

we have

$$|l^x_{n+1}| = \left| h^4 \left(\frac{1}{24} x^{(4)}(\xi_1) - \mathcal{R}_x(\xi_2) \right) \right| \le C h^4, \tag{4.10}$$

which proves (*i*).

Next expanding the exact solution $x(t)$, the interpolating polynomial $\mathcal{X}(t)$ and its coefficients in a Taylor series about t_n gives

$$x(t_n + \delta h) - \mathcal{X}(t_n + \delta h) = \frac{(\delta h)^4}{24} x^{(4)}(\xi_1) - h^4 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_1(\delta) h^4, \tag{4.11a}$$

where $\mathcal{R}_{\mathcal{X}}$ is the remainder term for the interpolating polynomial, and $0 \leq \delta \leq 1$, see (C.3) in Appendix C. This shows that the interpolating polynomial provides a locally fourth order accurate approximation to $x(t)$ on the interval $[t_n, t_{n+1}]$, proving (*ii*). Additionally, analyzing the local error of the derivatives of the interpolating polynomial $\mathcal{X}'(t)$ reveals

$$x'(t_n + \delta h) - \mathcal{X}'(t_n + \delta h) = \frac{(\delta h)^3}{6} x^{(4)}(\xi_1) - h^3 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_2(\delta) h^3, \tag{4.11b}$$

$$x''(t_n + \delta h) - \mathcal{X}''(t_n + \delta h) = \frac{(\delta h)^2}{2} x^{(4)}(\xi_1) - h^2 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_3(\delta) h^2, \tag{4.11c}$$

where $0 \leq \delta \leq 1$. This show that $\mathcal{X}'(t)$ gives a locally third order approximation to $x'(t)$ and $\mathcal{X}''(t)$ gives a locally second order approximation to $x''(t)$on the interval $[t_n, t_{n+1}]$.

Next, we analyze the local error in (4.7f)

$$l^y_{n,k+1} \equiv y(t_{n,k+1}) - y_{n,k+1}. \tag{4.12}$$

We assume there was no error committed prior to $t_{n.k}$, i.e., $x_{n,j} = x(t_{n,j}), y_{n,j} = y(t_{n,j}), j = 0, 1, \ldots, k$. First, substituting (4.11) into (4.7) with $\delta = k/K$ gives

$$X^{(1)}_k = x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^4 + \frac{2}{3}\left(\frac{h}{K}\right)\left(f(x(t_{n,k}), y(t_{n,k})) + C_2 \left(\frac{k}{K}\right) h^3\right), \tag{4.13a}$$

$$Y^{(1)}_k = y(t_{n,k}) + \frac{2}{3}\left(\frac{h}{K}\right) g\left(x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^4, y(t_{n,k})\right), \tag{4.13b}$$

$$\begin{aligned} X^{(2)}_k = {} & x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^4 + \frac{2}{3}\left(\frac{h}{K}\right)\left(f(x(t_{n,k}), y(t_{n,k})) + C_2 \left(\frac{k}{K}\right) h^3\right) \\ & + \frac{4}{9}\left(\frac{h}{K}\right)^2 \left((f_x f + f_y g)(x(t_{n,k}), y(t_{n,k})) + C_3 \left(\frac{k}{K}\right) h^2\right), \end{aligned} \tag{4.13c}$$

$$Y^{(2)}_k = y(t_{n,k}) + \frac{2}{3}\left(\frac{h}{K}\right) g\left(X^{(1)}_k, Y^{(1)}_k\right). \tag{4.13d}$$

Then (4.7f) becomes

$$\begin{aligned} y_{n,k+1} = y(t_{n,k}) + {} & \left(\frac{h_{n+1}}{K}\right)\left(\frac{1}{4}\right)\left(g\left(x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^4, y(t_{n,k})\right)\right. \\ & \left. + \frac{3}{2} g(X^{(1)}_k, Y^{(1)}_k) + \frac{3}{2} g(X^{(2)}_k, Y^{(2)}_k)\right). \end{aligned} \tag{4.13e}$$

Substituting (4.13) into (4.12) and expanding each term in a Taylor series about $t_{n,k}$ gives

$$|l^y_{n,k+1}| = \left| \left(\frac{h}{K}\right)^4 \left(\frac{y^{(4)}(\xi_1)}{24} - \mathcal{R}_{y,k+1}(\xi_2)\right) + \mathcal{O}(h^5) \right| \leq Ch^4, \tag{4.14}$$

where $\mathcal{R}_{y,k+1}$ is the remainder of the Taylor series of (4.13e) about $t_{n,k}$. The terms of $y^{(4)}(t)$ and $\mathcal{R}_{y,k+1}(t)$ are given by (C.4) and (C.5), respectively, in Appendix C. This proves (*iii*).

The proof of (*iv*) follows from the proof of Lemma 1(*iv*) in [22], using the error estimates (4.10), (4.14). This completes the proof.

□

4.2.2 Runge-Kutta 4 Local Time-Stepping

The following local-time stepping method is based on the classical Runge-Kutta 4 method shown below in Table 4.2.

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

Table 4.2: Butcher tableau for the classical Runge-Kutta 4 method to be used for local time-stepping.

We follow the same procedure as Section 4.2.1 where we approximate the inner Runge-Kutta stages. We state the inner stage approximations without justification as they are derived by approximating the Taylor expansion of the stages. We refer the reader to Appendix B for a detailed derivation of the following method.

Assume that at time t_n both solutions x_n and y_n are known. We also assume we have stored some previous data, namely $f(x_{n-1}, y_{n-1})$, $g(x_{n-1}, y_{n-1})$, $f(x_{n-2}, y_{n-2})$, $g(x_{n-2}, y_{n-2})$, and time step sizes h_n and h_{n-1}.

We begin by advancing x from t_n to t_{n+1}. To do this, we must approximate the intermediate stages of y. First, for y, we define a "correction term". This term is nonzero when $h_{n+1} \neq h_n$, and is used to replace high-order derivatives in the Taylor expansion

multiplied by time step size h_n with high-order derivatives multiplied by the current time step size h_{n+1}. It is given by

$$Corr_y = \left(2\frac{h_{n+1}-h_n}{h_n+h_{n-1}}\right)\left(\frac{g(x_n,y_n)-g(x_{n-1},y_{n-1})}{h_n} - \frac{g(x_{n-1},y_{n-1})-g(x_{n-2},y_{n-2})}{h_{n-1}}\right). \tag{4.15}$$

To advance x_n to x_{n+1} we have the following scheme

$$X^{(1)} = x_n + \frac{1}{2}h_{n+1}f(x_n,y_n), \tag{4.16a}$$

$$Y^{(1)} = y_n + \frac{1}{2}h_{n+1}g(x_n,y_n), \tag{4.16b}$$

$$X^{(2)} = x_n + \frac{1}{2}h_{n+1}f(X^{(1)},Y^{(1)}), \tag{4.16c}$$

$$Y^{(2)} = y_n + \frac{1}{2}h_{n+1}g(x_n,y_n) + \frac{1}{4}h_{n+1}^2\left(\frac{g(x_n,y_n)-g(x_{n-1},y_{n-1})}{h_n} - \frac{1}{2}Corr_y\right), \tag{4.16d}$$

$$X^{(3)} = x_n + h_{n+1}f(X^{(2)},Y^{(2)}), \tag{4.16e}$$

$$\begin{aligned} Y^{(3)} = {} & y_n + h_{n+1}g(x_n,y_n) + \frac{1}{2}h_{n+1}^2\left(\frac{g(x_n,y_n)-g(x_{n-1},y_{n-1})}{h_n} - \frac{1}{2}Corr_y\right) \\ & + \frac{3}{4}h_{n+1}^3\left(\frac{2}{h_n+h_{n-1}}\right)\left(\frac{g(x_n,y_n)-g(x_{n-1},y_{n-1})}{h_n} - \frac{g(x_{n-1},y_{n-1})-g(x_{n-2},y_{n-2})}{h_{n-1}}\right), \end{aligned} \tag{4.16f}$$

$$x_{n+1} = x_n + \frac{h_{n+1}}{6}\left(f(x_n,y_n) + 2f(X^{(1)},Y^{(1)}) + 2f(X^{(2)},Y^{(2)} + f(X^{(3)},Y^{(3)})\right). \tag{4.16g}$$

We see in (4.16) that all steps are standard Runge-Kutta 4 steps applied the system (4.1) except for (4.16d) and (4.16f).

Having advanced x_n to x_{n+1}, we seek to construct a polynomial of degree 4, $\mathcal{X}(t)$ which interpolates the numerical solution x along the segment $[t_n, t_{n+1}]$. We construct this polynomial in such a way that under the assumption that $x_n = x(t_n)$, it satisfies $|x(t) - \mathcal{X}(t)| = \mathcal{O}(h_{n+1}^5)$ for all $t_n \le t \le t_{n+1}$.

Similarly to the Runge-Kutta 3 local time-stepping polynomial, we choose $\mathcal{X}(t_n) = x_n, \mathcal{X}(t_{n+1}) = x_{n+1}$ and $\mathcal{X}'(t_n) = f(x_n,y_n)$. To have a quartic interpolating polynomial, we require two more points. Using the solution values x_{n+1} and x_n, along with the current and previous derivative values $f(x_n,y_n), f(x_{n-1},y_{n-1})$ and $f(x_{n-2},y_{n-2})$, we use finite difference

methods to approximate derivatives of $x(t)$

$$\beta = \frac{6(2h_{n+1}+3h_n)}{6h_n^2+8h_{n+1}h_n+6h_nh_{n-1}+3h_{n+1}^2+4h_{n+1}h_{n-1}}\Bigg[\left(\frac{6}{2h_{n+1}+3h_n}\right)\left(2\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-\frac{f(x_n,y_n)-f(x_{n-1},y_{n-1})}{h_n}\right) - \left(\frac{2}{h_n+h_{n-1}}\right)\left(\frac{f(x_n,y_n)-f(x_{n-1},y_{n-1})}{h_n}-\frac{f(x_{n-1},y_{n-1})-f(x_{n-2},y_{n-2})}{h_{n-1}}\right)\Bigg], \tag{4.17a}$$

$$\alpha = \left(\frac{6}{2h_{n+1}+3h_n}\right)\left(2\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-\frac{f(x_n,y_n)-f(x_{n-1},y_{n-1})}{h_n}\right) + \left(\frac{3h_{n+1}^2+6h_{n+1}h_n+2h_n^2}{2(2h_{n+1}+3h_n)}\right)\beta, \tag{4.17b}$$

where $\alpha = x^{(3)}(t_{n+1}) + \mathcal{O}(h_{n+1}^2)$ and $\beta = x^{(4)}(t_n) + \mathcal{O}(h_{n+1})$. Choosing $\mathcal{X}^{(3)}(t_{n+1}) = \alpha$ and $\mathcal{X}^{(4)}(t_n) = \beta$, we get the unique interpolating polynomial

$$\begin{aligned}\mathcal{X}(t) &= x_n + (t-t_n)f(x_n,y_n) \\ &\quad + (t-t_n)^2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-\frac{h_{n+1}}{6}\alpha+\frac{h_{n+1}^2}{8}\beta\right) \\ &\quad + \frac{(t-t_n)^3}{6}(\alpha-h_{n+1}\beta)+\frac{(t-t_n)^4}{24}\beta, \quad t_n \le t \le t_{n+1}.\end{aligned} \tag{4.17c}$$

We are now tasked with advancing y through the intermediate time steps $t_{n,k} = t_n + kh_{n+1}/K, k = 0, 1, \dots K$. Denote the computed value of y at time $t_{n,k}$ by $y_{n,k}$, where $y_{n,0} = y_n$ and $y_{n,K} = y_{n+1}$. Additionally, let $X_k^{(i)}, Y_k^{(i)}, i = 1, 2, 3$, denote the inner Runge-Kutta stages at the fractional step k. We advance $y_{n,k}$ to $y_{n,k+1}$ with the following scheme

$$x_{n,k} = \mathcal{X}(t_{n,k}), \tag{4.18a}$$

$$X_k^{(1)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}), \tag{4.18b}$$

$$Y_k^{(1)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(x_{n,k}, y_{n,k}), \tag{4.18c}$$

$$X_k^{(2)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^2\mathcal{X}''(t_{n,k}), \tag{4.18d}$$

$$Y_k^{(2)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(X_k^{(1)}, Y_k^{(1)}), \tag{4.18e}$$

$$X_k^{(3)} = x_{n,k} + \left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2\mathcal{X}''(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^3\mathcal{X}'''(t_{n,k}), \tag{4.18f}$$

$$Y_k^{(3)} = y_{n,k} + \left(\frac{h_{n+1}}{K}\right) g(X_k^{(2)}, Y_k^{(2)}), \tag{4.18g}$$

$$\begin{aligned} y_{n,k+1} = y_{n,k} &+ \left(\frac{h_{n+1}}{K}\right)\left(\frac{1}{6}\right)\Big(g(x_{n,k}, y_{n,k}) + 2g(X_k^{(1)}, Y_k^{(1)}) \\ &+ 2g(X_k^{(2)}, Y_k^{(2)}) + g(X_k^{(3)}, Y_k^{(3)})\Big). \end{aligned} \tag{4.18h}$$

In (4.18) all steps are standard Runge-Kutta 4 steps applied to (4.1) except for (4.18b), (4.18d) and (4.18f).

Next, we will show that (4.16)-(4.18) is a fourth order scheme.

Lemma 4. *If f, g are C^5, then*

(*i*) *the local error of (4.16) is $\mathcal{O}(h^5)$,*

(*ii*) *the interpolating polynomial $\mathcal{X}(t)$ is a localy fifth order approximation of $x(t)$ on the interval $[t_n, t_{n+1}]$ provided $x_n = x(t_n)$,*

(*iii*) *the local error of (4.18) is $\mathcal{O}(h^5)$,*

(*iv*) *the global error of (4.16)-(4.18) is $\mathcal{O}(h^4)$.*

Proof. Assume a uniform time step, i.e., $h_{n-1} = h_n = h_{n+1} = h$. The nonuniform case is similar.

First, we analyze the local error in x

$$l_{n+1}^x \equiv x(t_{n+1}) - x_{n+1}. \tag{4.19}$$

Assuming no error is committed prior to step n, i.e., $x_j = x(t_j), y_j = y(t_j)$, $j \le n$, we expand (4.16g) and $x(t_{n+1})$ in a Taylor series about t_n, obtaining

$$\begin{aligned}
x(t_{n+1}) - x_{n+1} &= x(t_{n+1}) \\
&\quad - \Bigg(x_n + \frac{h_{n+1}}{6}\Big(f(x_n, y_n) + 2f(X^{(1)}, Y^{(1)}) + 2f(X^{(2)}, Y^{(2)}) \\
&\quad + f(X^{(3)}, Y^{(3)})\Big)\Bigg) \\
&= \left(x(t_n) + hx' + \frac{h^2}{2}x'' + \frac{h^3}{6}x''' + \frac{h^4}{24}x^{(4)} + \frac{h^5}{120}x^{(5)}(\xi_1)\right) - \Bigg(x_n + hf \\
&\quad + \frac{h^2}{2}(f_x f + f_y g) + \frac{h^3}{6}\Big(f_{xx}f^2 + 2f_{xy}fg + f_x^2 f + f_x f_y g + f_{yy}g^2 \\
&\quad + f_y g_x f + f_y g_y g\Big) \\
&\quad + \frac{h^4}{24}\Big(3f_{xx}f f_y g + 5f_{xy}g f_x f + 3f_{xy}f g_y g + 2f_x f_y g_x f + f_x f_y g_y g + 3f_{yy}g g_x f \\
&\quad + 2f_y g_{xy} f g + f_y g_y g_x f + 3f_{xxy}f^2 g + 3f_{xyy}f g^2 + f_x^2 f_y g + f_y g_{xx} f^2 \\
&\quad + f_y g_y^2 g + g_x f_y^2 g + 4f_{xx}f^2 f_x + f_{xxx}f^3 + f_{yyy}g^3 + f_x^3 f + f_y g_{yy} g^2 \\
&\quad + f_x f_{yy} g^2 + 3f_{xy}g^2 f_y + 3f_{xy}f^2 g_x + 3f_{yy}g^2 g_y\Big) + h^5 \mathcal{R}_x(\xi_2)\Bigg),
\end{aligned}$$

where ξ_1 is a point in the interval $[t_n, t_{n+1}]$, ξ_2 is a point on the line connecting (x_n, y_n) to $(x_n + hf, y_n + hg)$, and $\mathcal{R}_x$ is the remainder term in the Taylor expansion of (4.16g). The terms of $x^{(5)}(t)$ and $\mathcal{R}_x(t)$ are given by (C.6) and (C.7), respectively, in Appendix C. Noting that

$$\begin{aligned}
x' &= f, \quad x'' = f_x f + f_y g, \\
x''' &= f_{xx}f^2 + 2f_{xy}fg + f_x^2 f + f_x f_y g + f_{yy}g^2 + f_y g_x f + f_y g_y g, \\
x^{(4)} &= 3f_{xx}f f_y g + 5f_{xy}g f_x f + 3f_{xy}f g_y g + 2f_x f_y g_x f + f_x f_y g_y g + 3f_{yy}g g_x f \\
&\quad + 2f_y g_{xy} f g + f_y g_y g_x f + 3f_{xxy}f^2 g + 3f_{xyy}f g^2 + f_x^2 f_y g + f_y g_{xx} f^2 \\
&\quad + f_y g_y^2 g + g_x f_y^2 g + 4f_{xx}f^2 f_x + f_{xxx}f^3 + f_{yyy}g^3 + f_x^3 f + f_y g_{yy} g^2 \\
&\quad + f_x f_{yy} g^2 + 3f_{xy}g^2 f_y + 3f_{xy}f^2 g_x + 3f_{yy}g^2 g_y,
\end{aligned}$$

we have

$$|l_{n+1}^x| = \left| h^5 \left(\frac{1}{120} x^{(5)}(\xi_1) - \mathcal{R}_x(\xi_2) \right) \right| \leq Ch^5, \tag{4.20}$$

which proves (*i*).

Next expanding the exact solution $x(t)$, the interpolating polynomial $\mathcal{X}(t)$ and its coefficients in a Taylor series about t_n gives

$$x(t_n + \delta h) - \mathcal{X}(t_n + \delta h) = \frac{(\delta h)^5}{120} x^{(5)}(\xi_1) - h^5 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_1(\delta) h^5, \tag{4.21a}$$

where $\mathcal{R}_{\mathcal{X}}$ is the remainder term for the interpolating polynomial, and $0 \leq \delta \leq 1$, see (C.8) in Appendix C. This proves (*ii*). The interpolating polynomial provides a locally fifth order accurate approximation to $x(t)$ on the interval $[t_n, t_{n+1}]$. Additionally, analyzing the local error of the derivatives of the interpolating polynomial reveals

$$x'(t_n + \delta h) - \mathcal{X}'(t_n + \delta h) = \frac{(\delta h)^4}{24} x^{(5)}(\xi_1) - h^4 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_2(\delta) h^4, \tag{4.21b}$$

$$x''(t_n + \delta h) - \mathcal{X}''(t_n + \delta h) = \frac{(\delta h)^3}{6} x^{(5)}(\xi_1) - h^3 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_3(\delta) h^3, \tag{4.21c}$$

$$x'''(t_n + \delta h) - \mathcal{X}'''(t_n + \delta h) = \frac{(\delta h)^2}{2} x^{(5)}(\xi_1) - h^2 \mathcal{R}_{\mathcal{X}}(\xi_2) = C_4(\delta) h^2, \tag{4.21d}$$

where $0 \leq \delta \leq 1$. This show that $\mathcal{X}'(t)$ gives a locally fourth order approximation to $x'(t)$, $\mathcal{X}''(t)$ gives a locally third order approximation to $x''(t)$ and $\mathcal{X}'''(t)$ gives a locally second order approximation to $x'''(t)$on the interval $[t_n, t_{n+1}]$.

Next, we analyze the local error in (4.18h)

$$l_{n,k+1}^y \equiv y(t_{n,k+1}) - y_{n,k+1}. \tag{4.22}$$

We assume there was no error committed prior to $t_{n.k}$, i.e., $x_{n,j} = x(t_{n,j}), y_{n,j} =$

$y(t_{n,j}), j = 0, 1, \ldots, k$. First, substituting (4.21) into (4.18) with $\delta = k/K$ gives

$$X_k^{(1)} = x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^5 + \frac{1}{2}\left(\frac{h}{K}\right)\left(f(x(t_{n,k}), y(t_{n,k})) + C_2\left(\frac{k}{K}\right) h^4\right), \tag{4.23a}$$

$$Y_k^{(1)} = y(t_{n,k}) + \frac{1}{2}\left(\frac{h}{K}\right) g\left(x(t_{n,k}) + C_1\left(\frac{k}{K}\right) h^5, y(t_{n,k})\right), \tag{4.23b}$$

$$\begin{aligned} X_k^{(2)} = {} & x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^5 + \frac{1}{2}\left(\frac{h}{K}\right)\left(f(x(t_{n,k}), y(t_{n,k})) + C_2\left(\frac{k}{K}\right) h^4\right) \\ & + \frac{1}{4}\left(\frac{h}{K}\right)^2 \left(\frac{d}{dt} f(x(t_{n,k}), y(t_{n,k})) + C_3\left(\frac{k}{K}\right) h^3\right), \end{aligned} \tag{4.23c}$$

$$Y_k^{(2)} = y(t_{n,k}) + \frac{1}{2}\left(\frac{h}{K}\right) g\left(X_k^{(1)}, Y_k^{(1)}\right), \tag{4.23d}$$

$$\begin{aligned} X_k^{(3)} = {} & x(t_{n,k}) + C_1 \left(\frac{k}{K}\right) h^5 + \left(\frac{h}{K}\right)\left(f(x(t_{n,k}), y(t_{n,k})) + C_2\left(\frac{k}{K}\right) h^4\right) \\ & + \frac{1}{2}\left(\frac{h}{K}\right)^2 \left(\frac{d}{dt} f(x(t_{n,k}), y(t_{n,k})) + C_3\left(\frac{k}{K}\right) h^3\right) \\ & + \frac{1}{4}\left(\frac{h}{K}\right)^3 \left(\frac{d^2}{dt^2} f(x(t_{n,k}), y(t_{n,k})) + C_4\left(\frac{k}{K}\right) h^2\right), \end{aligned} \tag{4.23e}$$

$$Y_k^{(3)} = y(t_{n,k}) + \left(\frac{h}{K}\right) g\left(X_k^{(2)}, Y_k^{(2)}\right). \tag{4.23f}$$

Then (4.18h) becomes

$$\begin{aligned} y_{n,k+1} = y(t_{n,k}) + \left(\frac{h_{n+1}}{K}\right)\left(\frac{1}{6}\right) \Bigg(& g\left(x(t_{n,k}) + C_1\left(\frac{k}{K}\right) h^5, y(t_{n,k})\right) \\ & + 2g(X_k^{(1)}, Y_k^{(1)}) + 2g(X_k^{(2)}, Y_k^{(2)}) + g(X_k^{(3)}, Y_k^{(3)}) \Bigg). \end{aligned} \tag{4.23g}$$

Substituting (4.23) into (4.22) and expanding each term in a Taylor series about $t_{n,k}$ gives

$$|l_{n,k+1}^y| = \left| \left(\frac{h}{K}\right)^5 \left(\frac{y^{(5)}(\xi_1)}{120} - \mathcal{R}_{y,k+1}(\xi_2)\right) \right| \leq Ch^5, \tag{4.24}$$

where $\mathcal{R}_{y,k+1}$ is the remainder of the Taylor series of (4.23g) about $t_{n,k}$. The terms of $y^{(5)}(t)$ and $\mathcal{R}_{y,k+1}(t)$ are given by (C.9) and (C.10), respectively, in Appendix C. This proves (*iii*).

The proof of (*iv*) follows from the proof of Lemma 1(*iv*) in [22], using the error estimates (4.20), (4.24). This completes the proof. □

4.3 Local Time-Stepping with the Discontinuous Galerkin Method

The global CFL condition for the discontinuous Galerkin method when paired with Runge-Kutta time-stepping was discussed in Section 3.5. There we saw that when using a globally stable time step, we were constrained by using the stability condition of the smallest element in the mesh. To overcome this, we will adapt the local time-stepping algorithms developed in Section 4.2 to be used for the time integration of the discontinuous Galerkin spacial discretization.

To apply these algorithms, we must first sort the mesh elements. Let r be the size of the largest element in the mesh with Δt being the corresponding stable time step on this element. We then group elements into a "bin" based on their size relative to the maximum element size. Element k with element size r_k, satisfying $r/2^{i+1} \leq r_k < r/2^i$ will be placed into bin(i). Bin(i) will be advanced with the time step $\Delta t/2^{i+1}$.

Next, we must categorize elements based on which bin they are in and which bins their neighboring elements are in. Neighboring elements are elements which share a vertex or an edge in one- or two- dimensions, respectively. Within each bin, elements are categorized as follows: interior elements are elements in which all neighboring elements are in the same bin; large interface elements are elements which have at least one neighbor belonging to a bin of smaller elements; small interface elements are elements which have at least one neighbor belonging to a bin of larger elements. Elements can be categorized as both a large interface element and a small interface element.

When explicit time-stepping is used, the discontinuous Galerkin method is local which means that we can solve our system element-wise in any order. The only communication between elements is through the surface integral

$$\int_{\partial\Omega_j} \psi \mathbf{F}_n(\mathbf{U}_j, \mathbf{U}_{j+}) \cdot \mathbf{n} d\xi, \tag{4.25}$$

where $\mathbf{U}_{j+}$ is a neighboring element of $\mathbf{U}_j$. With this method, boundary conditions are imposed through the evaluation of the surface integral (4.25). Since no neighboring element $\mathbf{U}_{j+}$ is present on the boundary of the domain we assign a ghost value which will be used in place of the neighbor.

To deal with the computation of the surface integral (4.25) on the interface between large and small elements, we will adopt the approach used to deal with boundary elements. Due to the local property of the discontinuous Galerkin method, we can think of advancing

each bin as its own smaller problem with the boundaries being either physical boundaries or neighboring elements from different bins. On the interfaces between large and small elements we will compute ghost approximations to solution coefficient which will be denoted by the superscript G. Then, the solution value at the element boundary can be computed as

$$U(\xi_b, t) = \sum_{i=0}^{N_p} c_i^G \psi_i(\xi_b), \tag{4.26}$$

where ξ is the local variable on the canonical element Ω_0, and ξ_b corresponds to the values of ξ which lie along the interface. The functions ψ_i correspond to the ith basis function on Ω_0. The solution values using the ghost approximations at each of the Runge-Kutta stages can be computed the same way.

The proposed algorithms only require us to change the time-stepping procedure on interface elements. We are able to use standard Runge-Kutta time stepping on all interior element using their locally stable time step. The advantage of this is that these local time-stepping algorithms can be implemented into preexisting code without having to rewrite the entire time-stepping functionality.

4.3.1 Runge-Kutta 3 Local Time-Stepping with the Discontinuous Galerkin Method

In this section, we adapt the algorithm developed in Section 4.2.1 to be used with the discontinuous Galerkin method. Assume that at time t_n, all elements are at the same time level. Further, assume we have stored $\mathbf{f}(\mathbf{c})$ at time t_{n-1} on the large and small interface elements and we have stored the time step size $h_n = t_n - t_{n-1}$.

We begin by advancing the large elements with time step $h_{n+1} = t_{n+1} - t_n$, the largest stable time step on these elements. The solution coefficients on each of the small interface elements for the first Runge-Kutta stage are given by

$$C_{si}^{(1),G} = c_{si}^n + \frac{2}{3} h_{n+1} f_{si}(\mathbf{c}^n), \quad i = 0, 1, \ldots, N_p. \tag{4.27a}$$

Using (4.2d), the coefficients for the second Runge-Kutta stage on the small interface elements are given by

$$C_{si}^{(2),G} = c_{si}^n + \frac{2}{3} h_{n+1} f_{si}(\mathbf{c}^n) + \frac{4}{9} h_{n+1}^2 \left(\frac{f_{si}(\mathbf{c}^n) - f_{si}(\mathbf{c}^{n-1})}{h_n} \right), \quad i = 0, 1, \ldots, N_p. \tag{4.27b}$$

The large elements can now be advanced from t_n to t_{n+1}.

Now, using (4.6), we create a cubic interpolating polynomial along each large element interface to use as a boundary condition for updating the solution coefficients on the small interface elements. This polynomial is given by

$$b_{li}(t) = c_{li}^n + (t-t_n)f_{li}(\mathbf{c}^n) + (t-t_n)^2\left(\frac{c_{li}^{n+1} - c_{li}^n - h_{n+1}f_{li}(\mathbf{c}^n)}{h_{n+1}^2} - h_{n+1}\beta_{li}\right) + (t-t_n)^3\beta_{li}, \tag{4.28a}$$

$$\beta_{li} = \frac{1}{2h_{n+1}+3h_n}\left(2\frac{c_{li}^{n+1} - c_{li}^n - h_{n+1}f_{li}(\mathbf{c}^n)}{h_{n+1}^2} - \frac{f_{li}(\mathbf{c}^n) - f_{li}(\mathbf{c}^{n-1})}{h_n}\right), \tag{4.28b}$$

for $i = 0, 1, \ldots, N_p$ and $t_n \le t \le t_{n+1}$.

The time step on the small elements is divided into K levels, depending on the size relative to the large elements. The local sub-step time levels are $t_{n,k} = t_n + (k/K)h_{n+1}, k = 0, 1, \ldots, K$. To advance the small elements from $t_{n,k}$ to $t_{n.k+1}, k = 0, 1, 2, \ldots, K-1$, ghost coefficients on the large interface elements are given by

$$c_{li,k}^G = b_{li}(t_{n,k}), \quad i = 0, 1, \ldots, N_p. \tag{4.29a}$$

Then, (4.7b) gives the ghost coefficients for the first Runge-Kutta stage on each of the large interface elements. They are written as

$$C_{li,k}^{(1),G} = b_{li}(t_{n,k}) + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)b'_{li}(t_{n,k}), \tag{4.29b}$$

for $i = 0, 1, \ldots, N_p$. Finally, using (4.7d), the ghost coefficients for the second Runge-Kutta stage on the large interface elements are given by

$$C_{li,k}^{(2),G} = b_{li}(t_{n,k}) + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)b'_{li}(t_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2 b''_{li}(t_{n,k}), \tag{4.29c}$$

for $i = 0, 1, \ldots, N_p$. The small elements can now be advanced to the next time level until reaching $t_{n,K} = t_{n+1}$.

4.3.2 Runge-Kutta 4 Local Time-Stepping with the Discontinuous Galerkin Method

In this section, we adapt the algorithm developed in Section 4.2.2 to be used with the discontinuous Galerkin method. Assume that at time t_n, all elements are at the same time

level. Further, assume we have stored $\mathbf{f}(\mathbf{c})$ at time t_{n-1} and time t_{n-1} on the large and small interface elements and we have stored the time step sizes h_n and h_{n-1}.

On each small interface element, we use (4.15) to define the "correction term". This term is used to replace the high-order derivative terms which are multiplied by h_n by high-order derivatives which are multiplied by the current time step h_{n+1}. Using (4.15), the correction term is given by

$$Corr_{s,i} = \left(2\frac{h_{n+1}-h_n}{h_n+h_{n-1}}\right)\left(\frac{f_{si}(\mathbf{c}^n)-f_{si}(\mathbf{c}^{n-1})}{h_n} - \frac{f_{si}(\mathbf{c}^{n-1})-f_{si}(\mathbf{c}^{n-2})}{h_{n-1}}\right), \tag{4.30}$$

for $i = 0, 1, \ldots, N_p$. Note that if $h_n = h_{n+1}$ then $Corr_{s,i} = 0$.

We begin by advancing the large elements with time step $h_{n+1} = t_{n+1} - t_n$, the largest stable time step on these elements. The solution coefficients on each of the small interface elements for the first Runge-Kutta stage are given by

$$C_{si}^{(1),G} = c_{si}^n + \frac{h_{n+1}}{2} f_{si}(\mathbf{c}^n), \quad i = 0, 1, \ldots, N_p. \tag{4.31a}$$

Using (4.16d), the coefficients for the second Runge-Kutta stage on the small interface elements are given by

$$C_{si}^{(2),G} = c_{si}^n + \frac{h_{n+1}}{2} f_{si}(\mathbf{c}^n) + \frac{h_{n+1}^2}{4}\left(\frac{f_{si}(\mathbf{c}^n)-f_{si}(\mathbf{c}^{n-1})}{h_n} - \frac{1}{2}Corr_{s,i}\right), \tag{4.31b}$$

for $i = 0, 1, \ldots, N_p$. The coefficients for the third Runge-Kutta stage on the small interface elements, given by 4.16f, are

$$\begin{aligned} C_{si}^{(3),G} = c_{si}^n + h_{n+1} f_{si}(\mathbf{c}^n) + \frac{h_{n+1}^2}{2}\left(\frac{f_{si}(\mathbf{c}^n)-f_{si}(\mathbf{c}^{n-1})}{h_n} - \frac{1}{2}Corr_{s,i}\right) \\ + \frac{3h_{n+1}^3}{4}\left(\frac{2}{h_n+h_{n-1}}\right)\left(\frac{f_{si}(\mathbf{c}^n)-f_{si}(\mathbf{c}^{n-1})}{h_n} - \frac{f_{si}(\mathbf{c}^{n-1})-f_{si}(\mathbf{c}^{n-2})}{h_{n-1}}\right), \end{aligned} \tag{4.31c}$$

for $i = 0, 1, \ldots, N_p$. The large elements can now be advanced from t_n to t_{n+1}.

We now use (4.17) to create a quartic interpolating polynomial along each large element interface to use as a boundary condition for updating the solution coefficients on the small

elements. This polynomial is given by

$$
\begin{aligned}
b_{li}(t) &= c_{li}^n + (t-t_n) f_{li}(\mathbf{c}^n) \\
&\quad + (t-t_n)^2 \left(\frac{c_{li}^{n+1} - c_{li}^n - h_{n+1} f_{li}(c^n)}{h_{n+1}^2} - \frac{h_{n+1}}{6}\alpha_{li} + \frac{h_{n+1}^2}{8}\beta_{li} \right) \\
&\quad + \frac{(t-t_n)^3}{6}\left(\alpha_{li} - h_{n+1}\beta_{li}\right) + \frac{(t-t_n)^4}{24}\beta_{li}, \qquad (4.32a) \\
\beta_{li} &= \frac{6(2h_{n+1}+3h_n)}{6h_n^2 + 8h_{n+1}h_n + 6h_n h_{n-1} + 3h_{n+1}^2 + 4h_{n+1}h_{n-1}} \Bigg[\\
&\quad \left(\frac{6}{2h_{n+1}+3h_n}\right)\left(2\frac{c_{li}^{n+1} - c_{li}^n - h_{n+1} f_{li}(\mathbf{c}^n)}{h_{n+1}^2} - \frac{f_{li}(\mathbf{c}^n) - f_{li}(\mathbf{c}^{n-1})}{h_n}\right) \\
&\quad - \left(\frac{2}{h_n + h_{n-1}}\right)\left(\frac{f_{li}(\mathbf{c}^n) - f_{li}(\mathbf{c}^{n-1})}{h_n} - \frac{f_{li}(\mathbf{c}^{n-1}) - f_{li}(\mathbf{c}^{n-2})}{h_{n-1}}\right)\Bigg], \qquad (4.32b) \\
\alpha_{li} &= \frac{20}{9}\left(\frac{6}{2h_{n+1}+3h_n}\right)\left(2\frac{c_{li}^{n+1} - c_{li}^n - h_{n+1} f_{li}(\mathbf{c}^n)}{h_{n+1}^2} - \frac{f_{li}(\mathbf{c}^n) - f_{li}(\mathbf{c}^{n-1})}{h_n}\right) \\
&\quad + \left(\frac{3h_{n+1}^2 + 6h_{n+1}h_n + 2h_n^2}{2(2h_{n+1}+3h_n)}\right)\beta_{li}, \qquad (4.32c)
\end{aligned}
$$

for $i = 0, 1, \ldots, N_p$ and $t_n \le t \le t_{n+1}$.

The time step on the small elements is divided into K levels, depending on the size relative to the large element. The local sub-step time levels are $t_{n,k} = t_n + (k/K)h_{n+1}, k = 0, 1, \ldots, K$. To advance the small elements from $t_{n,k}$ to $t_{n.k+1}, k = 0, 1, 2, \ldots, K-1$, ghost coefficients on the large interface elements at $t_{n,k}$ are given by

$$
c_{li,k}^G = b_{li}(t_{n,k}), \quad i = 0, 1, \ldots, N_p. \qquad (4.33a)
$$

Next, using (4.18b), the ghost coefficients for the first Runge-Kutta stage on each of the large interface elements are given by

$$
C_{li,k}^{(1),G} = b_{li}(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) b_{li}'(t_{n,k}), \quad i = 0, 1, \ldots, N_p. \qquad (4.33b)
$$

The ghost coefficients for the second Runge-Kutta stage on the large interface elements, given by (4.18d) are

$$
C_{li,k}^{(2),G} = b_{li}(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) b_{li}'(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^2 b_{li}''(t_{n,k}), \qquad (4.33c)
$$

for $i = 0, 1, \ldots, N_p$. Finally, using (4.16f), the ghost coefficients for the third Runge-Kutta stage on the large interface elements are given by

$$C_{li,k}^{(3),G} = b_{li}(t_{n,k}) + \left(\frac{h_{n+1}}{K}\right) b'_{li}(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2 b''_{li}(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^3 b'''_{li}(t_{n,k}), \quad (4.33d)$$

for $i = 0, 1, \ldots, N_p$. The small elements can now be advanced to the next time level until reaching $t_{n,K} = t_{n+1}$.

4.4 Discussion

The methods described in Sections 4.3.1, 4.3.2 do not require additional evaluations of the function $\mathbf{f}(\mathbf{c})$, but require extra storage on the interface elements. The Runge-Kutta 3 local time-stepping algorithm requires the storage of $\mathbf{c}^n, \mathbf{f}(\mathbf{c}^n), \mathbf{f}(\mathbf{c}^{n-1})$ on all interface elements, $\mathbf{c}^{n+1}$ on the large interface elements, and storage of the previous time step h_n. This amounts to storing $N \cdot N_p \cdot (3 \cdot \text{Num(int)} + \text{Num(large int)}) + 1$ double-precision floating-point numbers, where N is the number of equations, N_p is the number of polynomial basis functions, Num(int) is the number of interface elements and Num(large int) is the number of large interface elements. The Runge-Kutta 4 local time-stepping algorithm requires the storage of $\mathbf{c}^n, \mathbf{f}(\mathbf{c}^n), \mathbf{f}(\mathbf{c}^{n-1}), \mathbf{f}(\mathbf{c}^{n-2})$ on all interface elements, $\mathbf{c}^{n+1}$ on the large interface elements, and storage of the previous time steps h_n, h_{n-1}. This amounts to storing $N \cdot N_p \cdot (4 \cdot \text{Num(int)} + \text{Num(large int)}) + 2$ double-precision floating-point numbers. This is not excessive as the number of interface elements is generally small compared to the total number of elements in the mesh.

These time-stepping algorithms preserve the local nature of the discontinuous Galerkin method in the sense that they do not extend the stencil outward in space. The ghost stages use previous time values to compute high order approximations to the solution coefficients. This allows the algorithm to be used in the case where a single element is surrounded by elements from different bins, a property that is particularly useful on unstructured meshes.

Before illustrating the algorithms, we would like to investigate the theoretical maximum-speed up we can achieve by using local time-stepping rather than global time-stepping. Consider the simple example, from [13], of a computational mesh Ω with E spatial elements. The mesh contains elements of two sizes, with two different stable time steps. Assume that a percentage of elements having the small time step Δt_1 is $\alpha < 1$ and the percentage of elements having the large time step Δt_2 is $(1 - \alpha)$. Assume the time steps satisfy $\Delta t_2 = \tau \Delta t_1, \tau > 1$. Let N^{LTS} be the number of space-time elements used in a local

time-stepping method in the space-time domain $\Omega \times T$, where $T = [t_0, t_f]$. Then,

$$N^{LTS} = \alpha E \frac{t_f - t_0}{\Delta t_1} + (1 - \alpha) E \frac{t_f - t_0}{\Delta t_2}. \tag{4.34}$$

For a global time-stepping algorithm, all elements must be advanced with the smaller time-step Δt_1. The number of global space-time elements is

$$N^{GTS} = E \frac{t_f - t_0}{\Delta t_1}. \tag{4.35}$$

We define the theoretical speed up σ to be the ratio of the number of space-time elements in (4.35) and (4.34)

$$\sigma = \frac{N^{GTS}}{N^{LTS}} = \frac{\tau}{1 - \alpha + \alpha\tau}. \tag{4.36}$$

We consider two extreme limits. First, the case where there is great disparity in the two time steps, $\tau \to \infty$. Taking the limit, we obtain

$$\lim_{\tau \to \infty} \sigma = \frac{1}{\alpha}. \tag{4.37}$$

We see that the theoretical speed up is not determined by the time step ratio τ but from the percentage α of elements with the smaller time-step. This means that when there is a small percentage of elements having time step Δt_1 we can expect a greater speed up than if there is a larger percentage elements with the smaller time step. Secondly, we consider the case when there are very few elements with the smaller time step, $\alpha \to 0$. Taking the limit, we obtain

$$\lim_{\alpha \to 0} \sigma = \tau. \tag{4.38}$$

In this case, we see that the theoretical maximum for speed up is given by the time step ratio τ.

To illustrate how the algorithms are implemented, we will present examples of how Runge-Kutta 3 and Runge-Kutta 4 local time-stepping are used on two types of meshes. The examples below have been adapted from [22] to fit the Runge-Kutta 3 and Runge-Kutta 4 local time-stepping algorithms.

First, using the discontinuous Galerkin method for spatial discretization of a one-dimensional scalar equation with two-for-one spacial refinement. The mesh contains elements of size r to the left of the interface and size $r/2$ to the right of the interface. The interface is located between elements j and $j + 1$. The elements of size r will be advanced

with time step h_{n+1}, whereas the elements of size $r/2$ will be advanced with time step $h_{n+1}/2$.

We begin with Runge-Kutta 3 local time-stepping. Suppose that until t_{n-1}, global time stepping has been used. In practice, only one global time step is needed before starting the RK3-LTS algorithm.

Algorithm 1: RK3-LTS

1. Using global time-stepping, advance all elements from $\mathbf{c}^{n-1}$ to $\mathbf{c}^n$ using a globally stable time step h_n. Store $\mathbf{f}(\mathbf{c}^{n-1})$ on elements j and $j+1$.
2. Compute $\mathbf{f}(\mathbf{c}^n)$ on all elements.
3. Compute the first stage on large elements $\mathbf{C}^{(1)}_{large}$ using the local time step h_{n+1}.
4. Compute the first ghost stage on the small interface element $j+1$ using (4.27a).
5. Compute $\mathbf{f}(\mathbf{C}^{(1)}_{large})$ and the second stage on all large elements $\mathbf{C}^{(2)}_{large}$.
6. Compute the second ghost stage on the small interface element $j+1$ using (4.27b).
7. Compute $\mathbf{f}(\mathbf{C}^{(2)}_{large})$ and advance the solution on all large elements to time $t_{n+1} = t_n + h_{n+1}$. Figure 4.4.
8. Store solution value $\mathbf{c}^{n+1}$ on the large interface element j, then set a ghost value equal to $\mathbf{c}^n$ on this element using (4.29a) with $K = 2$ and $k = 0$.
9. Compute the first stage on the first half-step on the small elements $\mathbf{C}^{(1)}_{small,0}$ using $\mathbf{f}(\mathbf{c}^n)$ computed in Step 2, and using the local time-step $h_{n+1}/2$
10. Compute the first ghost stage on the large interface element j using (4.29b) with $K = 2$ and $k = 0$.
11. Compute $\mathbf{f}(\mathbf{C}^{(1)}_{small,0})$ and the second stage on the first half-step on all small elements $\mathbf{C}^{(2)}_{small,0}$.
12. Compute the second ghost stage on the large interface element j using (4.29c) with $K = 2$ and $k = 0$.
13. Compute $\mathbf{f}(\mathbf{C}^{(2)}_{small,0})$ and advance the solution on all small elements to time $t_{n+1/2} = t_n + h_{n+1}/2$, $\mathbf{c}^{n+1/2}_{small}$. Figure 4.5.

14. Using (4.29a) with $K = 2$ and $k = 1$, set the ghost solution value on the large interface element j.

15. Compute $\mathbf{f}(\mathbf{c}_{small}^{n+1/2})$ on all small elements.

16. Compute the first stage on the second half-step on the small elements $\mathbf{C}_{small,0}^{(1)}$ using $\mathbf{f}(\mathbf{c}_{small}^{n+1/2})$ computed in Step 15, and using the local time-step $h_{n+1}/2$

17. Compute the first ghost stage on the large interface element j using (4.29b) with $K = 2$ and $k = 1$.

18. Compute $\mathbf{f}(\mathbf{C}_{small,1}^{(1)})$ and the second stage on the second half-step on all small elements $\mathbf{C}_{small,1}^{(2)}$.

19. Compute the second ghost stage on the large interface element j using (4.29c) with $K = 2$ and $k = 1$.

20. Compute $\mathbf{f}(\mathbf{C}_{small,1}^{(2)})$ and advance the solution on all small elements to time $t_{n+1} = t_{n+1/2} + h_{n+1}/2$. The solution on all small elements have been advanced to time t_{n+1}.

21. Reset the solution on element j to the value stored in Step 8. Figure 4.6.

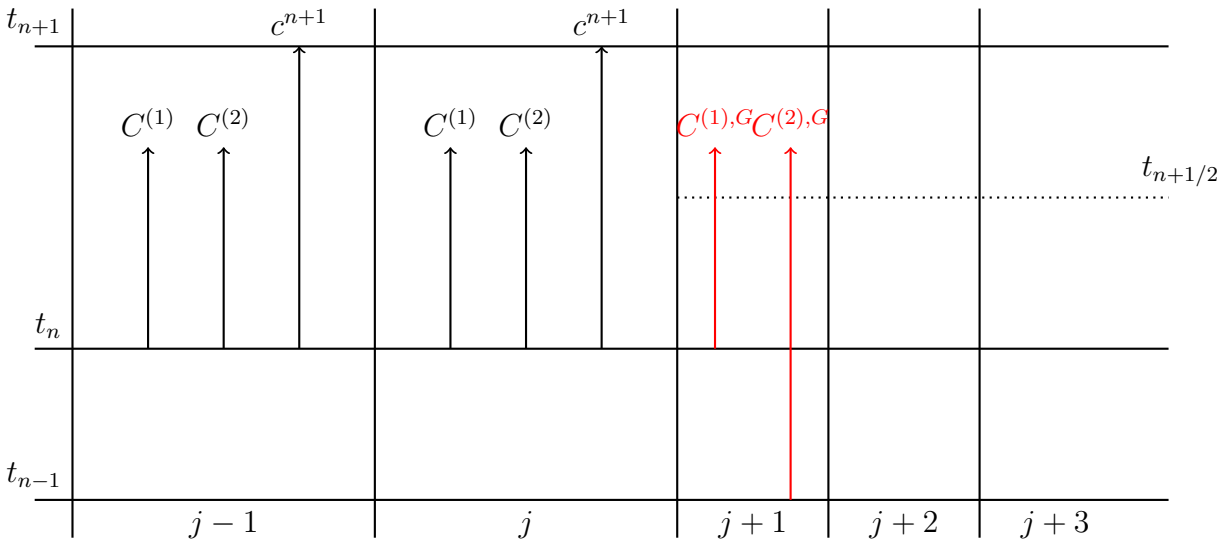


Figure 4.4: Algorithm 1: steps 2-7.

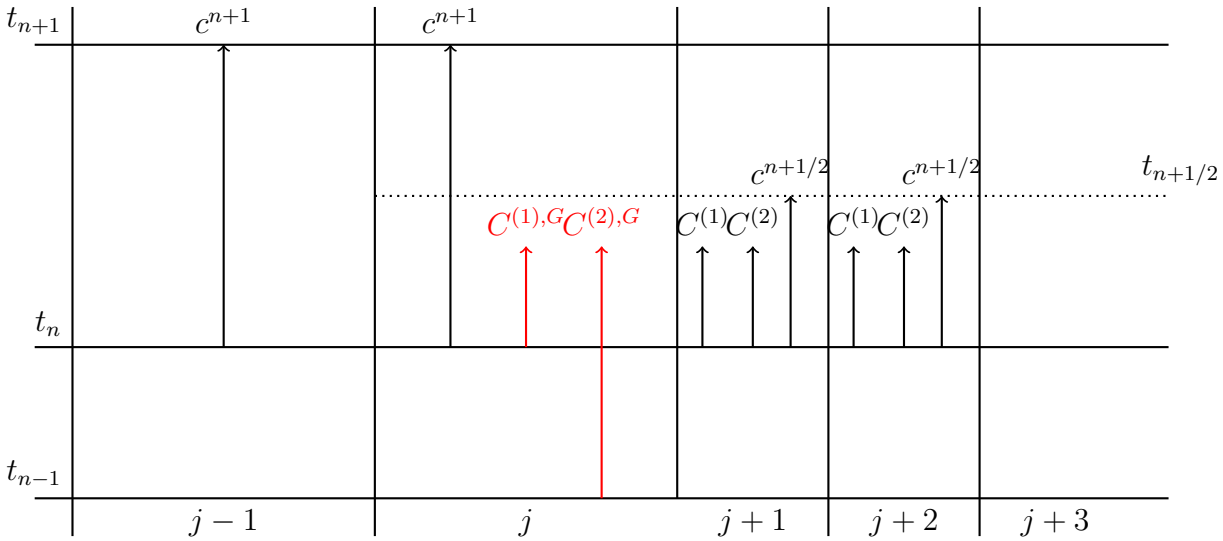


Figure 4.5: Algorithm 1: steps 8-13.

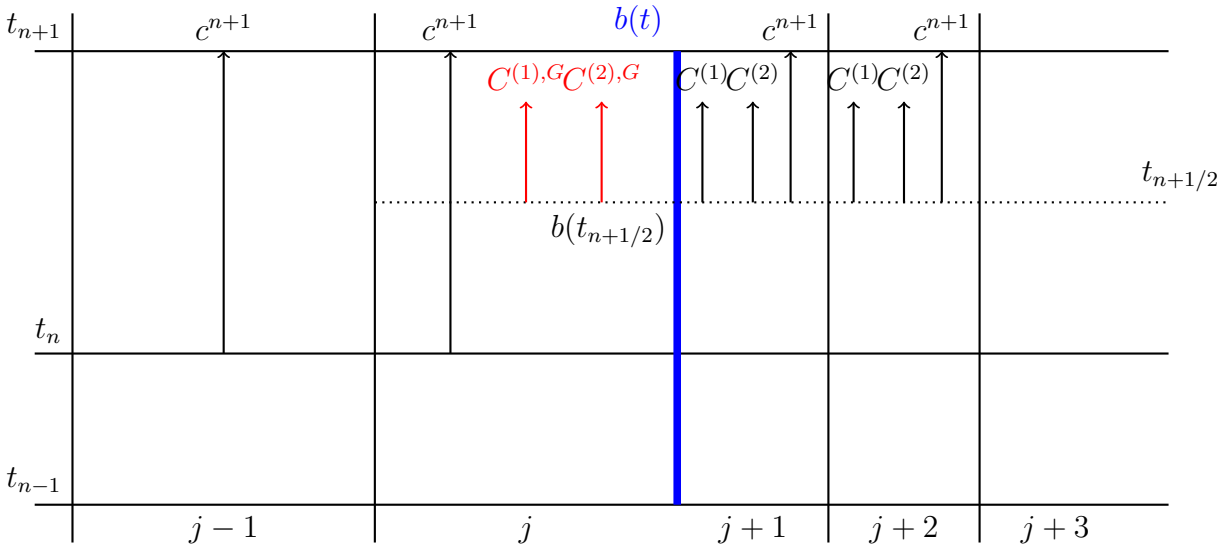


Figure 4.6: Algorithm 1: steps 14-21.

Next we show the Runge-Kutta 4 local time-stepping algorithm on the same problem. Suppose that until t_{n-2}, global time stepping has been used. In practice, only two global time steps are needed before starting the RK4-LTS algorithm. We do not present figures

to demonstrate this algorithm as they are similar to Algorithm 1.

Algorithm 2: RK4-LTS

1. Using global time-stepping, advance all elements from $\mathbf{c}^{n-2}$ to $\mathbf{c}^{n-1}$ using a globally stable time step h_{n-1}. Store $\mathbf{f}(\mathbf{c}^{n-2})$ on elements j and $j+1$.
2. Using global time-stepping, advance all elements from $\mathbf{c}^{n-1}$ to $\mathbf{c}^{n}$ using a globally stable time step h_{n}. Store $\mathbf{f}(\mathbf{c}^{n-1})$ on elements j and $j+1$.
3. Compute $\mathbf{f}(\mathbf{c}^{n})$ on all elements.
4. Compute the first stage on large elements $\mathbf{C}^{(1)}_{large}$ using the local time step h_{n+1}.
5. Compute the first ghost stage on the small interface element $j+1$ using (4.31a).
6. Compute $\mathbf{f}(\mathbf{C}^{(1)}_{large})$ and the second stage on all large elements $\mathbf{C}^{(2)}_{large}$.
7. Compute the second ghost stage on the small interface element $j+1$ using (4.31b).
8. Compute $\mathbf{f}(\mathbf{C}^{(2)}_{large})$ and the third stage on all large elements $\mathbf{C}^{(3)}_{large}$.
9. Compute the third ghost stage on the small interface element $j+1$ using (4.31c).
10. Compute $\mathbf{f}(\mathbf{C}^{(3)}_{large})$ and advance the solution on all large elements to time $t_{n+1} = t_n + h_{n+1}$.
11. Store solution value $\mathbf{c}^{n+1}$ on the large interface element j, then set the ghost value equal to $\mathbf{c}^{n}$ on this element using (4.33a) with $K = 2$ and $k = 0$.
12. Compute the first stage on the first half-step on the small elements $\mathbf{C}^{(1)}_{small,0}$ using $\mathbf{f}(\mathbf{c}^{n})$ computed in Step 3, and using the local time-step $h_{n+1}/2$
13. Compute the first ghost stage on the large interface element j using (4.33b) with with $K = 2$ and $k = 0$.
14. Compute $\mathbf{f}(\mathbf{C}^{(1)}_{small,0})$ and the second stage on the first half-step on all small elements $\mathbf{C}^{(2)}_{small,0}$.
15. Compute the second ghost stage on the large interface element j using (4.33c) with $K = 2$ and $k = 0$.

16. Compute $\mathbf{f}(\mathbf{C}^{(2)}_{small,0})$ and the second stage on the first half-step on all small elements $\mathbf{C}^{(3)}_{small,0}$.

17. Compute the third ghost stage on the large interface element j using (4.33d) with $K = 2$ and $k = 0$.

18. Compute $\mathbf{f}(\mathbf{C}^{(3)}_{small,0})$ and advance the solution on all small elements to time $t_{n+1/2} = t_n + h_{n+1}/2$, $\mathbf{c}^{n+1/2}_{small}$.

19. Using (4.33a) with with $K = 2$ and $k = 1$, set the ghost solution value on the large interface element j.

20. Compute $\mathbf{f}(\mathbf{c}^{n+1/2}_{small})$ on all small elements.

21. Compute the first stage on the second half-step on the small elements $\mathbf{C}^{(1)}_{small,1}$ using $\mathbf{f}(\mathbf{c}^{n+1/2}_{small})$ computed in Step 20, and using the local time-step $h_{n+1}/2$

22. Compute the first ghost stage on the large interface element j using (4.33b) with $K = 2$ and $k = 1$.

23. Compute $\mathbf{f}(\mathbf{C}^{(1)}_{small,1})$ and the second stage on the second half-step on all small elements $\mathbf{C}^{(2)}_{small,1}$.

24. Compute the second ghost stage on the large interface element j using (4.33c) with $K = 2$ and $k = 1$.

25. Compute $\mathbf{f}(\mathbf{C}^{(2)}_{small,1})$ and the third stage on the second half-step on all small elements $\mathbf{C}^{(3)}_{small,1}$.

26. Compute the third ghost stage on the large interface element j using (4.33d) with $K = 2$ and $k = 1$.

27. Compute $\mathbf{f}(\mathbf{C}^{(3)}_{small,1})$ and advance the solution on all small elements to time $t_{n+1} = t_{n+1/2} + h_{n+1}/2$. The solution on all small elements have been advanced to time t_{n+1}.

28. Reset the solution on element j to the value stored in Step 11.

Finally, we extend Algorithms 1 and 2 to meshes with multiple levels of refinement. Suppose the mesh contains elements of size $r, r/2, \ldots, r/2^m$ where there are $m + 1$ levels of refinement. Suppose elements have been grouped into bins based on their size, i.e.,

elements of size $r/2^i$ are grouped into bin(i) which has local time step $h/2^i$. Will illustrate this extension with an example of $m = 3$, or four levels of two-for-one refinement.

Algorithm 3

1. Advance all elements to time t_n using a globally stable time step, storing appropriate data.

2. Advance all elements in time by one locally stable time step, $h/2^i$ using steps 2-13 of Algorithm 1, or 3-18 of Algorithm 2. Begin by advancing the bin of largest elements bin(0), then bin(1), bin(2), and finally bin(3).

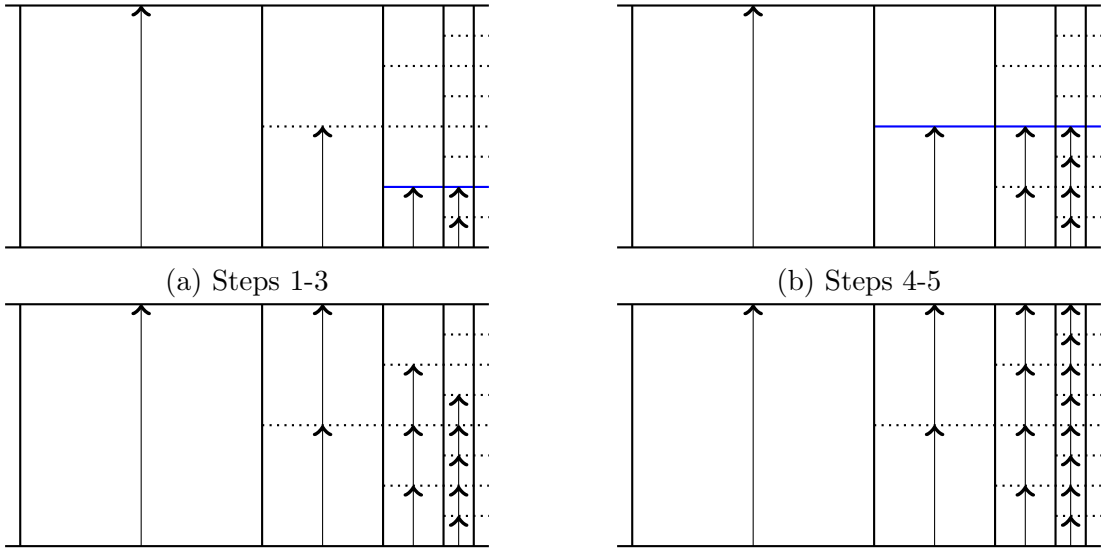
3. Advance elements in bin(3) by one local time step $h/8$ using steps steps 14-21 of Algorithm 1, or 19-28 of Algorithm 2. Now bin(2) and bin(3) are at the same time level, Figure 4.7a.

4. Advance bin(2) and bin(3) with local time steps $h/4$ and $h/8$ using steps 2-13 of Algorithm 1, or 3-18 of Algorithm 2. Store and update appropriate data.

5. Advance elements in bin(3) by one local time step $h/8$ using steps steps 14-21 of Algorithm 1, or 19-28 of Algorithm 2. Now bin(1), bin(2) and bin(3) are at the same time level, Figure 4.7b.

6. Advance bin(1), bin(2) and bin(3) with local time steps $h/2$, $h/4$, and $h/8$ using steps 2-13 of Algorithm 1, or 3-18 of Algorithm 2. Store and update appropriate data, , Figure 4.7c.

7. Repeat steps 3-5. All bins are advanced to t_{n+1}, Figure 4.7d.



(a) Steps 1-3

(b) Steps 4-5

(c) Step 6

(d) Step 7

Figure 4.7: Algorithm 3.

Chapter 5

Numerical Results

In this section we provide a number of numerical examples demonstrating the accuracy and efficiency of the algorithms proposed in Section 4. We use a discontinuous Galerkin spatial discretization, as described in Section 3, with a second degree polynomial basis coupled with the Runge-Kutta 3 local time-stepping algorithm and a third degree polynomial basis coupled with the Runge-Kutta 4 local time-stepping algorithm. The time step for each problem is chosen with the rule

$$\Delta t = C \cdot \frac{S}{\lambda_{max}} \cdot \frac{1}{2p+1}, \tag{5.1}$$

where S is some measure of the element size, λ_{max} is the maximum wave speed in the computational domain, p is the degree of polynomial basis and C is a safety factor so that we do not violate the CFL condition. In one-dimension $S = \Delta x$ and in two-dimensions $S = r_{max}/2$, where r_{max} is the maximum inscribed radius of the elements in the mesh as described in Section 4.3. Unless otherwise stated, we take $C = 0.9$.

5.1 One-Dimensional Examples

5.1.1 One-Dimensional Linear Advection

In the first example, we solve the linear advection equation

$$\begin{aligned} u_t + u_x &= 0, \quad t > 0, \\ u(x,0) &= \sin(\pi x), \end{aligned} \tag{5.2}$$

on the interval $\Omega = [-1, 1]$ with periodic boundary conditions $u(-1, t) = u(1, t)$. The domain is divided into two subdomains $[-1, 0]$ and $[0, 1]$. The left subdomain is unrefined with element size Δx. The right subdomain is considered refined with element sizes $\Delta x/2$ corresponding to two-for-one refinement and $\Delta x/4$ corresponding to four-for-one refinement. Due to the periodic boundary conditions, the two subdomains will be separated by two interfaces: at $x = 0$, where the wave moves from the unrefined region into the refined region and at $x = \pm 1$, where the wave moves from the refined region into the unrefined region. The errors and rates of convergence in the L^2 and max norms are presented in Table 5.1 and Table 5.2 after five full rotations, i.e, at time $t = 10$. The max norm is computed using 10 uniformly distributed points within each element.

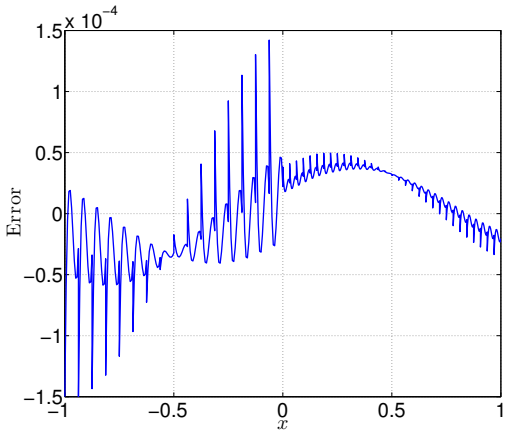
	2-for-1 Refinement				4-for-1 Refinement			
Δx	$\|e\|_2$	r	$\|e\|_\infty$	r	$\|e\|_2$	r	$\|e\|_\infty$	r
2^{-3}	4.51 e-04	-	1.20 e-03	-	5.05 e-04	-	1.17 e-03	-
2^{-4}	5.50 e-05	3.04	1.50 e-04	3.00	6.14 e-05	3.04	1.47 e-04	2.99
2^{-5}	6.84 e-06	3.01	1.89 e-05	2.99	7.64 e-06	3.01	1.85 e-05	2.99
2^{-6}	8.55 e-07	3.00	2.38 e-06	2.99	9.55 e-07	3.00	2.31 e-06	3.00
2^{-7}	1.07 e-07	3.00	2.95 e-07	3.01	1.19 e-07	3.00	2.90 e-07	3.00

Table 5.1: Errors in the L^2 and max norms with rates of convergence for Runge-Kutta 3 local time-stepping in one-dimension.

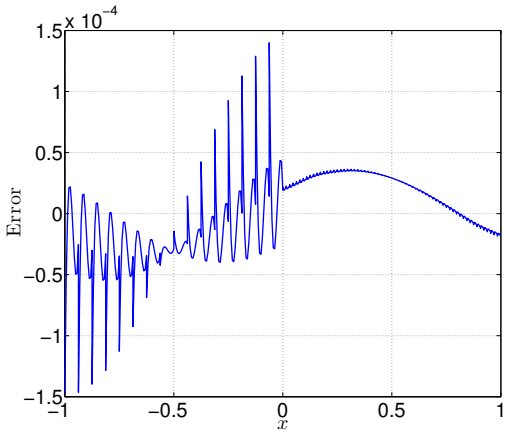
	2-for-1 Refinement				4-for-1 Refinement			
Δx	$\|e\|_2$	r	$\|e\|_\infty$	r	$\|e\|_2$	r	$\|e\|_\infty$	r
2^{-3}	4.06 e-06	-	2.88 e-05	-	5.30 e-06	-	2.87 e-05	-
2^{-4}	2.53 e-07	4.00	1.83 e-06	3.98	3.32 e-07	4.00	1.82 e-06	3.98
2^{-5}	1.59 e-08	3.99	1.15 e-07	3.99	2.08 e-08	4.00	1.15 e-07	3.99
2^{-6}	9.97 e-10	3.99	7.19 e-09	4.00	1.30 e-09	4.00	7.17 e-09	4.00
2^{-7}	6.30 e-11	3.98	4.50 e-10	4.00	8.23 e-11	3.98	4.63 e-10	3.95

Table 5.2: Errors in the L^2 and max norms with rates of convergence for Runge-Kutta 4 local time-stepping in one-dimension with $C = 0.65$.

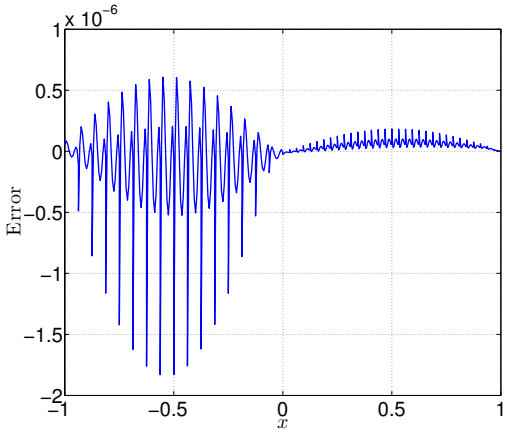
We achieve the theoretical $p + 1$ order of convergence of the discontinuous Galerkin method with both two-for-one refinement and four-for-one refinement. The Runge-Kutta 4 simulations were computed with $C = 0.65$. The reduced safety factor for Runge-Kutta 4 was to ensure convergence in the max-norm, and indicates that there is a reduced stability region for this method.



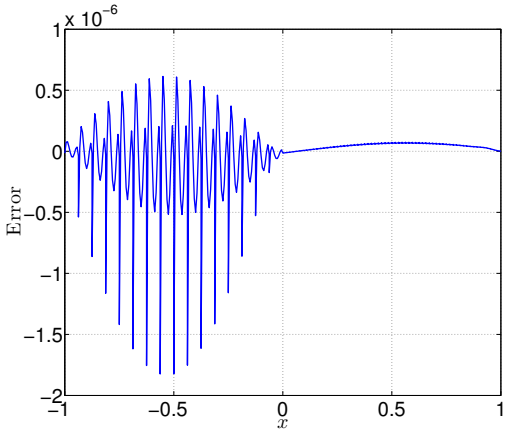
(a) Runge-Kutta 3, 2-for-1 refinement.



(b) Runge-Kutta 3, 4-for-1 refinement.



(c) Runge-Kutta 4, 2-for-1 refinement.



(d) Runge-Kutta 4, 4-for-1 refinement.

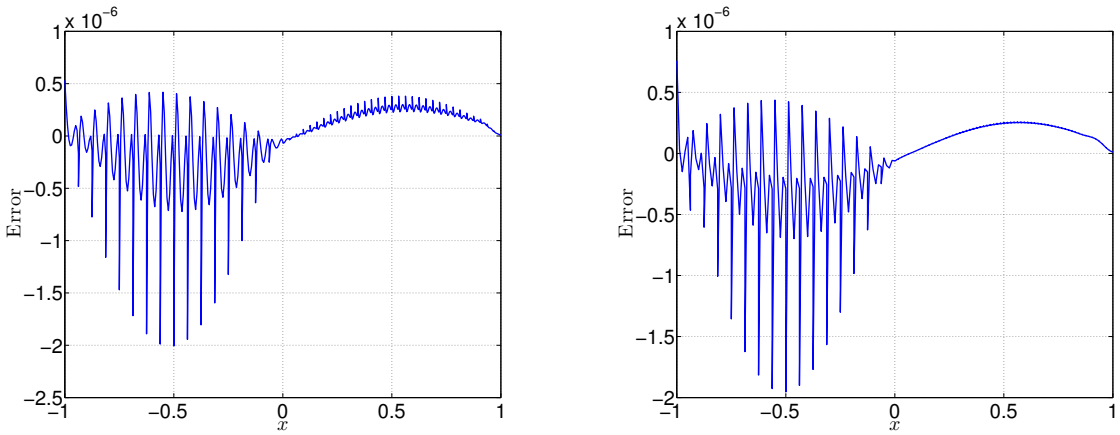
Figure 5.1: Point-wise error for 2-for-1 (left) and 4-for-1 (right) refinements at $t = 10$ for Runge-Kutta 3 local time-stepping (top) and Runge-Kutta 4 local time-stepping (bottom). The unrefined region of the domain is discretized with $\Delta x = 1/16$.

In Figure 5.1, the point-wise errors of the obtained solutions are plotted. We see that there are no oscillations or errors produced on the interface elements. When using the Runge-Kutta 4 local time-stepping method, small errors appear on the inflow edge of the large interface element when information is traveling from the small elements, this can be seen in Figure 5.2. Errors and rates of convergence in the L^2 and max norms with a safety factor of $C = 0.9$ are presented in Table 5.3. We see that these errors have a large effect on

the rate of convergence in the max norm. Reducing the time-step suppresses these errors so that they do not exceed the maximum error produced on the interior elements.

	2-for-1 Refinement				4-for-1 Refinement			
Δx	$\|e\|_2$	r	$\|e\|_\infty$	r	$\|e\|_2$	r	$\|e\|_\infty$	r
2^{-3}	6.74 e-06	-	3.12 e-05	-	8.58 e-06	-	3.12 e-05	-
2^{-4}	4.08 e-07	4.05	2.01 e-06	3.96	5.04 e-07	4.10	1.95 e-06	4.00
2^{-5}	2.56 e-08	3.99	1.27 e-07	3.98	3.16 e-08	3.99	1.25 e-07	3.96
2^{-6}	1.62 e-09	3.98	1.03 e-08	3.62	2.02 e-09	3.97	1.27 e-08	3.30
2^{-7}	1.03 e-10	3.98	1.23 e-09	3.06	1.34 e-10	3.91	1.99 e-09	2.67

Table 5.3: Errors in the L^2 and max norms with rates of convergence for Runge-Kutta 4 local time-stepping in one-dimension with $C = 0.9$.



(a) 2-for-1 refinement.

(b) 4-for-1 refinement.

Figure 5.2: Point-wise error for 2-for-1 (left) and 4-for-1 (right) refinements at $t = 10$ for Runge-Kutta 4 local time-stepping with $C = 0.9$. The unrefined region of the domain is discretized with $\Delta x = 1/16$. Note the increased error on the interface boundary at $x = -1$.

5.2 Two-Dimensional Examples

In the two-dimensional examples we use unstructured triangular meshes where the meshes do not contain adjacent elements which differ by more than the factor of refinement, which

we have set to two.

5.2.1 Two-Dimensional Advection

For the first two-dimensional test problem, we consider the classical rotating hill advection problem given by

$$u_t - (2\pi y)u_x + (2\pi x)u_y = 0, \tag{5.3a}$$

$$u(x, y, 0) = \alpha \exp(-((x - x_0)^2 + (y - y_0)^2)/2r^2), \tag{5.3b}$$

with parameters $\alpha = 5, r = 0.15, x_0 = 0.2, y_0 = 0$, on the domain $\Omega = [-1, 1] \times [-1, 1]$. Along the boundary of the domain, we enforce the exact solution

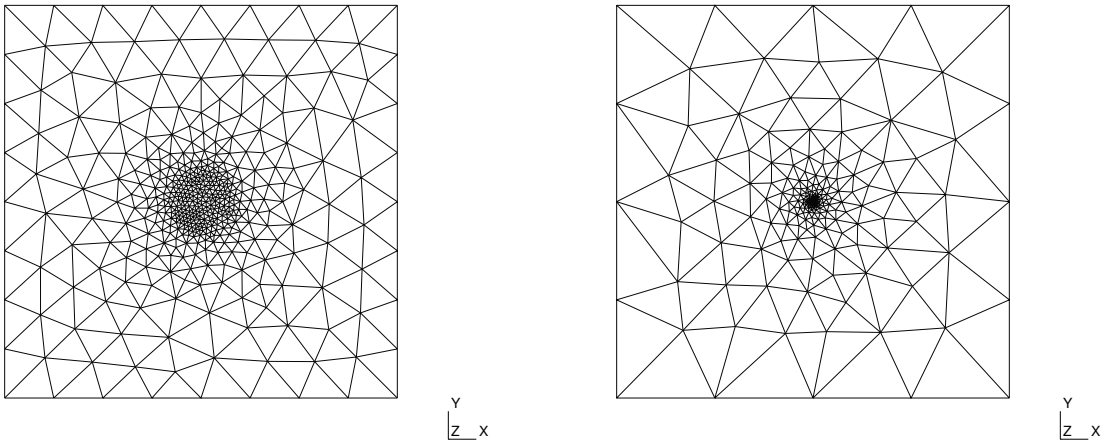
$$u(x, y, t) = \alpha \exp\Big(- ((x\cos(2\pi t) + y\sin(2\pi t) - x_0)^2 + (-x\sin(2\pi t) + y\sin(2\pi t) - y_0)^2)/2r^2\Big). \tag{5.3c}$$

We solve the rotating hill problem on two sets of meshes. Both meshes have an attractor point at the center of the domain which reduces the size of the elements in the vicinity of the point. The first mesh, Mesh 1, has four levels of refinement and the second mesh, Mesh 2, has nine levels of refinement. Mesh 1-A begins with 846 elements and Mesh 2-A begins with 616 elements, they are shown in Figure 5.3. Each successive mesh, B-D, is created through refinement by splitting where each element is split into four smaller elements, quadrupling the number of elements in the mesh.

In Tables 5.4 and 5.5, we present the errors and rates of convergence in the L^2 and max norms at time $t = 1$, when the pulse has completed one full rotation around the origin. We see that on both sets of meshes we achieve the theoretical $p + 1$ rate of convergence. We note that the reduced CFL is not needed for the Runge-Kutta 4 local time-stepping algorithm in two-dimensions. This can be explained by the fact that using the radius of the inscribed circle on an element gives an overly conservative measure of the element size. In fact, it has been shown that using the radius of the inscribed circle as a measure of element size in the time step calculation is nearly a factor of $\sqrt{2}$ smaller than the maximum stable time step of the problem [5].

5.2.2 Shallow Water Equations

The shallow water equations are an approximation to the Navier-Stokes equations which result from assuming that the vertical length scale of the fluid is much smaller than hori-



(a) Mesh 1-A (b) Mesh 2-A

Figure 5.3: Nonuniform meshes used for the rate of convergence studies.

	RK3-LTS				RK4-LTS			
Mesh	$\|e\|_2$	r	$\|e\|_\infty$	r	$\|e\|_2$	r	$\|e\|_\infty$	r
Mesh 1-A	5.12 e-03	-	3.58 e-03	-	5.35 e-04	-	3.85 e-04	-
Mesh 1-B	5.57 e-04	3.20	3.44 e-04	3.38	3.97 e-05	3.75	2.56 e-05	3.91
Mesh 1-C	5.94 e-05	3.23	3.45 e-05	3.32	2.11 e-06	4.23	1.33 e-06	4.27
Mesh 1-D	6.56 e-06	3.17	3.74 e-06	3.20	1.26 e-07	4.07	7.60 e-08	4.12

Table 5.4: Errors in the L^2 and max norms with rates of convergence for Runge-Kutta 3 and Runge-Kutta 4 local time-stepping on Mesh 1.

zontal length scale. Written in terms of the conserved variables $\mathbf{u} = (h, uh, vh)^T$, they are given by

$$\frac{\partial}{\partial t}\begin{pmatrix} h \\ uh \\ vh \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} uh \\ u^2h + \frac{1}{2}gh^2 \\ uvh \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} vh \\ uvh \\ v^2h + \frac{1}{2}gh^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{5.4a}$$

where h is the height of the fluid, u, v are the velocities in the x and y directions, and g is the acceleration due to gravity. See [26] for a full derivation of (5.4a). We solve (5.4a)

	RK3-LTS				RK4-LTS			
Mesh	$\|e\|_2$	r	$\|e\|_\infty$	r	$\|e\|_2$	r	$\|e\|_\infty$	r
Mesh 2-A	1.72 e-02	-	1.16 e-02	-	2.55 e-03	-	2.01 e-03	-
Mesh 2-B	2.14 e-03	3.01	1.40 e-03	3.06	1.83 e-04	3.80	1.40 e-04	3.84
Mesh 2-C	1.94 e-04	3.47	1.26 e-04	3.47	1.09 e-05	4.07	7.54 e-06	4.21
Mesh 2-D	2.02 e-05	3.26	1.28 e-05	3.30	5.89 e-07	4.21	3.93 e-07	4.26

Table 5.5: Errors in the L^2 and max norms with rates of convergence for Runge-Kutta 3 and Runge-Kutta 4 local time-stepping on Mesh 2.

with the initial conditions

$$\begin{pmatrix} h \\ uh \\ vh \end{pmatrix} = \begin{pmatrix} 10 + 5\exp\left(-((x-x_0)^2 + (y-y_0)^2)/2r^2\right) \\ 0 \\ 0 \end{pmatrix}, \tag{5.4b}$$

on the domain $[-1,1]\times[-1,1]$, where $x_0 = 0.25, y_0 = 0.25$ and $r = 0.1$. Along the boundary we enforce reflecting boundary conditions and solve until a final time of $t_f = 0.1$. The reflected boundary conditions are implemented as follows. Let $\mathbf{u}_j$ be a boundary element and let $\mathbf{n} = (n_x, n_y)$ be the outward facing normal along the boundary edge. Let $\mathbf{u}_+$ be the ghost solution used in the calculation of the surface integral 4.25. The value of $\mathbf{u}_+$ is given by

$$\begin{pmatrix} h_+ \\ (uh)_+ \\ (vh)_+ \end{pmatrix} = \begin{pmatrix} h_j \\ (uh)_j - 2n_x((uh)_j n_x + (vh)_j n_y) \\ (vh)_j - 2n_y((uh)_j n_x + (vh)_j n_y) \end{pmatrix}. \tag{5.4c}$$

We solve the shallow water equations on a mesh containing very few small elements. The small elements are concentrated in a region near the origin, elsewhere the mesh is uniform in size (uniform in the sense that the elements have size r satisfying $r_{max}/2 \leq r \leq r_{max}$, where r_{max} is the size of the largest element in the mesh). We separate elements into bins in the way described in Section 4.3. The mesh contains 26329 elements in Bin(0), 109 elements in Bin(1), 68 elements in Bin(2), 58 elements in Bin(3), and 16 elements in Bin(4). The few elements in Bin(4) will be very influential on the globally stable time step and so, there will be a high maximum theoretical speed-up.

Let α_i be the percentage of mesh elements contained in Bin(i), let E be the total number of elements in the mesh, let Δt_{min} be the globally stable time step calculated based on the smallest element size and let Δt_{max} be the time step calculated with the largest element size. Recall from Section 4.3 that Bin(i) will be advanced with time step

$\Delta t_{max}/2^{i+1}$. Then, the maximum theoretical speed-up σ_T is given by

$$\sigma_T = \frac{E\frac{t_f}{\Delta t_{min}}}{\alpha_0 E\frac{t_f}{\Delta t_{max/2}} + \alpha_1 E\frac{t_f}{\Delta t_{max/4}} + \alpha_2 E\frac{t_f}{\Delta t_{max/8}} + \alpha_3 E\frac{t_f}{\Delta t_{max/16}} + \alpha_4 E\frac{t_f}{\Delta t_{max/32}}} \tag{5.5}$$

$$= \frac{\Delta t_{max}}{\Delta t_{min}} \cdot \frac{1}{2\alpha_0 + 4\alpha_1 + 8\alpha_2 + 16\alpha_3 + 32\alpha_4}. \tag{5.6}$$

Assuming the difference in calculating the maximum wave speed for Δt_{min} and Δt_{max} is negligible, we have

$$\sigma_T = \frac{r_{max}}{r_{min}} \cdot \frac{1}{2\alpha_0 + 4\alpha_1 + 8\alpha_2 + 16\alpha_3 + 32\alpha_4} \tag{5.7}$$

$$= 10.82. \tag{5.8}$$

In Table 5.6, we compare the CPU time (in seconds) to run the simulations with local time-stepping against global time-stepping. In the implementation of the local time-stepping algorithms, before the time-stepping begins, we sort the edge list by bin. For comparing these methods, we record the CPU time with edge sorting as well as the CPU time starting after the edges have been sorted. These are labeled LTS and LTS-NS, respectively. The CPU time for global time-stepping is labeled GTS. We denote the numerical speed-up by σ_N^1, σ_N^2 for the numerical speed-up, with and without sorting.

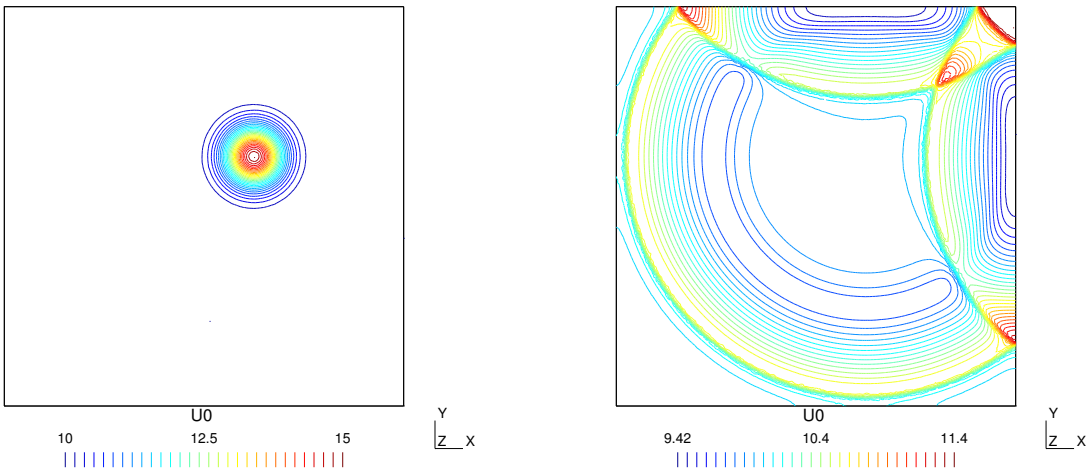
	LTS	LTS - NS	GTS	σ_T	σ_N^1	σ_N^2	σ_N^1/σ_T	σ_N^2/σ_T
RK3	255.33	254.35	2707.88	10.82	10.61	10.65	0.98	0.98
RK4	988.85	987.83	10284.42	10.82	10.40	10.41	0.96	0.96

Table 5.6: Computational time for the shallow water simulations.

We see that with the numerical speed-up of our local time-stepping methods are within 2% of the maximum theoretical speed-up with the Runge-Kutta 3 local time-stepping algorithm and within 4% of the maximum theoretical speed-up with the Runge-Kutta 4 local time-stepping algorithm. This suggests that the algorithms have little over-head on this problem. Additionally, we have plotted the height h of the numerical solutions in Figures 5.4 and 5.5. We see that there are no numerical artifacts produced as the wave moves through the refined region at the origin.

5.2.3 The Euler Equations

The Euler equations are a nonlinear system of hyperbolic conservation laws that are used to describe the dynamics of a compressible fluid that is inviscid and isotropic. These equations



(a) $t = 0$ (b) $t = 0.1$

Figure 5.4: Height of the shallow water equations solved using Runge-Kutta 3 local time-stepping.

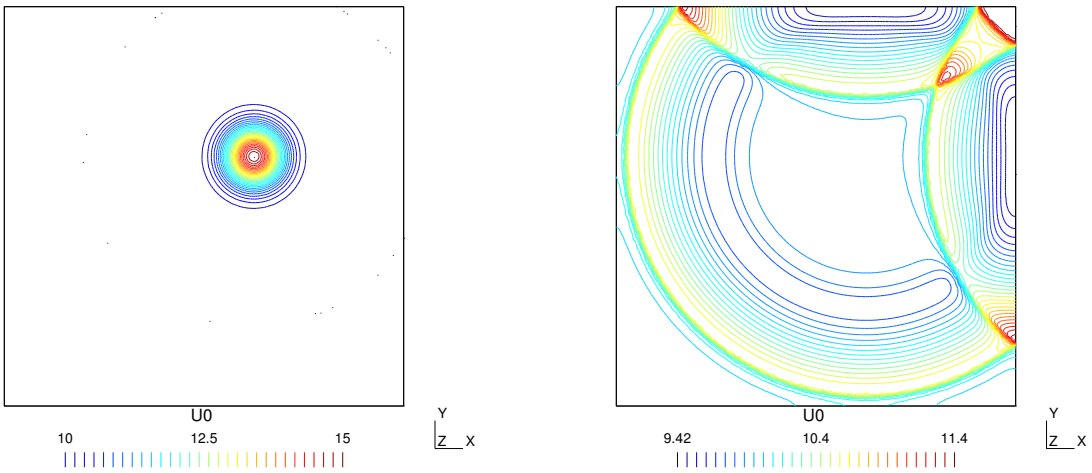
are derived from the physical laws of conservation of mass, momentum and energy, see [26]. The physical variables are the mass density ρ, the x-velocity u, the y-velocity v, and the pressure P. Given in terms of the conserved variables ρ, the x-momentum ρu, the y-momentum ρv, and the total energy per unit mass E, the two-dimensional Euler equations are

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2+P\\ \rho uv\\ u(E+P)\end{pmatrix}+\frac{\partial}{\partial y}\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2+P\\ v(E+P)\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 0\\ 0\end{pmatrix}. \tag{5.9}$$

We see in (5.9) that there are four equations in five unknows. To close the system we use the equation of state

$$P=(\gamma-1)\left(E-\frac{\rho\sqrt{u^2+v^2}}{2}\right) \tag{5.10}$$

to compute the pressure P, where γ is an adiabatic constant. For air, we choose $\gamma = 1.4$.



(a) $t = 0$ (b) $t = 0.1$

Figure 5.5: Height of the shallow water equations solved using Runge-Kutta 4 local time-stepping.

Smooth Vortex Problem

In this problem, we simulate a smooth vortex moving upwards through the plane. The domain is $[-10, 10] \times [-10, 10]$, with initial conditions given in the physical variables

$$\rho_0 = \left(1 - (\gamma - 1)\frac{(SM)^2}{8\pi^2} e^{\frac{1-x^2-y^2}{R^2}}\right)^{\frac{1}{\gamma-1}}, \tag{5.11a}$$

$$u_0 = \frac{Sy}{2\pi R} e^{\frac{1-x^2-y^2}{2R^2}}, \tag{5.11b}$$

$$v_0 = 1 - \frac{Sx}{2\pi R} e^{\frac{1-x^2-y^2}{2R^2}}, \tag{5.11c}$$

$$P_0 = \frac{1}{\gamma M^2}\rho_0^\gamma, \tag{5.11d}$$

where $S = 13.5, M = 0.4, R = 1.5$. Along the boundaries, we apply constant boundary conditions

$$\begin{pmatrix} \rho \\ u \\ v \\ P \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ \frac{1}{\gamma M^2} \end{pmatrix}. \tag{5.11e}$$

The domain is meshed in such a way that along the line connecting the point $(-10, 2.5)$ with the point $(10, 2.5)$ we have a line of refined elements, see Figure 5.6. The computational mesh is more dense than the mesh shown in Figure 5.6. It is obtained by splitting each element in this mesh into four smaller elements. This is done twice resulting in a mesh with sixteen times more elements than seen in Figure 5.6.

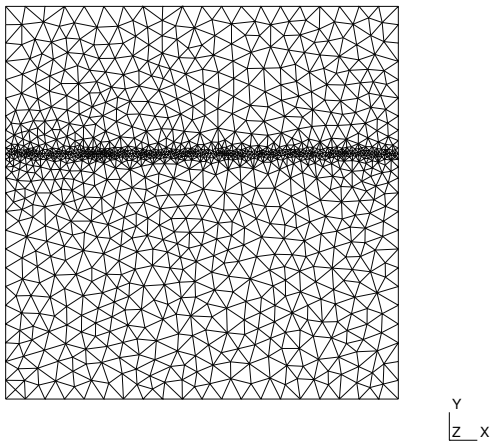
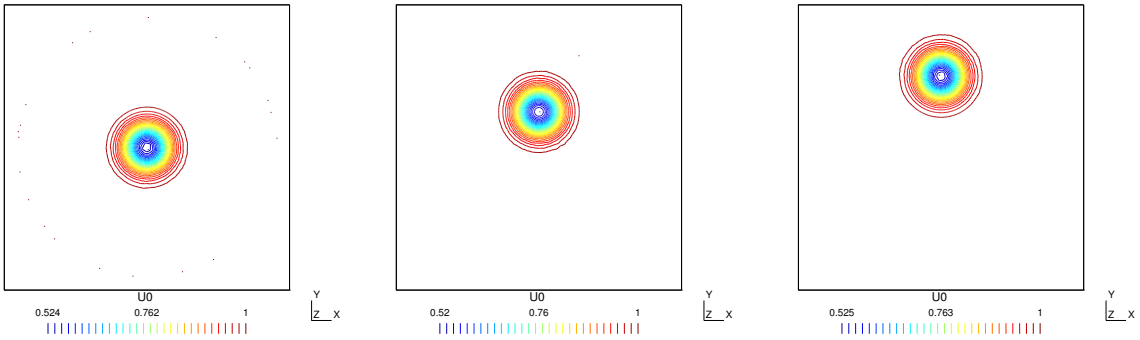


Figure 5.6: Nonuniform mesh for solving the smooth vortex problem.

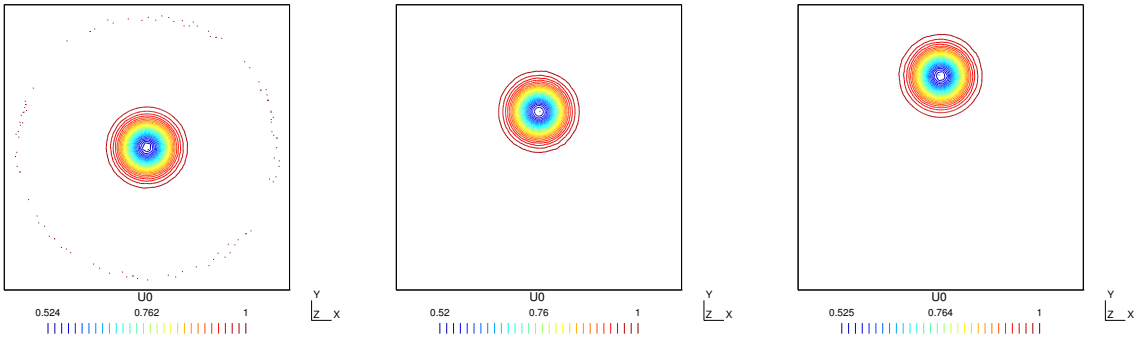
We plot the numerical solution of the density ρ at three time points $t = 0, 2.5, 5$, see Figures 5.7, 5.8. At $t = 0$, the refined region is completely above the vortex, Figures 5.7a, 5.8a; at $t = 2.5$, the center of the vortex is located along the line connecting the point $(-10, 2.5)$ with the point $(10, 2.5)$, Figures 5.7b, 5.8b; at $t = 5$ the vortex has completely passed through the refined region, Figures 5.7b, 5.8b.

We see no qualitative change in the solution as it passes through the refined area, that is, there is no artificial reflected wave or other mesh artifacts produced along the interface between large and small elements.



(a) $t = 0$ (b) $t = 2.5$ (c) $t = 5$

Figure 5.7: Density plots for the smooth vortex problem with Runge-Kutta 3 local time-stepping.



(a) $t = 0$ (b) $t = 2.5$ (c) $t = 5$

Figure 5.8: Density plots for the smooth vortex problem with Runge-Kutta 4 local time-stepping.

5.3 Discussion

We see in Sections 5.1.1 and 5.2.1 that the theoretical $p + 1$ rate of convergence of the discontinuous Galerkin method in the L^2 and max norms is achieved for linear problems in both one- and two-dimensions. For nonlinear problems that do not require limiting, numerical solutions are oscillation free when advanced with their locally stable time-step.

Furthermore, we see that when solutions pass through refined regions there are no artificial waves being reflected. The numerical examples presented in Sections 5.1 and 5.2 provide evidence that these local time-stepping algorithms are stable but indicate that there is a reduced stability region for the Runge-Kutta 4 local time-stepping algorithm.

In Appendix D, the mesh data for each two-dimensional problem are given. For each mesh we give the number of elements in each bin, and the number of interior, large and small interface elements in each bin. Similar to Section 5.2.2, we calculate the numerical speed-up given by the local time-stepping methods for each problem and each mesh. They are given in Table 5.7 for Runge-Kutta 3 and Table 5.8 for Runge-Kutta 4. We see that the numerical speed-up is a high percentage of the maximum theoretical speed-up. Runge-Kutta 3 local time-stepping gets closer to the maximum theoretical speed-up due to the reduced overhead compared to Runge-Kutta 4 local time-stepping. Overall, these methods produce numerical speed-ups close to the theoretical maximum on a variety of meshes and problems.

Mesh	LTS	LTS - NS	GTS	σ_T	σ_N^1	σ_N^2	σ_N^1/σ_T	σ_N^2/σ_T
Mesh 1-A	11.89	11.89	16.70	1.46	1.40	1.40	0.96	0.96
Mesh 1-B	94.75	94.73	136.30	1.46	1.44	1.44	0.98	0.98
Mesh 1-C	773.27	773.07	1130.39	1.46	1.46	1.46	1.00	1.00
Mesh 1-D	6270.04	6266.67	9140.31	1.46	1.46	1.46	1.00	1.00
Mesh 2-A	50.89	50.89	110.09	2.45	2.16	2.16	0.88	0.88
Mesh 2-B	411.10	411.09	937.71	2.45	2.37	2.37	0.98	0.98
Mesh 2-C	3344.83	3344.71	7813.16	2.45	2.34	2.34	0.96	0.96
Mesh 2-D	27526.52	27524.66	64493.51	2.45	2.34	2.34	0.96	0.96
Shallow W.	255.33	254.35	2707.88	10.82	10.61	10.65	0.98	0.98
Vortex	430.77	430.64	927.39	2.39	2.15	2.15	0.90	0.90

Table 5.7: Numerical speed-up of the Runge-Kutta 3 local time-stepping algorithm for all two-dimensional examples.

Mesh	LTS	LTS - NS	GTS	σ_T	σ_N^1	σ_N^2	σ_N^1/σ_T	σ_N^2/σ_T
Mesh 1-A	48.39	48.39	65.61	1.46	1.36	1.36	0.93	0.93
Mesh 1-B	403.36	403.35	564.78	1.46	1.40	1.40	0.96	0.96
Mesh 1-C	3390.77	3390.55	4788.49	1.46	1.41	1.41	0.97	0.97
Mesh 1-D	26899.72	26896.27	38840.11	1.46	1.44	1.44	0.99	0.99
Mesh 2-A	194.64	194.64	407.30	2.45	2.09	2.09	0.85	0.85
Mesh 2-B	1693.02	1693.02	3746.32	2.45	2.21	2.21	0.90	0.90
Mesh 2-C	14245.02	14244.91	32479.94	2.45	2.28	2.28	0.93	0.93
Mesh 2-D	116974.82	116973.10	268871.65	2.45	2.30	2.30	0.94	0.94
Shallow W.	988.85	987.83	10284.42	10.82	10.40	10.41	0.96	0.96
Vortex	1764.67	1764.54	3573.13	2.39	2.03	2.03	0.85	0.85

Table 5.8: Numerical speed-up of the Runge-Kutta 4 local time-stepping algorithm for all two-dimensional examples.

Chapter 6

Conclusion

We have presented two Runge-Kutta based local time-stepping methods based off of a third order Runge-Kutta method and the classical fourth order Runge-Kutta method. Mesh elements were sorted into bins by size, based on the element size relative to the the maximum element. Bins were advanced by their locally stable time-step in order from the bin containing the largest elements to the bin containing the smallest elements. Intermediate stages of the large interface elements were computed by approximating the stages of the small interface elements. Current and previous solution values were used in the approximations of the small interface element stages. Next, on each large interface element, an interpolating polynomial was constructed. This polynomial approximated the numerical solution along the large interface with the accuracy of the underlying Runge-Kutta method. To advance the small elements with their locally stable time step, the interpolating polynomial was imposed as a boundary condition for the small elements along the interface.

We showed that these local time-stepping methods support an arbitrary level and depth of refinement while maintaining the order of accuracy of the underlying Runge-Kutta method in the L^2 and max norms. We provided evidence that they are numerically stable and do not produce numerical artifacts while transitioning from coarse to refined regions, or vice versa. A formal justification for the stability of these methods is needed as well as determining whether the standard Runge-Kutta discontinuous Galerkin CFL condition is suitable.

These methods have been shown to work well on static meshes with refined regions but it is not evident how they can be paired with adaptive mesh refinement where interface elements will be changing as the mesh is refined and coarsened throughout the simulation.

Determining interface element approximations that do not require past information is one area of future study that will allow for pairing with adaptive mesh refinement.

Other areas of future work include generalizing these methods to higher order explicit Runge-Kutta methods as well as developing local time-stepping methods for low-storage Runge-Kutta methods. Efficiently implementing local time-stepping methods into parallel platforms such as graphics processing units or CPU clusters to solve large scale problems is another attractive area of work.

References

[1] Uri M Ascher and Linda R Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, volume 61. Siam, 1998.

[2] Marsha J Berger and Phillip Colella. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[3] Marsha J Berger and Joseph Oliger. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[4] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008.

[5] Noel Chalmers. *Superconvergence, Superaccuracy, and Stability of the Discontinuous Galerkin Finite Element Method*. PhD thesis, University of Waterloo, 2015.

[6] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. The Runge–Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws IV: The Multidimensional Case. *Mathematics of Computation*, 54(190):545–581, 1990.

[7] Bernardo Cockburn, San-Yih Lin, and Chi-Wang Shu. TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws III: One-Dimensional Systems. *Journal of Computational Physics*, 84(1):90–113, 1989.

[8] Bernardo Cockburn and Chi-Wang Shu. TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework. *Mathematics of computation*, 52(186):411–435, 1989.

[9] Bernardo Cockburn and Chi-Wang Shu. The Runge-Kutta Local Projection P^1-Discontinuous Galerkin Finite Element Method for Scalar Conservation Laws. *RAIRO-Modélisation mathématique et analyse numérique*, 25(3):337–361, 1991.

[10] Bernardo Cockburn and Chi-Wang Shu. The Runge–Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[11] Bernardo Cockburn and Chi-Wang Shu. Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *Journal of scientific computing*, 16(3):173–261, 2001.

[12] Moshe Dubiner. Spectral Methods on Triangles and Other Domains. *Journal of Scientific Computing*, 6(4):345–390, 1991.

[13] Michael Dumbser, Martin Käser, and Eleuterio F Toro. An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes-V. Local Time Stepping and p-Adaptivity. *Geophysical Journal International*, 171(2):695–717, 2007.

[14] DA Dunavant. High Degree Efficient Symmetrical Gaussian Quadrature Rules for the Triangle. *International journal for numerical methods in engineering*, 21(6):1129–1148, 1985.

[15] Abdelaâziz Ezziani and Patrick Joly. Local Time Stepping and Discontinuous Galerkin Methods for Symmetric First Order Hyperbolic Systems. *Journal of Computational and Applied Mathematics*, 234(6):1886–1895, 2010.

[16] Gregor J Gassner, Florian Hindenlang, and Claus-Dieter Munz. A Runge-Kutta Based Discontinuous Galerkin Method with Time Accurate Local Time Stepping. *Advances in Computational Fluid Dynamics*, 2:95–118, 2011.

[17] Marcus J Grote, Michaela Mehlin, and Teodora Mitkova. Runge–Kutta-Based Explicit Local Time-Stepping Methods for Wave Propagation. *SIAM Journal on Scientific Computing*, 37(2):A747–A775, 2015.

[18] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2008.

[19] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Texts in Applied Mathematics. Springer New York, 2007.

[20] Alex Kanevsky, Mark H Carpenter, David Gottlieb, and Jan S Hesthaven. Application of Implicit–Explicit High Order Runge–Kutta Methods to Discontinuous-Galerkin Schemes. *Journal of Computational Physics*, 225(2):1753–1781, 2007.

[21] Rainer Kress. *Numerical Analysis, Volume 181 of Graduate Texts in Mathematics*. Springer, 1998.

[22] Lilia Krivodonova. An Efficient Local Time-Stepping Scheme for Solution of Nonlinear Conservation Laws. *Journal of Computational Physics*, 229(22):8537–8551, 2010.

[23] Randall J LeVeque. *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge university press, 2002.

[24] Wm H Reed and TR Hill. Triangular Mesh Methods for the Neutron Transport Equation. *Los Alamos Report LA-UR-73-479*, 1973.

[25] Arne Taube, Michael Dumbser, Claus-Dieter Munz, and Rudolf Schneider. A high-order discontinuous galerkin method with time-accurate local time stepping for the maxwell equations. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 22(1):77–103, 2009.

[26] Eleuterio F Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: a Practical Introduction*. Springer Science & Business Media, 2013.

[27] Paul Woodward and Phillip Colella. The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks. *Journal of computational physics*, 54(1):115–173, 1984.

APPENDICES

Appendix A

Derivation of Runge-Kutta 3 Local Time-Stepping

In this appendix, we will give a detailed explanation of the derivation for the Runge-Kutta 3 local time-stepping scheme given in Section 4.2.1. To derive this scheme, we will analyze the local error of the inner stages of the method using their Taylor series expansions. We will then use previously computed values to create approximations to the inner stages that are at least as accurate as the inner stages of the method. When possible, we will match the high order derivative terms of the Taylor expansions using finite difference methods of appropriate order. This will give us the desired accuracy but will change the leading order error coefficient of the method.

To begin, we assume that the functions f and g in (4.1) are sufficiently smooth. For the sake of readability, unless otherwise state, the functions f and g and their derivatives are evaluated at $t = t_n$, i.e., $g = g(x(t_n), y(t_n))$. We will also assume that all data up to $t = t_n$ is exact. When referencing the actual Runge-Kutta stages we will denote the stages by $X_{rk}^{(i)}, Y_{rk}^{(i)}, i = 1, 2$, whereas the stages used in the local time-stepping scheme will be denoted as $X^{(i)}, Y^{(i)}, i = 1, 2$.

The standard Runge-Kutta method for advancing x_n to x_{n+1} is

$$X_{rk}^{(1)} = x_n + \frac{2}{3}h_{n+1}f(x_n, y_n), \tag{A.1a}$$

$$Y_{rk}^{(1)} = y_n + \frac{2}{3}h_{n+1}g(x_n, y_n), \tag{A.1b}$$

$$X_{rk}^{(2)} = x_n + \frac{2}{3}h_{n+1}f(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{A.1c}$$

$$Y_{rk}^{(2)} = y_n + \frac{2}{3}h_{n+1}g(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{A.1d}$$

$$x_{n+1} = x_n + \left(\frac{h_{n+1}}{4}\right)\left(f(x_n, y_n) + \frac{3}{2}f(X_{rk}^{(1)}, Y_{rk}^{(1)}) + \frac{3}{2}f(X_{rk}^{(2)}, Y_{rk}^{(2)})\right). \tag{A.1e}$$

As discussed in Section 4.1, we do not want to compute stages higher than stage 1, but develop approximations to these stages. First, we directly compute the first stage of y because it uses only information at the current time level t_n

$$Y_{rk}^{(1)} = Y^{(1)} = y_n + \frac{2}{3}h_{n+1}g(x_n, y_n). \tag{A.2}$$

Next, we look at the Taylor expansion of the second stage

$$\begin{aligned} Y_{rk}^{(2)} &= y_n + \frac{2}{3}h_{n+1}g(X^{(1)}, Y^{(1)}) \\ &= y_n + \frac{2}{3}h_{n+1}g\left(x_n + \frac{2}{3}h_{n+1}f(x_n, y_n), y_n + \frac{2}{3}h_{n+1}g(x_n, y_n)\right) \\ &= y_n + \frac{2}{3}h_{n+1}g + \frac{4}{9}h_{n+1}^2(g_x f + g_y g) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{A.3}$$

We observe that the h_{n+1}^2 component of (A.3) is the first derivative of $g(x(t), y(t))$ at time t_n. We approximate this value using a first order backwards difference

$$g_x f + g_y g = \frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} + \mathcal{O}(h_n). \tag{A.4}$$

Assuming that $h_n = ch_{n+1}$ for some constant c, we can substitute (A.4) into (A.3) to create the approximate stage

$$Y^{(2)} = y_n + \frac{2}{3}h_{n+1}g(x_n, y_n) + \frac{4}{9}h_{n+1}^2\left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n}\right) \tag{A.5}$$

$$= y_n + \frac{2}{3}h_{n+1}g + \frac{4}{9}h_{n+1}^2(g_x f + g_y g) + \mathcal{O}(h_{n+1}^3), \tag{A.6}$$

where $h_n h_{n+1}^2$ was absorbed into the $\mathcal{O}(h_{n+1}^3)$ term.

Using these approximations for $Y^{(1)}$ and $Y^{(2)}$, we take the Taylor expansions of (4.2e) and compare it with the exact solution of x. The Taylor expansion of the exact solution at $t = t_n + h_{n+1}$ is

$$\begin{aligned} x(t_n + h_{n+1}) = x(t_n) &+ h_{n+1} f + \frac{1}{2} h_{n+1}^2 \left(f_x f + f_y g \right) \\ &+ \frac{1}{6} h_{n+1}^3 \left(f_{xx} f^2 + 2 f_{xy} f g + f_x^2 f + f_x f_y g + f_{yy} g^2 + f_y g_x f + f_y g_y g \right) \\ &+ \mathcal{O}(h_{n+1}^4). \end{aligned} \tag{A.7}$$

Expanding (4.2e) yields

$$\begin{aligned} x_{n+1} &= x_n + \frac{h_{n+1}}{4} \left(f(x_n, y_n) + \frac{3}{2} f\left(X^{(1)}, Y^{(1)} \right) + \frac{3}{2} f\left(X^{(2)}, Y^{(2)} \right) \right) \\ &= x_n + h_{n+1} f + \frac{1}{2} h_{n+1}^2 \left(f_x f + f_y g \right) \\ &\quad + \frac{1}{6} h_{n+1}^3 \left(f_{xx} f^2 + 2 f_{xy} f g + f_x^2 f + f_x f_y g + f_{yy} g^2 + f_y g_x f + f_y g_y g \right) + \mathcal{O}(h_{n+1}^4). \end{aligned} \tag{A.8}$$

Comparing (A.7) with (A.8), we have $|x(t_{n+1}) - x_{n+1}| = \mathcal{O}(h_{n+1}^4)$.

Next, we are tasked with advancing y from t_n to t_{n+1} using K locally stable time steps of size h_{n+1}/K. To do so, we need to approximate the numerical solution x and its derivatives on the interval $[t_n, t_{n+1}]$. Using computed and stored information we will create an interpolating polynomial, $\mathcal{X}(t)$, satisfying $|x(t) - \mathcal{X}(t)| = \mathcal{O}(h_{n+1}^4)$ for $t \in [t_n, t_{n+1}]$. Our polynomial will have the form

$$\mathcal{X}(t) = a_0 + (t - t_n) a_1 + (t - t_n)^2 a_2 + (t + t_n)^3 a_3. \tag{A.9}$$

We use a cubic polynomial because we will need to match coefficients of the powers of δh_{n+1} up to and including $(\delta h_{n+1})^3$ of the Taylor expansion of the interpolating polynomial and the exact solution for $t = t_n + \delta h_{n+1}$, $0 \le \delta \le 1$. Using a cubic polynomial will give us the desired accuracy with the least number of interpolating points.

We require the polynomial to satisfy $\mathcal{X}(t_n) = x_n, \mathcal{X}(t_{n+1}) = x_{n+1}$ and $\mathcal{X}'(t_n) = f(x_n, y_n)$. To have a cubic interpolating polynomial we require a fourth point. Looking at the expansions of the inner stages of x we see that

$$x'\left(t_n + \frac{2}{3} h_{n+1} \right) = \frac{1}{2} f\left(X^{(1)}, Y^{(1)} \right) + \frac{1}{2} f\left(X^{(2)}, Y^{(2)} \right) + \mathcal{O}(h_{n+1}^3). \tag{A.10}$$

Choosing the fourth point to satisfy $\mathcal{X}'(t_n+2h_{n+1}/3) = \left(f\left(X^{(1)},Y^{(1)}\right)+f\left(X^{(2)},Y^{(2)}\right)\right)/2$, we get the family of interpolating polynomials

$$\begin{aligned}\mathcal{X}(t) = x_n + (t-t_n)f(x_n,y_n) + (t-t_n)^2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2} - h_{n+1}\beta\right)\\ + (t-t_n)^3\beta, \quad t_n \le t \le t_{n+1},\end{aligned} \tag{A.11}$$

where β is a parameter. Evaluating $\mathcal{X}(t)$ at $t_n+\delta h_{n+1}, 0 \le \delta \le 1$ and expanding the coefficients in a Taylor series around t_n, we get

$$\begin{aligned}\mathcal{X}(t_n+\delta h_{n+1}) &= x_n + (\delta h_{n+1})f(x_n,y_n) + (\delta h_{n+1})^2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2} - h_{n+1}\beta\right)\\ &\quad + (\delta h_{n+1})^3\beta\\ &= x_n + (\delta h_{n+1})f(x_n,y_n) + \frac{(\delta h_{n+1})^2}{2}\left(f_xf+f_yg\right)\\ &\quad + \frac{h_{n+1}^3}{6}\Big[\delta^2\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg-6\beta\right)\\ &\quad + \delta^3 6\beta\Big] + \mathcal{O}(h_{n+1}^4).\end{aligned} \tag{A.12}$$

Observe that if $\beta = x'''(t_n)/6 + \mathcal{O}(h_{n+1})$, we will achieve the desired accuracy. Using finite differences to approximate this derivative, we have

$$\begin{aligned}\beta &= \frac{1}{2h_{n+1}+3h_n}\left(2\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2} - \frac{f(x_n,y_n)-f(x_{n-1},y_{n-1})}{h_n}\right)\\ &= \frac{1}{6}\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)+\mathcal{O}(h_{n+1}).\end{aligned} \tag{A.13}$$

Then,

$$\begin{aligned}\mathcal{X}\left(t_n+\delta h_{n+1}\right) &= x_n + (\delta h_{n+1})f + \frac{(\delta h_{n+1})^2}{2}\left(f_xf+f_yg\right)\\ &\quad + \frac{(\delta h_{n+1})^3}{6}\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\ &\quad + \mathcal{O}(h_{n+1}^4),\end{aligned} \tag{A.14}$$

and $|x(t)-\mathcal{X}(t)| = \mathcal{O}(h_{n+1}^4)$ on $t \in [t_n, t_{n+1}]$, as desired.

Checking the accuracy of the derivatives of the interpolating polynomial we have

$$\begin{aligned}
\mathcal{X}'(t_n+\delta h_{n+1}) &= f(x_n,y_n)+2(\delta h_{n+1})\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-h_{n+1}\beta\right)\\
&\quad+3(\delta h_{n+1})^2\beta\\
&= f(x_n,y_n)+(\delta h_{n+1})\left(f_xf+f_yg\right)\\
&\quad+\frac{(\delta h_{n+1})^2}{2}\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\
&\quad+\mathcal{O}(h_{n+1}^3), \qquad \text{(A.15)}\\
\mathcal{X}''(t_n+\delta h_{n+1}) &= 2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-h_{n+1}\beta\right)+6(\delta h_{n+1})\beta\\
&= \left(f_xf+f_yg\right)\\
&\quad+(\delta h_{n+1})\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\
&\quad+\mathcal{O}(h_{n+1}^2). \qquad \text{(A.16)}
\end{aligned}$$

We see that $|x'(t)-\mathcal{X}'(t)|=\mathcal{O}(h_{n+1}^3)$ and $|x''(t)-\mathcal{X}''(t)|=\mathcal{O}(h_{n+1}^3)$ on $t\in[t_n,t_{n+1}]$.

Denote by $x_{n,k}$ and $y_{n,k}$ the numerical solutions of $x(t)$ and $y(t)$ at the sub time level $t_{n,k}=t_n+(k/K)h_{n+1}, k=0,1,\ldots,K$. Additionally let $X_k^{(i)}$ and $Y_k^{(i)}$, $i=1,2$ denote the inner Runge-Kutta stages at the fractional step k. We can think of advancing $y_{n,k}$ to $y_{n,k+1}$ as its own smaller problem to develop a scheme that works for each k for $k=0,1,\ldots,K-1$.

Concerned only with advancing $y_{n,k}$ to $y_{n,k+1}$, we have the standard Runge-Kutta scheme

$$X_{rk}^{(1)} = x_{n,k}+\frac{2}{3}\left(\frac{h_{n+1}}{K}\right)f(x_{n,k},y_{n,k}), \qquad \text{(A.17a)}$$

$$Y_{rk}^{(1)} = y_{n,k}+\frac{2}{3}\left(\frac{h_{n+1}}{K}\right)g(x_{n,k},y_{n,k}), \qquad \text{(A.17b)}$$

$$X_{rk}^{(2)} = x_{n,k}+\frac{2}{3}\left(\frac{h_{n+1}}{K}\right)f(X_{rk}^{(1)},Y_{rk}^{(1)}), \qquad \text{(A.17c)}$$

$$Y_{rk}^{(2)} = y_{n,k}+\frac{2}{3}\left(\frac{h_{n+1}}{K}\right)g(X_{rk}^{(1)},Y_{rk}^{(1)}), \qquad \text{(A.17d)}$$

$$y_{n,k+1} = y_{n,k}+\frac{1}{4}\left(\frac{h_{n+1}}{K}\right)\left(g(x_{n,k},y_{n,k})+\frac{3}{2}g(X_{rk}^{(1)},Y_{rk}^{(1)})+\frac{3}{2}g(X_{rk}^{(2)},Y_{rk}^{(2)})\right). \qquad \text{(A.17e)}$$

Now, since we have already computed x_{n+1}, we do not wish to compute the values of $x_{n,k}$ for each k. Additionally, we do not wish to compute any additional function evaluations

of $f(x,y)$. Using the interpolating polynomial, we have $x(t_{n,k}) = \mathcal{X}(t_{n,k}) + \mathcal{O}(h_{n+1}^4)$. Thus, in (A.17) we set $x_{n,k} = \mathcal{X}(t_{n,k})$. Furthermore, we know $f(x(t), y(t)) = x'(t)$ and $x'(t_{n,k}) = \mathcal{X}'(t_{n,k}) + \mathcal{O}(h_{n+1}^3)$, so in (A.17a) we set $f(x_{n,k}, y_{n,k}) = \mathcal{X}'(t_{n,k})$. This gives the approximation to the first stage

$$X_k^{(1)} = \mathcal{X}(t_{n,k}) + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}). \tag{A.18}$$

Next, we would like to approximate (A.17c). Expanding in a Taylor series about $t_{n,k}$ we have

$$\begin{aligned} X_{rk}^{(2)} &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f\left(X_{rk}^{(1)}, Y_{rk}^{(1)}\right) \\ &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f\left(x_{n,k} + \frac{2}{3}\left(\frac{h}{K}\right) f(x_{n,k}, y_{n,k}), y_{n,k} + \frac{2}{3}\left(\frac{h}{K}\right) g(x_{n,k}, y_{n,k})\right) \\ &= x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) f(x_{n,k}, y_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2 (f_x f + f_y g)(x_{n,k}, y_{n,k}) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{A.19}$$

To accurately approximate (A.17c), we need an approximation to $f(x_{n,k}, y_{n,k})$ and $(f_x f + f_y g)(x_{n,k}, y_{n,k})$. Again, we use $f(x_{n,k}, y_{n,k}) = \mathcal{X}'(t_{n,k})$. Observing that $(f_x f + f_y g)(x(t_{n,k}), y(t_{n,k})) = x''(t_{n,k})$ and $x''(t_{n,k}) = \mathcal{X}''(t_{n,k}) + \mathcal{O}(h_{n+1}^2)$ we arrive at the approximation to the second stage, given by

$$X_k^{(2)} = \mathcal{X}(t_{n,k}) + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2 \mathcal{X}''(t_{n,k}). \tag{A.20}$$

Using these approximations, we have the new scheme

$$x_{n,k} = \mathcal{X}(t_{n,k}), \tag{A.21a}$$

$$X_k^{(1)} = x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}), \tag{A.21b}$$

$$Y_k^{(1)} = y_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) g(x_{n,k}, y_{n,k}), \tag{A.21c}$$

$$X_k^{(2)} = x_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{4}{9}\left(\frac{h_{n+1}}{K}\right)^2\mathcal{X}''(t_{n,k}), \tag{A.21d}$$

$$Y_k^{(2)} = y_{n,k} + \frac{2}{3}\left(\frac{h_{n+1}}{K}\right) g(X_k^{(1)}, Y_k^{(1)}), \tag{A.21e}$$

$$y_{n,k+1} = y_{n,k} + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)\left(g(x_{n,k}, y_{n,k}) + \frac{3}{2}g(X_k^{(1)}, Y_k^{(1)}) + \frac{3}{2}g(X_k^{(2)}, Y_k^{(2)})\right). \tag{A.21f}$$

Expanding (A.21f) in a Taylor series about $t_{n,k}$, we have

$$\begin{aligned} y_{n,k+1} = y_{n,k} &+ \left(\frac{h_{n+1}}{K}\right) g + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2 (g_x f + g_y g) \\ &+ \frac{1}{6}\left(\frac{h_{n+1}}{K}\right)^3 \left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + g_{yy}g^2 + g_y g_x f + g_y^2 g\right) + \mathcal{O}(h_{n+1}^4), \end{aligned} \tag{A.22}$$

where each function is evaluated at $(x_{n,k}, y_{n,k})$. Expanding the exact solution in a Taylor series about $t_{n,k}$ we have

$$\begin{aligned} y(t_{n,k+1}) = y(t_{n,k}) &+ \left(\frac{h_{n+1}}{K}\right) g + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2 (g_x f + g_y g) \\ &+ \frac{1}{6}\left(\frac{h_{n+1}}{K}\right)^3 \left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + g_{yy}g^2 + g_y g_x f + g_y^2 g\right) + \mathcal{O}(h_{n+1}^4), \end{aligned} \tag{A.23}$$

where each function is evaluated at $(x(t_{n,k}), y(t_{n,k}))$. Subtracting (A.22) from (A.23) and assuming $(x_{n,k}, y_{n,k}) = (x(t_{n,k}), y(t_{n,k}))$ we have $|y(t_{n,k+1}) - y_{n,k+1}| = \mathcal{O}(h_{n+1}^4)$ as desired.

Appendix B

Derivation of Runge-Kutta 4 Local Time-Stepping

In this appendix, we will give a detailed explanation of the derivation for the Runge-Kutta 4 local time-stepping scheme given in Section 4.2.2. To derive this scheme, we will analyze the local error of the inner stages of the method using their Taylor series expansions. We will then use previously computed values to create approximations to the inner stages that are at least as accurate as the inner stages of the method. When possible, we will match the high order derivative terms of the Taylor expansions using finite difference methods of appropriate order. This will give us the desired accuracy but will change the leading order error coefficient of the method.

To begin, we assume that the functions f and g in (4.1) are sufficiently smooth. For the sake of readability, unless otherwise state, the functions f and g and their derivatives are evaluated at $t = t_n$, i.e., $g = g(x(t_n), y(t_n))$. We will also assume that all data up to $t = t_n$ is exact. When referencing the actual Runge-Kutta stages we will denote the stages by $X_{rk}^{(i)}, Y_{rk}^{(i)}, i = 1, 2, 3$, whereas the stages used in the local time-stepping scheme will be denoted as $X^{(i)}, Y^{(i)}, i = 1, 2, 3$.

The standard Runge-Kutta method for advancing x_n to x_{n+1} is

$$X_{rk}^{(1)} = x_n + \frac{1}{2}h_{n+1}f(x_n, y_n), \tag{B.1a}$$

$$Y_{rk}^{(1)} = y_n + \frac{1}{2}h_{n+1}g(x_n, y_n), \tag{B.1b}$$

$$X_{rk}^{(2)} = x_n + \frac{1}{2}h_{n+1}f(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{B.1c}$$

$$Y_{rk}^{(2)} = y_n + \frac{2}{2}h_{n+1}g(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{B.1d}$$

$$X_{rk}^{(3)} = x_n + h_{n+1}f(X_{rk}^{(2)}, Y_{rk}^{(2)}), \tag{B.1e}$$

$$Y_{rk}^{(3)} = y_n + h_{n+1}g(X_{rk}^{(2)}, Y_{rk}^{(2)}), \tag{B.1f}$$

$$x_{n+1} = x_n + \left(\frac{h_{n+1}}{6}\right)\left(f(x_n, y_n) + 2f(X_{rk}^{(1)}, Y_{rk}^{(1)}) + 2f(X_{rk}^{(2)}, Y_{rk}^{(2)}) + f(X_{rk}^{(3)}, Y_{rk}^{(3)})\right). \tag{B.1g}$$

As discussed in Section 4.1, we do not want to compute stages higher than stage 1, but develop approximations to these stages. First, we directly compute the first stage of y because it uses only information at the current time level t_n

$$Y_{rk}^{(1)} = Y^{(1)} = y_n + \frac{1}{2}h_{n+1}g(x_n, y_n). \tag{B.2}$$

Next, we look at the Taylor expansion of the second stage of the Runge-Kutta scheme for the function y

$$\begin{aligned} Y_{rk}^{(2)} &= y_n + \frac{1}{2}h_{n+1}g(X_{rk}^{(1)}, Y_{rk}^{(1)}) \\ &= y_n + \frac{1}{2}h_{n+1}g\left(x_n + \frac{1}{2}h_{n+1}f(x_n, y_n), y_n + \frac{1}{2}h_{n+1}g(x_n, y_n)\right) \\ &= y_n + \frac{1}{2}h_{n+1}g + \frac{1}{4}h_{n+1}^2(g_x f + g_y g) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{B.3}$$

We observe that the h_{n+1}^2 component of (B.3) is the first derivative of $g(x(t), y(t))$ at time t_n. We approximate this value using a first order backwards difference

$$g_x f + g_y g = \frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} + \mathcal{O}(h_n). \tag{B.4}$$

Assuming that $h_n = ch_{n+1}$ for some constant c, we can substitute (B.4) into (B.3) to create the approximate stage

$$Y^{(2)} = y_n + \frac{1}{2}h_{n+1}g(x_n, y_n) + \frac{1}{4}h_{n+1}^2\left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n}\right). \tag{B.5}$$

Expanding this second stage in a Taylor series we have

$$\begin{aligned} Y^{(2)} = y_n &+ \frac{1}{2}h_{n+1}g(x_n, y_n) + \frac{1}{4}h_{n+1}^2(g_x f + g_y g) \\ &- \frac{1}{8}h_{n+1}^2 h_n\left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + f_{yy}g^2 + g_y g_x f + g_y^2 g\right) + \mathcal{O}(h_{n+1}^4). \end{aligned} \tag{B.6}$$

Since the Runge-Kutta 4 methods is locally fifth order, we require that the the h^3 terms of the stages to be in terms of only h_{n+1}. We define a correction term that will replace

$$\frac{1}{8}h_{n+1}^2 h_n\left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + f_{yy}g^2 + g_y g_x f + g_y^2 g\right) \tag{B.7}$$

with

$$\frac{1}{8}h_{n+1}^3\left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + f_{yy}g^2 + g_y g_x f + g_y^2 g\right). \tag{B.8}$$

The correction term is given by

$$Corr_y = \left(2\frac{h_{n+1} - h_n}{h_n + h_{n-1}}\right)\left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{g(x_{n-1}, y_{n-1}) - g(x_{n-2}, y_{n-2})}{h_{n-1}}\right) \tag{B.9}$$

and has Taylor expansion

$$Corr_y = (h_{n+1} - h_n)\left(g_{xx}f^2 + 2g_{xy}fg + g_x f_x f + g_x f_y g + f_{yy}g^2 + g_y g_x f + g_y^2 g\right) + \mathcal{O}(h_{n+1}). \tag{B.10}$$

This gives us the second stage

$$Y^{(2)} = y_n + \frac{1}{2}h_{n+1}g(x_n, y_n) + \frac{1}{4}h_{n+1}^2\left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{1}{2}Corr_y\right). \tag{B.11}$$

Next, we must look at the Taylor expansion of the third Runge-Kutta stage of y

$$\begin{aligned} Y_{rk}^{(3)} &= y_n + h_{n+1}g(X_{rk}^{(2)}, Y_{rk}^{(2)}) \\ &= y_n + h_{n+1}g\left(x_n + \frac{1}{2}h_{n+1}f(X_{rk}^{(1)}, Y_{rk}^{(1)}), y_n + \frac{1}{2}h_{n+1}g(X_{rk}^{(1)}, Y_{rk}^{(1)})\right) \\ &= y_n + h_{n+1}g + \frac{1}{2}h_{n+1}^2(g_x f + g_y g) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{B.12}$$

Again, we use (B.4) to approximate $g_x f + g_y g$ with the added correction term. If we were to simply use

$$Y^{(3)} = y_n + h_{n+1} g(x_n, y_n) + \frac{1}{2} h_{n+1}^2 \left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{1}{2} Corr_y \right)$$

as our approximate third stage, we would not have accuracy in the h^4 term of x_{n+1}. We need added information in terms of the third derivative of g in the third ghost stage. We approximate this derivative using finite differences. To find the corresponding coefficient of this term we apply the approximation with a parameter A and compute x_{n+1}. With the parameter A, the third stage is

$$\begin{aligned} Y^{(3)} = y_n + h_{n+1} g(x_n, y_n) + \frac{1}{2} h_{n+1}^2 \left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{1}{2} Corr_y \right) \\ + A h_{n+1}^3 \left(\frac{2}{h_{n+1} + h_n} \right) \left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{g(x_{n-1}, y_{n-1}) - g(x_{n-2}, y_{n-2})}{h_{n-1}} \right). \end{aligned} \tag{B.13}$$

The value of x_{n+1} is computed using

$$x_{n+1} = x_n + \frac{h_{n+1}}{6} \left(f(x_n, y_n) + 2f\left(X^{(1)}, Y^{(1)}\right) + 2f\left(X^{(2)}, Y^{(2)}\right) + f\left(X^{(3)}, Y^{(3)}\right) \right) \tag{B.14}$$

We then expand x_{n+1} in a Taylor series and subtract it from the exact solution. The Taylor expansion of the exact solution $x(t_{n+1})$ about t_n is

$$\begin{aligned} x(t_{n+1}) = x(t_n) + h_{n+1} f + \frac{1}{2} h_{n+1}^2 \left(f_x f + f_y g \right) \\ + \frac{1}{6} h_{n+1}^3 \left(f_{xx} f^2 + 2 f_{xy} f g + f_x^2 f + f_x f_y g + f_{yy} g^2 + f_y g_x f + f_y g_y g \right) \\ + \frac{1}{24} h_{n+1}^4 \Bigg(3 f_{xx} f f_y g + 5 f_{xy} g f_x f + 3 f_{xy} f g_y g + 2 f_x f_y g_x f + f_x f_y g_y g + 3 f_{yy} g g_x f \\ + 2 f_y g_{xy} f g + f_y g_y g_x f + 3 f_{xxy} f^2 g + 3 f_{xyy} f g^2 + f_x^2 f_y g + f_y g_{xx} f^2 + f_y g_y^2 g \\ + g_x f_y^2 g + 4 f_{xx} f^2 f_x + f_{xxx} f^3 + f_{yyy} g^3 + f_x^3 f + f_y g_{yy} g^2 + f_x f_{yy} g^2 + 3 f_{xy} g^2 f_y \\ + 3 f_{xy} f^2 g_x + 3 f_{yy} g^2 g_y \Bigg) + \mathcal{O}(h_{n+1}^5). \end{aligned} \tag{B.15}$$

Subtracting the numerical solution from the exact solution gives

$$\begin{aligned} x(t_{n+1}) - x_{n+1} = h_{n+1}^4 \left(\frac{1}{8} - \frac{A}{6} \right) \Big(g_{yy} f_y g^2 + g_{xx} f_y f^2 \\ + f_y g_y^2 g g_x f_y^2 g + f_x f_y g_x f + 2 f_y g_{xy} f g + f_y g_y g_x f \Big) + \mathcal{O}(h_{n+1}^5). \end{aligned} \tag{B.16}$$

Solving for A so that $|x(t_{n+1}) - x_{n+1}| = \mathcal{O}(h_{n+1}^5)$ gives a value of $A = 3/4$ and so the third stage in y is

$$Y^{(3)} = y_n + h_{n+1} g(x_n, y_n) + \frac{1}{2} h_{n+1}^2 \left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{1}{2} Corr_y \right) + \frac{3}{4} h_{n+1}^3 \left(\frac{2}{h_{n+1} + h_n} \right) \left(\frac{g(x_n, y_n) - g(x_{n-1}, y_{n-1})}{h_n} - \frac{g(x_{n-1}, y_{n-1}) - g(x_{n-2}, y_{n-2})}{h_{n-1}} \right).$$

Next, we are tasked with advancing y from t_n to t_{n+1} using K locally stable time steps of size h_{n+1}/K. To do so, we need to approximate the numerical solution x and its derivatives on the interval $[t_n, t_{n+1}]$. Using computed and stored information we will create an interpolating polynomial $\mathcal{X}(t)$ satisfying $|x(t) - \mathcal{X}(t)| = \mathcal{O}(h_{n+1}^5)$ for $t \in [t_n, t_{n+1}]$. Our polynomial will have the form

$$\mathcal{X}(t) = a_0 + (t - t_n) a_1 + (t - t_n)^2 a_2 + (t + t_n)^3 a_3 + (t - t_n)^4 a_4. \tag{B.17}$$

We use a quartic polynomial because we will need to match coefficients of the powers of δh_{n+1} up to and including $(\delta h_{n+1})^4$ of the Taylor expansion of the interpolating polynomial and the exact solution for $t = t_n + \delta h_{n+1}$, $0 \leq \delta \leq 1$. Using a quartic polynomial will give us the desired accuracy with the least number of interpolating points.

Similarly to the Runge-Kutta 3 local time stepping method, we require the polynomial to satisfy $\mathcal{X}(t_n) = x_n, \mathcal{X}(t_{n+1}) = x_{n+1}$ and $\mathcal{X}'(t_n) = f(x_n, y_n)$. To have a quartic interpolating polynomial, we require five interpolating points. Using finite difference approximations we are able to compute

$$\beta = \frac{6(2h_{n+1} + 3h_n)}{6h_n^2 + 8h_{n+1}h_n + 6h_n h_{n-1} + 3h_{n+1}^2 + 4h_{n+1}h_{n-1}} \Bigg[\left(\frac{6}{2h_{n+1} + 3h_n} \right) \left(2 \frac{x_{n+1} - x_n - h_{n+1} f(x_n, y_n)}{h_{n+1}^2} - \frac{f(x_n, y_n) - f(x_{n-1}, y_{n-1})}{h_n} \right) - \left(\frac{2}{h_n + h_{n-1}} \right) \left(\frac{f(x_n, y_n) - f(x_{n-1}, y_{n-1})}{h_n} - \frac{f(x_{n-1}, y_{n-1}) - f(x_{n-2}, y_{n-2})}{h_{n-1}} \right) \Bigg], \tag{B.18}$$

$$\alpha = \left(\frac{6}{2h_{n+1} + 3h_n} \right) \left(2 \frac{x_{n+1} - x_n - h_{n+1} f(x_n, y_n)}{h_{n+1}^2} - \frac{f(x_n, y_n) - f(x_{n-1}, y_{n-1})}{h_n} \right) + \left(\frac{3h_{n+1}^2 + 6h_{n+1}h_n + 2h_n^2}{2(2h_{n+1} + 3h_n)} \right) \beta, \tag{B.19}$$

where $\alpha = x^{(3)}(t_{n+1}) + \mathcal{O}(h_{n+1}^2)$ and $\beta = x^{(4)}(t_n) + \mathcal{O}(h_{n+1})$. Choosing $\mathcal{X}^{(3)}(t_{n+1}) = \alpha$ and $\mathcal{X}^{(4)}(t_n) = \beta$, we get the unique interpolating polynomial

$$\begin{aligned}\mathcal{X}(t) = x_n &+ (t-t_n)f(x_n,y_n) \\ &+ (t-t_n)^2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2} - \frac{h_{n+1}}{6}\alpha + \frac{h_{n+1}^2}{8}\beta\right) \\ &+ \frac{(t-t_n)^3}{6}(\alpha - h_{n+1}\beta) + \frac{(t-t_n)^4}{24}\beta, \quad t_n \le t \le t_{n+1}. \end{aligned} \tag{B.20}$$

Expanding each coefficient in a Taylor series about t_n and evaluating the polynomial at $t = t_n + \delta h_{n+1}$ for $0 \le \delta \le 1$ gives

$$\begin{aligned}\mathcal{X}(t_n+\delta h_{n+1}) = x_n &+ (\delta h_{n+1})f(x_n,y_n) + \frac{(\delta h_{n+1})^2}{2}(f_x f + f_y g) \\ &+ \frac{(\delta h_{n+1})^3}{6}\left(f_{xx}f^2 + 2f_{xy}fg + f_x^2 f + f_x f_y g + f_{yy}g^2 + f_y g_x f + f_y g_y g\right) \\ &+ \frac{(\delta h_{n+1})^4}{24}\Bigg(3f_{xx}ff_yg + 5f_{xy}gf_xf + 3f_{xy}fg_yg + 2f_xf_yg_xf + f_xf_yg_yg \\ &+ 3f_{yy}gg_xf + 2f_yg_{xy}fg + f_yg_yg_xf + 3f_{xxy}f^2g + 3f_{xyy}fg^2 + f_x^2f_yg \\ &+ f_yg_{xx}f^2 + f_yg_y^2g + g_xf_y^2g + 4f_{xx}f^2f_x + f_{xxx}f^3 + f_{yyy}g^3 + f_x^3f \\ &+ f_yg_{yy}g^2 + f_xf_{yy}g^2 + 3f_{xy}g^2f_y + 3f_{xy}f^2g_x + 3f_{yy}g^2g_y\Bigg) + \mathcal{O}(h_{n+1}^5). \end{aligned} \tag{B.21}$$

Expanding $x(t_n + \delta h_{n+1})$ and subtracting the interpolating polynomial, we have $|x(t) - \mathcal{X}(t)| = \mathcal{O}(h_{n+1}^5)$ on $t \in [t_n, t_{n+1}]$.

Checking the accuracy of the derivatives of interpolating polynomial for $0 \le \delta \le 1$, we

have

$$\begin{aligned}
\mathcal{X}'(t_n+\delta h_{n+1}) &= f(x_n,y_n)\\
&\quad + 2(\delta h_{n+1})\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-\frac{h_{n+1}}{6}\alpha+\frac{h_{n+1}^2}{8}\beta\right)\\
&\quad + \frac{(\delta h_{n+1})^2}{2}\left(\alpha-h_{n+1}\beta\right)+\frac{(\delta h_{n+1})^3}{6}\beta\\
&= f(x_n,y_n)+(\delta h_{n+1})\left(f_xf+f_yg\right)\\
&\quad + \frac{(\delta h_{n+1})^2}{2}\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\
&\quad + \frac{(\delta h_{n+1})^3}{6}\Bigg(3f_{xx}ff_yg+5f_{xy}gf_xf+3f_{xy}fg_yg+2f_xf_yg_xf+f_xf_yg_yg\\
&\quad + 3f_{yy}gg_xf+2f_yg_{xy}fg+f_yg_yg_xf+3f_{xxy}f^2g+3f_{xyy}fg^2+f_x^2f_yg\\
&\quad + f_yg_{xx}f^2+f_yg_y^2g+g_xf_y^2g+4f_{xx}f^2f_x+f_{xxx}f^3+f_{yyy}g^3+f_x^3f\\
&\quad + f_yg_{yy}g^2+f_xf_{yy}g^2+3f_{xy}g^2f_y+3f_{xy}f^2g_x+3f_{yy}g^2g_y\Bigg)+\mathcal{O}(h_{n+1}^4),
\end{aligned} \tag{B.22}$$

and $|x'(t)-\mathcal{X}'(t)|=\mathcal{O}(h_{n+1}^4)$ on $t\in[t_n,t_{n+1}]$.

Expanding the second derivative of interpolating polynomial gives

$$\begin{aligned}
\mathcal{X}''(t_n+\delta h_{n+1}) &= 2\left(\frac{x_{n+1}-x_n-h_{n+1}f(x_n,y_n)}{h_{n+1}^2}-\frac{h_{n+1}}{6}\alpha+\frac{h_{n+1}^2}{8}\beta\right)\\
&\quad+(\delta h_{n+1})\left(\alpha-h_{n+1}\beta\right)+\frac{(\delta h_{n+1})^2}{2}\beta\\
&=(f_xf+f_yg)\\
&\quad+(\delta h_{n+1})\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\
&\quad+\frac{(\delta h_{n+1})^2}{2}\Bigg(3f_{xx}ff_yg+5f_{xy}gf_xf+3f_{xy}fg_yg+2f_xf_yg_xf\\
&\quad+f_xf_yg_yg+3f_{yy}gg_xf+2f_yg_{xy}fg+f_yg_yg_xf+3f_{xxy}f^2g+3f_{xyy}fg^2\\
&\quad+f_x^2f_yg+f_yg_{xx}f^2+f_yg_y^2g+g_xf_y^2g+4f_{xx}f^2f_x+f_{xxx}f^3+f_{yyy}g^3\\
&\quad+f_x^3f+f_yg_{yy}g^2+f_xf_{yy}g^2+3f_{xy}g^2f_y+3f_{xy}f^2g_x+3f_{yy}g^2g_y\Bigg)+\mathcal{O}(h_{n+1}^3).
\end{aligned}\tag{B.23}$$

We see that $|x''(t)-\mathcal{X}''(t)|=\mathcal{O}(h_{n+1}^3)$ on $t\in[t_n,t_{n+1}]$.

Finally, expanding the third derivative of the interpolating polynomial, we have

$$\begin{aligned}
\mathcal{X}'''(t_n+\delta h_{n+1}) &= (\alpha-h_{n+1}\beta)+(\delta h_{n+1})\beta\\
&=\left(f_{xx}f^2+2f_{xy}fg+f_x^2f+f_xf_yg+f_{yy}g^2+f_yg_xf+f_yg_yg\right)\\
&\quad+(\delta h_{n+1})\Bigg(3f_{xx}ff_yg+5f_{xy}gf_xf+3f_{xy}fg_yg+2f_xf_yg_xf\\
&\quad+f_xf_yg_yg+3f_{yy}gg_xf+2f_yg_{xy}fg+f_yg_yg_xf+3f_{xxy}f^2g+3f_{xyy}fg^2\\
&\quad+f_x^2f_yg+f_yg_{xx}f^2+f_yg_y^2g+g_xf_y^2g+4f_{xx}f^2f_x+f_{xxx}f^3+f_{yyy}g^3\\
&\quad+f_x^3f+f_yg_{yy}g^2+f_xf_{yy}g^2+3f_{xy}g^2f_y+3f_{xy}f^2g_x+3f_{yy}g^2g_y\Bigg)+\mathcal{O}(h_{n+1}^2).
\end{aligned}\tag{B.24}$$

This reveals that $|x'''(t)-\mathcal{X}'''(t)|=\mathcal{O}(h_{n+1}^2)$ on $t\in[t_n,t_{n+1}]$.

Denote by $x_{n,k}$ and $y_{n,k}$ the numerical solutions of $x(t)$ and $y(t)$ at the sub time level $t_{n,k}=t_n+(k/K)h_{n+1}, k=0,1,\ldots,K$. Additionally let $X_k^{(i)}$ and $Y_k^{(i)}$, $i=1,2$ denote the

inner Runge-Kutta stages at the fractional step k. We can think of advancing $y_{n,k}$ to $y_{n,k+1}$ as its own smaller problem to develop a scheme that works for each k for $k = 0, 1, \ldots, K-1$.

Concerned only with advancing $y_{n,k}$ to $y_{n,k+1}$, we have the standard Runge-Kutta scheme

$$X_{rk}^{(1)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f(x_{n,k}, y_{n,k}), \tag{B.25a}$$

$$Y_{rk}^{(1)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(x_{n,k}, y_{n,k}), \tag{B.25b}$$

$$X_{rk}^{(2)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{B.25c}$$

$$Y_{rk}^{(2)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(X_{rk}^{(1)}, Y_{rk}^{(1)}), \tag{B.25d}$$

$$X_{rk}^{(3)} = x_{n,k} + \left(\frac{h_{n+1}}{K}\right) f(X_{rk}^{(2)}, Y_{rk}^{(2)}), \tag{B.25e}$$

$$Y_{rk}^{(3)} = y_{n,k} + \left(\frac{h_{n+1}}{K}\right) g(X_{rk}^{(2)}, Y_{rk}^{(2)}), \tag{B.25f}$$

$$y_{n,k+1} = y_{n,k} + \frac{1}{6}\left(\frac{h_{n+1}}{K}\right)\left(g(x_{n,k}, y_{n,k}) + 2g(X_{rk}^{(1)}, Y_{rk}^{(1)}) + 2g(X_{rk}^{(2)}, Y_{rk}^{(2)}) + g(X_{rk}^{(3)}, Y_{rk}^{(3)})\right). \tag{B.25g}$$

Now, since we have already computed x_{n+1}, we do not wish to compute the values of $x_{n,k}$ for each k. Additionally, we do not wish to compute any additional function evaluations of $f(x, y)$. Using the interpolating polynomial, we have $x(t_{n,k}) = \mathcal{X}(t_{n,k}) + \mathcal{O}(h_{n+1}^4)$. Thus, in (B.25) we set $x_{n,k} = \mathcal{X}(t_{n,k})$. Furthermore, we know $f(x(t), y(t)) = x'(t)$ and $x'(t_{n,k}) = \mathcal{X}'(t_{n,k}) + \mathcal{O}(h_{n+1}^3)$, so in (B.25a) we set $f(x_{n,k}, y_{n,k}) = \mathcal{X}'(t_{n,k})$. This gives the approximation to the first stage

$$X_k^{(1)} = \mathcal{X}(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}). \tag{B.26}$$

Next, we would like to approximate (B.25c). Expanding in a Taylor series about $t_{n,k}$

we have

$$\begin{aligned} X_{rk}^{(2)} &= x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f\left(X_{rk}^{(1)}, Y_{rk}^{(1)}\right) \\ &= x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f\left(x_{n,k} + \frac{1}{2}\left(\frac{h}{K}\right) f(x_{n,k}, y_{n,k}), y_{n,k} + \frac{1}{2}\left(\frac{h}{K}\right) g(x_{n,k}, y_{n,k})\right) \\ &= x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f(x_{n,k}, y_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^2 (f_x f + f_y g)(x_{n,k}, y_{n,k}) + \mathcal{O}(h_{n+1}^3). \end{aligned} \tag{B.27}$$

To accurately approximate (B.27), we need an approximation to $f(x_{n,k}, y_{n,k})$ and $(f_x f + f_y g)(x_{n,k}, y_{n,k})$. Again, we use $f(x_{n,k}, y_{n,k}) = \mathcal{X}'(t_{n,k})$. Observing that $(f_x f + f_y g)(x(t_{n,k}), y(t_{n,k})) = x''(t_{n,k})$ and $x''(t_{n,k}) = \mathcal{X}''(t_{n,k}) + \mathcal{O}(h_{n+1}^2)$ we arrive at the approximation to the second stage, given by

$$X_k^{(2)} = \mathcal{X}(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^2 \mathcal{X}''(t_{n,k}). \tag{B.28}$$

Finally, we need to approximate (B.25e). Expanding in a Taylor series about $t_{n,k}$ we have

$$\begin{aligned} X_{rk}^{(3)} &= x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) f\left(X_{rk}^{(2)}, Y_{rk}^{(2)}\right) \\ &= x_{n,k} + \left(\frac{h_{n+1}}{K}\right) f(x_{n,k}, y_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2 (f_x f + f_y g)(x_{n,k}, y_{n,k}) \\ &\quad + \frac{1}{8}\left(\frac{h_{n+1}}{K}\right)^3 \left(f_{xx} f^2 + 2 f_{xy} f g + f_x f_y g + f_x^2 f + f_{yy} g^2 + f_y g_x f + f_y g_y g\right) + \mathcal{O}(h_{n+1}^4). \end{aligned} \tag{B.29}$$

To accurately approximate (B.29), we need an approximation to $f(x_{n,k}, y_{n,k})$ and $(f_x f + f_y g)(x_{n,k}, y_{n,k})$. Again, we use $f(x_{n,k}, y_{n,k}) = \mathcal{X}'(t_{n,k})$. Observing that $(f_x f + f_y g)(x(t_{n,k}), y(t_{n,k})) = x''(t_{n,k})$, we have $x''(t_{n,k}) = \mathcal{X}''(t_{n,k}) + \mathcal{O}(h_{n+1}^2)$. Finally, we have that

$$(f_{xx} f^2 + 2 f_{xy} f g + f_x f_y g + f_x^2 f + f_{yy} g^2 + f_y g_x f + f_y g_y g)(x(t_{n,k}), y(t_{n,k})) = x'''(t_{n,k}),$$

so we use the approximation $x'''(t_{n,k}) = \mathcal{X}'''(t_{n,k})$. We note that in (B.26) and (B.28) we have introduced higher order information into the scheme, so we may not be able to

approximate $X_{rk}^{(3)}$ with the same derivative coefficients. To find the coefficient of the last derivative term, we set the third stage as

$$X_k^{(3)} = \mathcal{X}(t_{n,k}) + \left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2\mathcal{X}''(t_{n,k}) + A\left(\frac{h_{n+1}}{K}\right)^3\mathcal{X}'''(t_{n,k}) \quad \text{(B.30)}$$

where A is a parameter we need to solve for. We compute (B.25g) with this approximate stage. We then expand the new solution in a Taylor series about $t_{n,k}$ and subtract it from the exact solution. We then find the value of A which satisfies $|y(t_{n,k+1}) - y_{n,k+1}| = \mathcal{O}(h_{n+1}^5)$. We see that

$$\begin{aligned} y(t_{n,k+1}) - y_{n,k+1} = \left(\frac{1}{384} - \frac{A}{96}\right)\left(\frac{h_{n+1}}{K}\right)\Big(&g_{xx}f^2 + 2g_{xy}fg + g_x f_x f \\ &+ g_x f_y g + f_{yy}g^2 + g_y g_x f + g_y^2 g\Big) + \mathcal{O}(h_{n+1}^5). \end{aligned} \quad \text{(B.31)}$$

Using $A = 1/4$ allows us to eliminate the h_{n+1}^4 component of the error, giving $|y(t_{n,k+1}) - y_{n,k+1}| = \mathcal{O}(h_{n+1}^5)$ as desired. This gives following approximation to the third stage

$$X_k^{(3)} = \mathcal{X}(t_{n,k}) + \left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2\mathcal{X}''(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^3\mathcal{X}'''(t_{n,k}). \quad \text{(B.32)}$$

Using these approximations, we have the new scheme

$$x_{n,k} = \mathcal{X}(t_{n,k}), \tag{B.33a}$$

$$X_k^{(1)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}), \tag{B.33b}$$

$$Y_k^{(1)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(x_{n,k}, y_{n,k}), \tag{B.33c}$$

$$X_k^{(2)} = x_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^2 \mathcal{X}''(t_{n,k}), \tag{B.33d}$$

$$Y_k^{(2)} = y_{n,k} + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right) g(X_k^{(1)}, Y_k^{(1)}), \tag{B.33e}$$

$$X_k^{(3)} = \mathcal{X}(t_{n,k}) + \left(\frac{h_{n+1}}{K}\right)\mathcal{X}'(t_{n,k}) + \frac{1}{2}\left(\frac{h_{n+1}}{K}\right)^2 \mathcal{X}''(t_{n,k}) + \frac{1}{4}\left(\frac{h_{n+1}}{K}\right)^3 \mathcal{X}'''(t_{n,k}), \tag{B.33f}$$

$$Y_k^{(3)} = y_{n,k} + \left(\frac{h_{n+1}}{K}\right) g(X_k^{(2)}, Y_k^{(2)}), \tag{B.33g}$$

$$y_{n,k+1} = y_{n,k} + \frac{1}{6}\left(\frac{h_{n+1}}{K}\right)\left(g(x_{n,k}, y_{n,k}) + 2g(X_k^{(1)}, Y_k^{(1)}) + 2g(X_k^{(2)}, Y_k^{(2)}) + g(X_k^{(3)}, Y_k^{(3)})\right). \tag{B.33h}$$

Appendix C

Taylor Expansions

In this appendix we give the remainder terms for the Taylor expansions for the proofs in Sections 4.2.1 and 4.2.2.

The exact fourth derivative of $x(t)$, where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned} x^{(4)}(t) = {} & 3f_{xx}ff_yg + 5f_{xy}gf_xf + 3f_{xy}fg_yg + 2f_xf_yg_xf + f_xf_yg_yg + 3f_{yy}gg_xf \\ & + 2f_yg_{xy}fg + f_yg_yg_xf + 3f_{xxy}f^2g + 3f_{xyy}fg^2 + f_x^2f_yg + f_yg_{xx}f^2 \\ & + f_yg_y^2g + g_xf_y^2g + 4f_{xx}f^2f_x + f_{xxx}f^3 + f_{yyy}g^3 + f_x^3f + f_yg_{yy}g^2 \\ & + f_xf_{yy}g^2 + 3f_{xy}g^2f_y + 3f_{xy}f^2g_x + 3f_{yy}g^2g_y. \end{aligned} \tag{C.1}$$

The remainder term when computing the fourth order Taylor series expansion of (4.2e), where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned} \mathcal{R}_x(t) = {} & \frac{1}{6}f_{xx}f^2f_x + \frac{1}{9}f_{xxy}f^2g + \frac{1}{9}f_{xyy}fg^2 - \frac{1}{12}f_yg_y^2g + \frac{1}{18}f_xf_{yy}g^2 - \frac{1}{12}f_yg_{yy}g^2 \\ & - \frac{1}{12}g_xf_y^2g + \frac{1}{9}f_{yy}g^2g_y + \frac{1}{9}f_{xy}f^2g_x - \frac{1}{12}f_yg_{xx}f^2 + \frac{1}{9}f_{xy}g^2f_y + \frac{1}{9}f_{xx}ff_yg \\ & + \frac{2}{9}f_{xy}gf_xf + \frac{1}{9}f_{xy}fg_yg - \frac{1}{12}f_xf_yg_xf + \frac{1}{9}f_{yy}gg_xf - \frac{1}{6}f_yg_{xy}fg \\ & + \frac{1}{27}f_{yyy}g^3 + \frac{1}{27}f_{xxx}f^3 - \frac{1}{12}f_yg_yg_xf. \end{aligned} \tag{C.2}$$

The remainder term when computing the fourth order Taylor series expansion of (4.6),

where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}
\mathcal{R}_{\mathcal{X}}(t) = \Bigg(& -\frac{1}{18}f_{xyy}fg^2 - \frac{1}{15}f_y g_y^2 g - \frac{1}{90}f_x f_{yy} g^2 - \frac{1}{15}f_y g_{yy} g^2 - \frac{1}{15}g_x f_y^2 g \\
& - \frac{1}{18}f_{yy}g^2 g_y - \frac{1}{18}f_{xy}f^2 g_x - \frac{1}{15}f_y g_{xx} f^2 - \frac{1}{18}f_{xy}g^2 f_y - \frac{1}{30}f_x^2 f_y g \\
& - \frac{1}{18}f_{xx}f f_y g - \frac{7}{90}f_{xy}g f_x f - \frac{1}{18}f_{xy}f g_y g - \frac{1}{10}f_x f_y g_x f - \frac{1}{18}f_{yy}g g_x f \\
& - \frac{2}{15}f_y g_{xy} f g - \frac{1}{15}f_y g_y g_x f - \frac{1}{30}f_x f_y g_y g - \frac{1}{15}f_{xx}f^2 f_x - \frac{1}{18}f_{xxy}f^2 g \\
& - \frac{1}{54}f_{yyy}g^3 - \frac{1}{54}f_{xxx}f^3 - \frac{1}{30}f_x^3 f \Bigg)\delta^3 \\
& + \Bigg(\frac{1}{18}f_{yyy}g^3 + \frac{1}{18}f_{xxx}f^3 + \frac{1}{30}f_x^2 f_y g + \frac{1}{30}f_x^3 f + \frac{1}{6}f_{xy}f^2 g_x \\
& + \frac{1}{6}f_{xy}g^2 f_y + \frac{1}{6}f_{yy}g g_x f - \frac{1}{30}f_y g_{xy} f g - \frac{1}{60}f_y g_y g_x f + \frac{3}{10}f_{xy}g f_x f \\
& + \frac{1}{30}f_x f_y g_y g + \frac{1}{6}f_{xx}f f_y g + \frac{1}{6}f_{xy}f g_y g + \frac{1}{6}f_{yy}g^2 g_y + \frac{7}{30}f_{xx}f^2 f_x \\
& + \frac{1}{6}f_{xxy}f^2 g + \frac{1}{15}f_x f_{yy}g^2 + \frac{1}{60}f_x f_y g_x f - \frac{1}{60}g_x f_y^2 g - \frac{1}{60}f_y g_{xx} f^2 \\
& + \frac{1}{6}f_{xyy}f g^2 - \frac{1}{60}f_y g_y^2 g - \frac{1}{60}f_y g_{yy} g^2 \Bigg)\delta^2. \qquad \text{(C.3)}
\end{aligned}$$

The exact fourth derivative of $y(t)$, where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}
y^{(4)}(t) = & \, 2g_x f_{xy} f g + g_x f_x f_y g + 3g_{xy} g^2 f_y + 3g_{xy} f^2 g_x + g_x f_{xx} f^2 + 2g_x f_y g_y g \\
& + 5g_{xy} g_y f g + 3g_{xx} f_y f g + 3g_{xy} f_x f g + 3g_{yy} g_x f g + 3g_{xx} f^2 f_x + g_{xxx} f^3 \\
& + g_{yyy} g^3 + 3g_{xyy} f g^2 + 3g_{xxy} f^2 g + g_y g_x f_x f + g_y^2 g_x f + g_y^3 g + g_x f_x^2 f \\
& + g_x f_{yy} g^2 + f_y g_x^2 f + g_y g_{xx} f^2 + 4g_{yy} g^2 g_y. \qquad \text{(C.4)}
\end{aligned}$$

The remainder term when computing the fourth order Taylor expansion of (4.13e),

where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}\mathcal{R}_{y,k+1}(t) = &\frac{1}{9}g_{xy}f^2g_x + \frac{1}{9}g_{xx}ff_yg + \frac{1}{9}g_{yy}gg_xf + \frac{1}{9}g_{xy}fg_yg + \frac{1}{27}g_{xxx}f^3 \\ &+ \frac{1}{9}g_{xxy}f^2g + \frac{1}{9}g_{xy}f_yg^2 + \frac{1}{9}g_{xyy}fg^2 + \frac{1}{9}g_{yy}g^2g_y + \frac{1}{27}g_{yyy}g^3 \\ &+ \frac{1}{9}g_{xy}f_xfg + \frac{1}{9}g_{xx}f^2f_x. \end{aligned} \tag{C.5}$$

The exact fifth derivative of $x(t)$, where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}x^{(5)}(t) = &f_yg^4 + f_xf^4 + f_{xxxx}f^4 + f_{yyyy}g^4 + 4\,{f_{xx}}^2f^3 + 10\,f_{xx}ff_xf_yg + 4\,f_{xx}ff_yg_yg \\ &+ 14\,f_{xy}gf_yg_xf + 6\,f_{xy}g_ygf_xf + 4\,f_{yy}gg_xf_xf + 10\,f_{yy}gg_yg_xf + 4\,f_{xxxy}gf_yg_xf \\ &+ 12\,f_{xx}f^2f_{xy}g + 4\,f_{xx}ff_{yy}g^2 + 4\,f_{xx}f^2f_yg_x + 6\,f_{xxx}f^2f_yg + 6\,f_{xxy}f^2g_yg \\ &+ 4\,f_{xy}g{f_x}^2f + 4\,f_{xy}g^2f_xf_y + 10\,f_{xy}g^2f_yg_y + 10\,f_{xy}g_xf^2f_x + 6\,f_{xyy}g^2f_xf \\ &+ 8\,f_{xy}f^2g_{xy}g + 4\,f_{xy}fg_{yy}g^2 + 4\,f_{xy}f^2g_yg_x + 4\,f_{xy}f{g_y}^2g + 4\,f_{yy}gg_{xx}f^2 \\ &+ 8\,f_{yy}g^2g_{xy}f + 4\,f_{yy}g^2g_xf_y + 6\,f_{yyy}g^2g_xf + 4\,f_{xxxy}gf_xf^2 + 8\,f_{xxxy}g^2f_{xy}f \\ &+ 4\,f_{xxxy}g{f_x}^2f + 4\,f_{xxxy}g^2f_xf_y + 4\,f_{xxxy}g^2f_yg_y + 12\,f_{xxy}f^2gf_x + 12\,f_{xxy}fg^2f_y \\ &+ 12\,f_{xyy}gf^2g_x + 12\,f_{xyy}g^2fg_y + 6\,f_{yyy}g^3g_y + 4\,f_{xy}g^3f_{yy} + 6\,f_{xyy}g^3f_y \\ &+ 4\,f_{yy}g^3g_{yy} + 4\,f_{xxxy}g^3f_{yy} + 6\,f_{xxx}f^3f_x + 7\,f_{yy}g^2{g_y}^2 + 6\,f_{xxyy}f^2g^2 \\ &+ 3\,f_{yy}{g_x}^2f^2 + 7\,f_{xx}f^2{f_x}^2 + 3\,f_{xx}{f_y}^2g^2 + 8\,{f_{xy}}^2g^2f + 4\,f_{xy}f^3g_{xx} \\ &+ 4\,f_{xyyy}fg^3 + 6\,f_{xxy}f^3g_x. \end{aligned} \tag{C.6}$$

The remainder term when computing the fifth order Taylor series expansion of (4.16g),

where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}
\mathcal{R}_x(t) = {} & \frac{5}{144}f_{xxxy}f^3g + \frac{5}{576}f_{xxxx}f^4 + \frac{5}{576}f_{yyyy}g^4 + \frac{1}{32}f_{xx}^2f^3 + \frac{1}{8}f_{xx}ff_xf_yg \\
& + \frac{1}{24}f_{xx}ff_yg_yg - \frac{1}{36}f_{xy}gf_yg_xf + \frac{1}{12}f_{xy}g_ygf_xf + \frac{1}{12}f_{yy}gg_xf_xf + \frac{1}{8}f_{yy}gg_yg_xf \\
& + \frac{3}{32}f_{xx}f^2f_{xy}g + \frac{1}{32}f_{xx}ff_{yy}g^2 - \frac{1}{18}f_{xx}f^2f_yg_x + \frac{5}{96}f_{xxx}f^2f_yg + \frac{5}{96}f_{xxy}f^2g_yg \\
& + \frac{1}{12}f_{xy}gf_x{}^2f + \frac{1}{16}f_{xy}g^2f_xf_y + \frac{5}{48}f_{xy}g^2f_yg_y + \frac{7}{48}f_{xy}g_xf^2f_x + \frac{7}{96}f_{xyy}g^2f_xf \\
& + \frac{1}{8}f_{xy}f^2g_{xy}g + \frac{1}{16}f_{xy}fg_{yy}g^2 + \frac{1}{16}f_{xy}f^2g_yg_x + \frac{1}{16}f_{xy}fg_y{}^2g + \frac{1}{16}f_{yy}gg_{xx}f^2 \\
& + \frac{1}{8}f_{yy}g^2g_{xy}f - \frac{5}{144}f_{yy}g^2g_xf_y + \frac{5}{96}f_{yyy}g^2g_xf + \frac{1}{8}f_{xxy}f^2gf_x + \frac{5}{48}f_{xxy}fg^2f_y \\
& + \frac{5}{48}f_{xyy}gf^2g_x + \frac{5}{48}f_{xyy}g^2fg_y + \frac{5}{96}f_{yyy}g^3g_y + \frac{1}{32}f_{xy}g^3f_{yy} + \frac{5}{96}f_{xyy}g^3f_y \\
& + \frac{1}{16}f_{yy}g^3g_{yy} + \frac{17}{288}f_{xxx}f^3f_x + \frac{3}{32}f_{yy}g^2g_y{}^2 + \frac{5}{96}f_{xxyy}f^2g^2 + \frac{1}{32}f_{yy}g_x{}^2f^2 \\
& + \frac{5}{48}f_{xx}f^2f_x{}^2 + \frac{1}{32}f_{xx}f_y{}^2g^2 + \frac{1}{16}f_{xy}{}^2g^2f + \frac{1}{16}f_{xy}f^3g_{xx} + \frac{5}{144}f_{xyyy}fg^3 \\
& + \frac{5}{96}f_{xxy}f^3g_x - \frac{7}{72}f_yg_y{}^2g_xf - \frac{5}{16}f_yg_{xx}f^2f_x - \frac{7}{24}f_yg_{xy}f^2g_x - \frac{7}{72}f_yg_yg_{xx}f^2 \\
& - \frac{7}{18}f_yg_{yy}g^2g_y - \frac{17}{144}f_yg_xf_x{}^2f + \frac{1}{144}f_xf_{yyy}g^3 - \frac{7}{72}f_yg_{yyy}g^3 - \frac{7}{24}g_{xy}g^2f_y{}^2 \\
& - \frac{7}{72}f_y{}^2g_x{}^2f - \frac{7}{72}f_yg_{xxx}f^3 - \frac{7}{72}f_yg_y{}^3g + \frac{1}{96}f_x{}^2f_{yy}g^2 - \frac{1}{3}f_yg_{xy}gf_xf \\
& - \frac{35}{72}f_yg_{xy}fg_yg - \frac{17}{144}f_yg_yg_xf_xf - \frac{7}{24}f_yg_{yy}gg_xf - \frac{1}{48}f_xf_yg_{yy}g^2 - \frac{1}{48}f_xf_yg_y{}^2g \\
& + \frac{1}{48}f_xf_{yy}g^2g_y - \frac{17}{144}g_xf_xf_y{}^2g - \frac{7}{36}g_xf_y{}^2g_yg - \frac{7}{24}g_{xx}ff_y{}^2g \\
& - \frac{7}{24}f_yg_{xyy}fg^2 - \frac{7}{24}f_yg_{xxy}f^2g. \qquad \text{(C.7)}
\end{aligned}$$

The remainder term when computing the fifth order Taylor series expansion of (4.17),

where each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}
\mathcal{R}_{\mathcal{X}}(t) = \left(\delta^2 + \delta^3 + \frac{1}{4}\delta^4\right)\Bigg(& - \frac{17}{324} f_x f_y g_y g_x f - \frac{4}{27} f_x f_y g_{xy} f g - \frac{35}{162} f_y g_{xy} f g_y g \\
& - \frac{7}{54} f_y g_{yy} g g_x f - \frac{7}{162} f_y {g_y}^3 g + \frac{5}{1296} f_{xxxx} f^4 + \frac{5}{1296} f_{yyyy} g^4 + \frac{1}{72} {f_{xx}}^2 f^3 \\
& + \frac{1}{36} f_{xy} f^3 g_{xx} + \frac{1}{72} f_{xy} g^3 f_{yy} + 1/36\, {f_{xy}}^2 g^2 f + \frac{17}{648} f_{xxx} f^3 f_x \\
& + \frac{5}{216} f_{xxyy} f^2 g^2 + \frac{5}{216} f_{xxy} f^3 g_x + \frac{5}{216} f_{xyy} g^3 f_y + \frac{1}{36} f_{yy} g^3 g_{yy} \\
& + \frac{1}{72} f_{xx} {f_y}^2 g^2 + \frac{1}{24} f_{yy} g^2 {g_y}^2 + \frac{1}{72} f_{yy} {g_x}^2 f^2 + \frac{5}{216} f_{yyy} g^3 g_y + \frac{5}{324} f_{xyyy} f g^3 \\
& + \frac{5}{108} f_{xx} f^2 {f_x}^2 + \frac{1}{18} f_{xx} f f_x f_y g + \frac{1}{54} f_{xx} f f_y g_y g - \frac{1}{81} f_{xy} g f_y g_x f + \frac{1}{27} f_{xy} g_y g f_x f \\
& + \frac{1}{27} f_{yy} g g_x f_x f + \frac{1}{18} f_{yy} g g_y g_x f + \frac{1}{24} f_{xx} f^2 f_{xy} g + \frac{1}{72} f_{xx} f f_{yy} g^2 - \frac{2}{81} f_{xx} f^2 f_y g_x \\
& + \frac{5}{216} f_{xxx} f^2 f_y g + \frac{5}{216} f_{xxy} f^2 g_y g + \frac{1}{27} f_{xy} g {f_x}^2 f + 1/36\, f_{xy} g^2 f_x f_y + \frac{5}{108} f_{xy} g^2 f_y g_y \\
& + \frac{7}{108} f_{xy} g_x f^2 f_x + \frac{7}{216} f_{xyy} g^2 f_x f + \frac{1}{18} f_{xy} f^2 g_{xy} g + \frac{1}{36} f_{xy} f g_{yy} g^2 \\
& + \frac{1}{36} f_{xy} f^2 g_y g_x + \frac{1}{36} f_{xy} f {g_y}^2 g + \frac{1}{36} f_{yy} g g_{xx} f^2 + \frac{1}{18} f_{yy} g^2 g_{xy} f - \frac{5}{324} f_{yy} g^2 g_x f_y \\
& + \frac{5}{216} f_{yyy} g^2 g_x f + \frac{1}{18} f_{xxy} f^2 g f_x + \frac{5}{108} f_{xxy} f g^2 f_y + \frac{5}{108} f_{xyy} g f^2 g_x + \frac{5}{108} f_{xyy} g^2 f g_y \\
& + \frac{5}{324} f_{xxxy} f^3 g + \frac{1}{108} f_x f_{yy} g^2 g_y - \frac{7}{54} g_{xx} f {f_y}^2 g - \frac{7}{81} g_x {f_y}^2 g_y g - \frac{7}{54} f_y g_{xxy} f^2 g \\
& - \frac{7}{162} f_y g_y g_{xx} f^2 - \frac{7}{54} f_y g_{xy} f^2 g_x - \frac{7}{54} f_y g_{xyy} f g^2 - \frac{14}{81} f_y g_{yy} g^2 g_y \\
& - \frac{7}{162} f_y {g_y}^2 g_x f - \frac{1}{108} f_x f_y {g_y}^2 g - \frac{1}{108} f_x f_y g_{yy} g^2 - \frac{17}{324} {f_x}^2 f_y g_x f \\
& - \frac{17}{324} f_x g_x {f_y}^2 g - \frac{5}{36} f_x f_y g_{xx} f^2 + \frac{1}{216} {f_x}^2 f_{yy} g^2 - \frac{7}{162} f_y g_{xxx} f^3 \\
& - \frac{7}{54} g_{xy} g^2 {f_y}^2 + \frac{1}{324} f_x f_{yyy} g^3 - \frac{7}{162} {f_y}^2 {g_x}^2 f - \frac{7}{162} f_y g_{yyy} g^3 \Bigg). \qquad \text{(C.8)}
\end{aligned}$$

The exact fifth derivative of $y(t)$, where each term is evaluated at $(x(t), y(t))$

$$
\begin{aligned}
y^{(5)}(t) = {} & g_y g^4 + g_x f^4 + g_{xxxx} f^4 + g_{yyyy} g^4 + 4\, {g_{yy}}^2 g^3 + 10\, g_{xx} f f_x f_y g \\
& + 4\, g_{xx} f f_y g_y g + 14\, g_{xy} g f_y g_x f + 6\, g_{xy} g_y g f_x f + 4\, g_{yy} g g_x f_x f + 10\, g_{yy} g g_y g_x f \\
& + 4\, g_{xxxy} g f_y g_x f + 4\, g_{xy} f^3 g_{xx} + 6\, g_{yyy} g^3 g_y + 4\, g_{xy} g^3 f_{yy} + 7\, g_{yy} {g^2 g_y}^2 \\
& + 6\, g_{xxx} f^3 f_x + 4\, g_{xx} f^3 f_{xx} + 3\, g_{xx} {f_y}^2 g^2 + 6\, g_{xyy} g^3 f_y + 6\, g_{xxy} f^3 g_x \\
& + 8\, {g_{xy}}^2 f^2 g + 4\, g_{xxxy} g^3 f_{yy} + 6\, g_{xxyy} f^2 g^2 + 3\, g_{yy} {g_x}^2 f^2 + 7\, g_{xx} f^2 {f_x}^2 \\
& + 4\, g_{xyyy} f g^3 + 8\, g_{xx} f^2 f_{xy} g + 4\, g_{xx} f f_{yy} g^2 + 4\, g_{xx} f^2 f_y g_x + 6\, g_{xxx} f^2 f_y g \\
& + 6\, g_{xxy} f^2 g_y g + 4\, g_{xy} g f_{xx} f^2 + 8\, g_{xy} g^2 f_{xy} f + 4\, g_{xy} g {f_x}^2 f + 4\, g_{xy} g^2 f_x f_y \\
& + 10\, g_{xy} g^2 f_y g_y + 10\, g_{xy} g_x f^2 f_x + 6\, g_{xyy} g^2 f_x f + 12\, g_{xy} f g_{yy} g^2 + 4\, g_{xy} f^2 g_y g_x \\
& + 4\, g_{xy} f {g_y}^2 g + 4\, g_{yy} g g_{xx} f^2 + 4\, g_{yy} g^2 g_x f_y + 6\, g_{yyy} g^2 g_x f + 4\, g_{xxxy} g f_{xx} f^2 \\
& + 8\, g_{xxxy} g^2 f_{xy} f + 4\, g_{xxxy} g {f_x}^2 f + 4\, g_{xxxy} g^2 f_x f_y + 4\, g_{xxxy} g^2 f_y g_y + 12\, g_{xxy} f^2 g f_x \\
& + 12\, g_{xxy} f g^2 f_y + 12\, g_{xyy} g f^2 g_x + 12\, g_{xyy} g^2 f g_y. \qquad \text{(C.9)}
\end{aligned}
$$

The remainder term when computing the fifth order Taylor expansion of (4.23g), where

each term is evaluated at $(x(t), y(t))$

$$\begin{aligned}\mathcal{R}_{y,k+1}(t) = &\frac{1}{144}\, g_y g_{xxx} f^3 + \frac{5}{576}\, g_{xxxx} f^4 + \frac{5}{576}\, g_{yyyy} g^4 + \frac{1}{32}\, {g_{yy}}^2 g^3 + \frac{5}{48}\, g_{xx} f f_x f_y g \\ &+ \frac{1}{16}\, g_{xx} f f_y g_y g + \frac{7}{48}\, g_{xy} g f_y g_x f + \frac{1}{12}\, g_{xy} g_y g f_x f + \frac{1}{24}\, g_{yy} g g_x f_x f \\ &+ \frac{1}{8}\, g_{yy} g g_y g_x f + \frac{1}{32}\, g_{xy} f^3 g_{xx} + \frac{1}{16}\, {g_{xy}}^2 f^2 g + \frac{1}{24}\, g_{xx} f^3 f_{xx} + \frac{7}{96}\, g_{xx} f^2 {f_x}^2 \\ &+ \frac{5}{48}\, g_{yy} g^2 {g_y}^2 + \frac{5}{96}\, g_{xxx} f^3 f_x + \frac{5}{96}\, g_{xyy} g^3 f_y + \frac{1}{24}\, g_{xy} g^3 f_{yy} + \frac{1}{32}\, g_{yy} {g_x}^2 f^2 \\ &+ \frac{5}{96}\, g_{xxyy} f^2 g^2 + \frac{5}{96}\, g_{xxy} f^3 g_x + \frac{17}{288}\, g_{yyy} g^3 g_y + \frac{5}{144}\, g_{xyyy} f g^3 + \frac{1}{32}\, g_{xx} {f_y}^2 g^2 \\ &+ \frac{1}{12}\, g_{xx} f^2 f_{xy} g + \frac{1}{24}\, g_{xx} f f_{yy} g^2 + \frac{1}{24}\, g_{xx} f^2 f_y g_x + \frac{5}{96}\, g_{xxx} f^2 f_y g + \frac{7}{96}\, g_{xxy} f^2 g_y g \\ &+ \frac{1}{24}\, g_{xy} g f_{xx} f^2 + \frac{1}{12}\, g_{xy} g^2 f_{xy} f + \frac{1}{24}\, g_{xy} g {f_x}^2 f + \frac{1}{24}\, g_{xy} g^2 f_x f_y + \frac{1}{8}\, g_{xy} g^2 f_y g_y \\ &+ \frac{5}{48}\, g_{xy} g_x f^2 f_x + \frac{5}{96}\, g_{xyy} g^2 f_x f + \frac{3}{32}\, g_{xy} f g_{yy} g^2 + \frac{1}{16}\, g_{xy} f^2 g_y g_x + \frac{1}{12}\, g_{xy} f {g_y}^2 g \\ &+ \frac{1}{32}\, g_{yy} g g_{xx} f^2 + \frac{1}{24}\, g_{yy} g^2 g_x f_y + \frac{5}{96}\, g_{yyy} g^2 g_x f + \frac{5}{48}\, g_{xxy} f^2 g f_x + \frac{5}{48}\, g_{xxy} f g^2 f_y \\ &+ \frac{5}{48}\, g_{xyy} g f^2 g_x + \frac{1}{8}\, g_{xyy} g^2 f g_y + \frac{1}{48}\, g_y g_{xx} f^2 f_x + \frac{5}{144}\, g_{xxxy} f^3 g + \frac{1}{96}\, {g_y}^2 g_{xx} f^2. \end{aligned} \tag{C.10}$$

Appendix D

Computational Mesh Information

Mesh data used for computing the maximum theoretical speed-up given by the local time-stepping methods.

Mesh 1-A.

	Interior	Large Interface	Small Interface	Total
Bin 0	156	26	0	182
Bin 1	65	30	28	123
Bin 2	62	41	34	137
Bin 3	357	0	47	404
Total	640	97	109	846

Mesh 1-B.

	Interior	Large Interface	Small Interface	Total
Bin 0	674	54	0	728
Bin 1	369	67	56	492
Bin 2	421	92	71	548
Bin 3	1517	0	99	1616
Total	2945	213	226	3384

Mesh 1-C.

	Interior	Large Interface	Small Interface	Total
Bin 0	2802	110	0	2912
Bin 1	1715	141	112	1968
Bin 2	1851	196	145	2192
Bin 3	6260	0	204	6464
Total	12629	447	460	13536

Mesh 1-D.

	Interior	Large Interface	Small Interface	Total
Bin 0	11426	222	0	11648
Bin 1	7359	289	224	7872
Bin 2	8071	404	293	8768
Bin 3	25445	0	411	25856
Total	52301	915	928	54144

Mesh 2-A.

	Interior	Large Interface	Small Interface	Total
Bin 0	39	18	0	57
Bin 1	25	19	18	62
Bin 2	20	17	18	55
Bin 3	29	17	19	65
Bin 4	13	18	18	49
Bin 5	26	20	17	63
Bin 6	28	28	24	80
Bin 7	150	6	27	183
Bin 8	0	0	2	2
Total	330	143	143	616

Mesh 2-B.

	Interior	Large Interface	Small Interface	Total
Bin 0	187	41	0	228
Bin 1	165	42	41	248
Bin 2	138	41	41	220
Bin 3	179	38	43	260
Bin 4	121	36	39	196
Bin 5	170	47	35	252
Bin 6	210	59	51	320
Bin 7	662	12	58	732
Bin 8	2	0	6	8
Total	1843	316	314	2464

Mesh 2-C.

	Interior	Large Interface	Small Interface	Total
Bin 0	825	87	0	912
Bin 1	817	88	87	992
Bin 2	704	89	87	880
Bin 3	869	80	91	1040
Bin 4	631	72	81	784
Bin 5	836	101	71	1008
Bin 6	1054	121	105	1280
Bin 7	2784	24	120	2928
Bin 8	14	0	18	32
Total	8534	662	660	9856

Mesh 2-D.

	Interior	Large Interface	Small Interface	Total
Bin 0	3469	179	0	3648
Bin 1	3609	180	179	3968
Bin 2	3156	185	179	3520
Bin 3	3809	164	187	4160
Bin 4	2827	144	165	3136
Bin 5	3680	209	143	4032
Bin 6	4662	245	213	5120
Bin 7	11420	48	244	11712
Bin 8	86	0	42	128
Total	36718	1354	1352	39424

Shallow Water Equations Mesh.

	Interior	Large Interface	Small Interface	Total
Bin 0	26288	41	0	26329
Bin 1	50	16	43	109
Bin 2	37	15	16	68
Bin 3	34	8	16	58
Bin 4	8	0	8	16
Total	26417	80	83	26580

Smooth Vortex Mesh.

	Interior	Large Interface	Small Interface	Total
Bin 0	4541	199	0	4740
Bin 1	658	303	219	1180
Bin 2	1536	657	331	2524
Bin 3	1152	0	644	1796
Total	7887	1159	1194	10240