Explicit Runge-Kutta time-stepping with the discontinuous Galerkin method

by

Khalida Parveen

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, 2018

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

In this thesis, the discontinuous Galerkin method is used to solve the hyperbolic equations. The DG method discretizes a system into a semi-discrete system and a system of ODEs is obtained. To solve this system of ODEs efficiently, numerous time-stepping techniques can be used. The most popular choice is Runge-Kutta methods. Classical Runge-Kutta methods need a lot of space in the computer memory to store the required information. The 2N-storage time-steppers store the values in two registers, where N is the dimension of the system. The 2N-storage schemes have more stages than classical RK schemes but are more efficient than classical RK schemes.

Several 2N-storage time-stepping techniques have been used reported in the literature. The linear stability condition is found using the eigenvalue analysis of DG method and spectrum of DG method has been scaled to fit inside the absolute stability regions of 2N-storage schemes. The one-dimensional advection equation has been solved using RK-DG pairings. It is shown that these high-order 2N-storage RK schemes are a good choice for use with the DG method to improve efficiency and accuracy over classical RK schemes.

Acknowledgments

In the name of Allah, the Most Beneficent, the Most Merciful, I would like to convey my sincere gratitude to my advisor Professor Lilia Krivodonova, for her guidance, advice and support during my research. I am extremely grateful to Professor Sander Rhebergen and Professor Giang Tran, for their time and valuable input as members of my examining committee.

I would like to express my deep gratitude and thanks to my dear family, for their continuous sacrifices, support, patience, love and prayers. I'll be indebted forever to my sweet sister Aamira Khokhar, for her guidance and support during my studies in Canada. I am grateful to my husband, Asim Rehan, for his love, support and understanding during my studies at the University of Waterloo.

I would also, like to thank my friend Humeyra Kiyak, for her precious time, help, support and coffee breaks during my studies.

Finally, I would like to extend special thanks to my late father-in-law, Abdul Razzaq, who did not live to see this dream come true. You will always be missed and remembered.

Dedication

This is dedicated to my cherished family, loving husband and adored children, *Mahnoor* and *Muhammad Hashim*.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Physical phenomena emerging from the real world can be modeled using differential equations. The differential equations are divided into two categories, partial differential equations and ordinary differential equations. The solution of partial differential equations are the main focus of current research. The first-order systems of partial differential equations in divergence form are called hyperbolic conservation laws. The solution of conservation laws has drawn the attention of scientists because the solution of such system may not exist in an analytical form. Hence, use of numerical methods has become essential. Thus, various numerical methods has been designed for the solution of hyperbolic systems of partial differential equations (PDEs).

When applying a numerical method, accuracy, stability and cost of computations are the primary concerns for the user. A numerical method is distinguished by the order of accuracy, i.e. the convergence rate and efficiency. Numerous numerical methods are designed for the solution of hyperbolic systems while keeping these requirements in mind and the discontinuous Galerkin method is one of them. The discontinuous Galerkin method is a blend of discretization techniques. It is based on local polynomial approximations, as in finite element method. It is also discontinuous at the boundaries of each element, unlike finite element method. This ambiguity is resolved by using a numerical flux as in finite volume method. The accuracy of the DG method can be improved by increasing the number of elements in the spatial domain of the given problem that results in more computations and memory use. The stability and accuracy of the DG method has been studied in depth in [11, 18, 22, 26]. The DG method is used for the solution of wave propagation problems [20, 12].

The discretization of time derivative is required when a discretization technique, the DG method is applied to one-dimensional advection equation. The system of ODEs from the spatial discretization can be solved by, for example, explicit time integrators. Explicit time integration schemes are easy to apply and adaptively adjust the time-step. However, these schemes may produce unstable solutions due to the Courant-Friedrich-Levy (CFL) number. The solution of wave propagation problems have been presented by explicit classical Runge-Kutta methods in [11, 24]. In addition, efficient 2N-storage RK schemes have been developed to minimize the cost computation of RKDG method, see [41]. 2N-storage schemes require two registers to store the values in computer memory, where N is the dimension of the system.

Firstly, a 2N-storage scheme was introduced by Carpenter and Kennedy [9], Carpenter(5,4). Allampalli *et al.* [2] derived 2N-storage schemes with respect to optimal stability region. Hu *et al.* [21] developed efficient 2N-storage RK schemes and optimized the dissipation and dispersion of wave propagation problems. Calvo *et al.* [8], also constructed 2N-storage schemes with respect to stability and accuracy. Pirozzoli [32], and Bernardini and Pirozzoli [5] established a general strategy for performance analysis of RK schemes.

Toulorge and Desmet [38] have constructed three low-storage techniques, RKF84, RKC84 and RKC73 to minimize the computational cost of the RKDG method for the solution of hyperbolic conservation laws. Toulorge and Desmet [38], and Bernardini and Pirozzoli [5] solved the under-determined system for the estimation of coefficients with increasing c_i.

However, not all RK schemes were designed for the discontinuous Galerkin method. This thesis describes the implementation of optimal 2N-storage RK time steppers associated with the DG method. The linear stability condition is found using RK-DG pairings and efficiency is improved.

This thesis is composed of four chapters. Chapter 1 deals with introduction of conservation laws, the discontinuous Galerkin method, Legendre polynomials, numerical flux, numerical quadrature, and CFL condition required for the implementation of the DG method. Chapter 2 provides the analysis of spectral values of the DG method to find the linear stability condition for the implementation of 2N-storage schemes with the DG method. Chapter 3 reports the detailed study of numerical solution of initial value problems. The main topics addressed in this chapter are; explicit Runge-Kutta methods, local and global error, absolute stability regions. In Chapter 4, 2N-storage efficient RK schemes have been stated with their coefficients in tables. The intervals of absolute stability regions and CFL numbers are presented using RK-DG pairings as well. Chapter 4 also includes the numerical simulations of these 2N-storage RK methods performed on one-dimensional advection equation with the sinusoidal initial profile. In conclusion, significant perfor-

mance of high-order 2N-storage RK schemes has been shown in terms of accuracy and computational cost.

1.1 Conservation laws

The time-dependent first order systems of partial differential equations in divergence form are called hyperbolic conservation laws. These systems present phenomena emerging in mathematical physics; thus, the subject is classical and can be traced back to Euler (1755) with latter contributions from Stokes, Riemann, Weyle and von Neumann among many others. Let us consider one-dimensional conservation law

$$\begin{aligned} \frac{\partial u_1}{\partial t} + \frac{\partial}{\partial x}[f_1(u_1, u_2, ..., u_m)] = 0, \\ \vdots \\ \frac{\partial u_m}{\partial t} + \frac{\partial}{\partial x}[f_m(u_1, u_2, ..., u_m)] = 0. \end{aligned} \tag{1.1}$$

The system (1.1) can be written in a vector notation as

$$\frac{\partial}{\partial t}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} f_1(u_1, u_2, ..., u_m) \\ f_2(u_1, u_2, ..., u_m) \\ \vdots \\ f_m(u_1, u_2, ..., u_m) \end{bmatrix} = 0, \tag{1.2}$$

or, in a compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{u}) = 0. \tag{1.3}$$

The components of the vector $\mathbf{u} = (u_1, ..., u_m)$ are conserved quantities, for example, mass or energy. The function $\mathbf{f} = (f_1, f_2, ..., f_m)$ in $\mathbb{R}^m$ is called the flux function. A vector $\mathbf{u}(x, t)$ is the solution for (1.3) if and only if it satisfies the system (1.1) and initial conditions $\mathbf{u}(x, 0) = \mathbf{u}_0(x)$, $\forall x \in \mathbb{R}$. For each flux function $\mathbf{f}$, a Jacobian matrix of dimension $m \times m$ is defined as

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \cdots & \frac{\partial f_m}{\partial u_m} \end{bmatrix}. \tag{1.4}$$

The system (1.3) is hyperbolic, if the matrix A has m real eigenvalues $\lambda_1 < \lambda_2 < ... < \lambda_m$ and has the full set of eigenvectors. The eigenvalues of matrix A are known as characteristic speeds. Moreover, the system of conservation law is strictly hyperbolic if, the matrix A has m real and distinct eigenvalues.

1.2 The discontinuous Galerkin method

We consider a simple example of (1.3), i.e., a scalar conservation law,

$$u_t + f(u)_x = 0, \quad x \in (a,b), \quad t > 0, \tag{1.5a}$$

$$u(x,0) = u_0(x), \tag{1.5b}$$

with suitable boundary conditions. We assume $f(u)$ to be sufficiently smooth and convex function over the domain $\Omega = [a,b]$. To construct the DG scheme, the domain Ω is divided into a finite mesh of N elements

$$\Omega = \bigcup_{j=1}^{N} \Omega_j, \tag{1.6}$$

where $\Omega_j = [x_j, x_{j+1}]$ are distinct and non-overlapping elements. The length of element Ω_j is $\Delta x_j = x_{j+1} - x_j$. Equation (1.5a) is multiplied by a smooth test function v. Then, integrated over Ω_j by parts to get

$$\int_{\Omega_j} u_t v dx - \int_{\Omega_j} f(u) v' dx + f(u) v|_{x_j}^{x_{j+1}} = 0. \tag{1.7}$$

Next, u is approximated on Ω_j with a function U_j in a finite-dimensional subspace of H^1, called the finite element space and denoted by $V(\Omega_j)$, that has dimension $p+1$ with a basis

$$\Phi = \{\phi_i\}_{i=0}^{p}. \tag{1.8}$$

Then, $U_j \in V(\Omega_j)$ can be expressed as a linear combination of the basis functions with coefficients $c_{ij}(t)$ as

$$U_j(x,t) = \sum_{i=0}^{p} c_{ij}(t)\phi_i(x). \tag{1.9}$$

Using (1.9), the global approximation U can be written as a direct sum of the local approximations U_j

$$U = \bigoplus_{j=1}^{N} U_j. \tag{1.10}$$

The general Galerkin formulation choses the test functions v from the same finite element space as the approximation U_j. This implies that equation (1.7) should hold for all $v \in V(\Omega_j)$. Thus, we select $v = \phi_k$, $k = 0, 1, 2, ..., p$. Hence, (1.7) becomes

$$\frac{\partial}{\partial t} \int_{\Omega_j} U_j \phi_k dx - \int_{\Omega_j} f(U_j) \phi_k' dx + f(U_j)\phi_k|_{x_j}^{x_{j+1}} = 0, \quad k = 0, 1, ..., p. \tag{1.11}$$

Substituting (1.9) in (1.11) results in

$$\frac{\partial}{\partial t}\int_{\Omega_j}(\sum_{i=0}^{p}c_{ij}\phi_i)\phi_k dx - \int_{\Omega_j} f(\sum_{i=0}^{p}c_{ij}\phi_i)\phi_k' dx + f(\sum_{i=0}^{p}c_{ij}\phi_i)\phi_k|_{x_j}^{x_{j+1}} = 0, \quad k = 0, 1, ..., p. \tag{1.12}$$

We introduce a linear mapping to map each element Ω_j to the canonical element $\Omega = [-1, 1]$

$$\xi = \frac{2}{\Delta x_j}(x - \frac{x_j + x_{j+1}}{2}). \tag{1.13}$$

Hence, (1.12) becomes

$$\frac{\Delta x_j}{2}\frac{\partial}{\partial t}\int_{-1}^{1}(\sum_{i=0}^{p}c_{ij}\bar{\phi}_i)\bar{\phi}_k d\xi - \int_{-1}^{1} f(\sum_{i=0}^{p}c_{ij}\bar{\phi}_i)\bar{\phi}_k' d\xi + f(\sum_{i=0}^{p}c_{ij}\bar{\phi}_i)\bar{\phi}_k\ |_1^{-1} = 0, \quad k = 0, 1, ..., p, \tag{1.14}$$

where $\bar{\phi}(\xi) = \bar{\phi}(x(\xi))$. Next, we introduce a suitable choice for basis functions using the Legendre polynomials.

1.3 Legendre polynomials as a DG basis

The most common choice for the basis functions for one-dimensional problems is the Legendre polynomials $P_k(\xi)$. These polynomials form an orthogonal system on $[-1, 1]$ with respect to the L^2 inner product with the normalization

$$\int_{-1}^{1} P_k(\xi)P_i(\xi)d\xi = \frac{2}{2k+1}\delta_{ki}, \tag{1.15}$$

where $\frac{2}{2k+1}$ is a factor due to a specific normalization and δ_{ki} is the Kronecker delta [1]. With the chosen normalization, the Legendre polynomials have the following properties

$$P_k(1) = 1, \quad P_k(-1) = (-1)^k. \tag{1.16}$$

The Legendre polynomials can be defined recursively as

$$P_0(\xi) = 1, \tag{1.17}$$

$$P_1(\xi) = \xi, \tag{1.18}$$

$$(k+1)P_{k+1}(\xi) = (2k+1)\xi P_k(\xi) - (k)P_{k-1}(\xi), \quad k = 2, 3, ... \tag{1.19}$$

The first five Legendre polynomials are given below

$$
\begin{aligned}
P_0 =& 1, \\
P_1 =& \xi, \\
P_2 =& \frac{1}{2}(3\xi^2 - 1), \\
P_3 =& \frac{1}{2}(5\xi^3 - 3\xi), \\
P_4 =& \frac{1}{8}(35\xi^4 - 30\xi^2 + 3),
\end{aligned}
$$

and we show them in Figure 1.1.

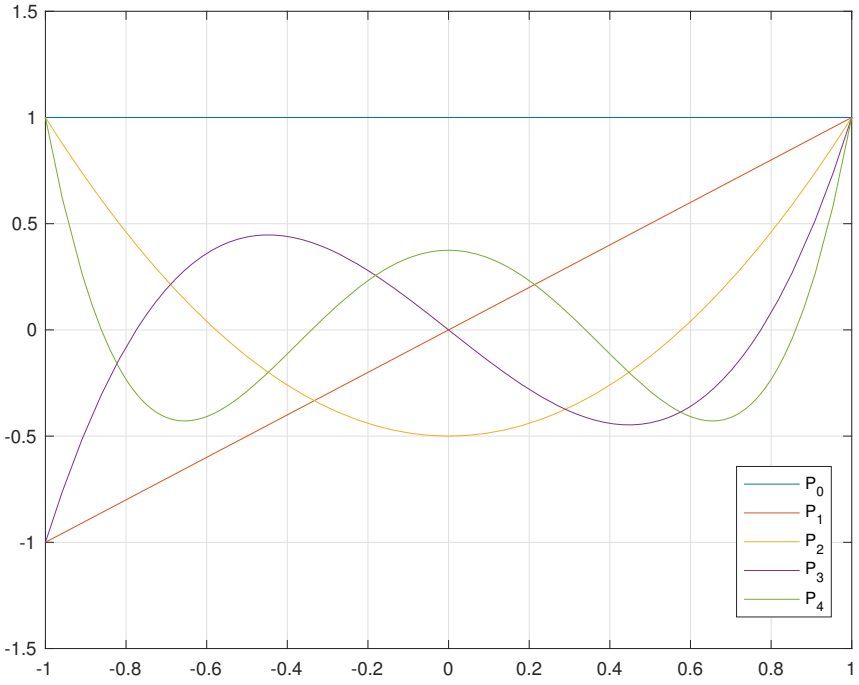

Figure 1.1: The first five Legendre polynomials.

1.4 Numerical flux

Flux function $f(U)$ represents the flow of some physical quantity that passes through a surface or substance. Since the approximated solution is discontinuous at interfaces of elements, this implies that $U_j(1,t) \neq U_{j+1}(-1,t)$ in (1.14). A Riemann solver is used to resolve these uncertainties. It takes information from both the left and right element to approximate the flux function $f(U)$ the interface between them. In discontinuous Galerkin method, for linear scalar conservation laws, the upwind flux is used. The numerical flux value at the left endpoints of element Ω_j at $x = x_j$, is given by

$$f(U) = \begin{cases} f\{U_{j-1}(1)\}, & f_u > 0, \\ f\{U_j(-1)\}, & f_u < 0. \end{cases}$$

The right endpoint is treated similarly. For a linear advection equation with $f(U) = \alpha u$, the upwind flux at $x = x_j$, is given by

$$f(U) = \begin{cases} \alpha\{U_{j-1}(1)\}, & \alpha > 0, \\ \alpha\{U_j(-1)\}, & \alpha < 0. \end{cases} \tag{1.20}$$

1.5 Numerical quadrature

We use numerical quadrature to project the initial condition (1.5b) onto the finite element space using L^2 projection. Thus, we get

$$\int_{-1}^{1} u_0(x(\xi))P_k(\xi)d\xi = \int_{-1}^{1} u(\xi)P_k(\xi)d\xi, \quad k = 0, 1, ..., p. \tag{1.21}$$

Using (1.9) in (1.21), we get

$$\int_{-1}^{1} u_0(\xi)P_k(\xi)d\xi = \int_{-1}^{1} \big(\sum_{i=0}^{p} c_{ij}(0)P_i(\xi)\big)P_k(\xi)d\xi, \quad k = 0, 1, ..., p. \tag{1.22}$$

Using the orthogonality property of Legendre polynomials (1.14), the initial solution coefficients $\mathbf{c}$ are computed as

$$c_{jk}(0) = \frac{2k+1}{2}\int_{-1}^{1} u_0(x(\xi))P_k(\xi)d\xi, \quad k = 0, 1, ..., p. \tag{1.23}$$

A Gaussian quadrature rule is used to approximate (1.23) and the integral involved in (1.14). The Gauss-Legendre quadrature rule is exact for the polynomials of degree $2p-1$, for linear case. The Gauss-Legendre quadrature rule can be used to approximate the integral $\int_{-1}^{1} f(U_j)P_k' d\xi$ involved in (1.14). For linear one-dimensional problem, Gauss-Legendre quadrature rule is defined as

$$\int_{-1}^{1} f(\xi)d\xi \approx \sum_{i=1}^{n} w_i f(\xi_i). \tag{1.24}$$

where w_i are weights and ξ_i are called nodes. Some nodes and weights of Gauss-Legendre rule are presented in Table 1.1.

n	nodes of $P_n(\xi)$	weights w_i
2	$-\frac{1}{\sqrt{3}}$	1
	$\frac{1}{\sqrt{3}}$	1
3	$-\frac{3}{\sqrt{5}}$	$\frac{5}{9}$
	0	$\frac{8}{9}$
	$\frac{3}{\sqrt{5}}$	$\frac{5}{9}$
4	$\frac{-\sqrt{(15+2\sqrt{30})}}{35}$	$\frac{(18-\sqrt{30})}{36}$
	$\frac{-\sqrt{(15+2\sqrt{30})}}{35}$	$\frac{(18+\sqrt{30})}{36}$
	$\frac{\sqrt{(15+2\sqrt{30})}}{35}$	$\frac{(18-\sqrt{30})}{36}$
	$\frac{-\sqrt{(15+2\sqrt{30})}}{35}$	$\frac{(18-\sqrt{30})}{36}$

Table 1.1: Gauss-Legendre nodes and weights.

1.6 The CFL condition for PDEs with classical Runge-Kutta method

A numerical ODE solver is used to solve the system of ODEs (1.14) that requires to move the solution forward in time using a stable time-step. It is necessary to choose a sufficiently small value of Δt to maintain the stability for an explicit numerical ODE solver. The DG method using a Runge-Kutta time integration scheme of order $p+1$ requires

$$\Delta t_n \leq \frac{\Delta x_{min}}{\lambda(2p+1)}, \tag{1.25}$$

where λ is the largest characteristic velocity in magnitude at t_n and Δx_{min} is the smallest element size. For linear problems, the characteristic velocity is $\lambda = \alpha$ and for non-linear problems, it becomes, see [10]

$$\lambda = \max_j |f_u(U)|. \tag{1.26}$$

Chapter 2

An analysis of the spectrum of the discontinuous Galerkin method

For one-dimensional linear advection equation, an explicit expression for the eigenvalues of the semi-discrete discontinuous Galerkin method was derived in [26]. A higher order ODE solver can be used to solve the system of ODEs after applying DG method to linear advection equation, but the question is, how one can find the essential condition for the stability of the numerical method? It is shown in [26] that the time step Δt should be small enough for spectrum of the DG method to fit inside the absolute stability region of the chosen higher order numerical ODE solver.

2.1 The linear advection equation

We consider the one-dimensional linear advection equation which is an example of (1.3) with $f(u) = \alpha u$,

$$u_t + \alpha u_x = 0, \tag{2.1}$$

with the initial condition $u(x_0) = u_0(x)$, and periodic boundary conditions. Without loss of generality, we assume that $\alpha > 0$. The DG method (1.14) is applied to (2.1) with $f(u) = \alpha u$. We use the mapping (1.13) on each element Ω_j to map to a canonical element $\Omega = [-1, 1]$. Also, we use the upwind flux (1.20) and the Legendre polynomials as basis

functions. Thus, (2.1) becomes

$$\frac{\Delta x}{2k+1}\frac{d}{dt}\int_{-1}^{1}(\sum_{i=0}^{p}c_{ji}P_i)P_k d\xi+\alpha\sum_{i=0}^{p}c_{ji}P_k(1)-\alpha\sum_{i=0}^{p}c_{j-1,i}P_k(-1)-\alpha\int_{-1}^{1}(\sum_{i=0}^{p}c_{ji}P_i)P_k' d\xi=0,\quad k=0,1,...,p. \tag{2.2}$$

Using the orthogonality property (1.15) and normalization of the Legendre polynomials (1.16), we obtain

$$\frac{\Delta x}{2k+1}\dot{c}_{jk}=-\alpha\left(\sum_{i=0}^{p}c_{ji}-(-1)^k\sum_{i=0}^{p}c_{j-1,i}\right)+\alpha\int_{-1}^{1}\left(\sum_{i=0}^{p}c_{ji}P_i\right)P_k' d\xi,\quad k=0,1,...,p, \tag{2.3}$$

where the dot in $\dot{c}_{jk}$ presents the derivative with respect to t. Rearranging like terms of c_{ji} returns

$$\dot{c}_{jk}=\alpha\frac{2k+1}{\Delta x}\left[(-1)^k\sum_{i=0}^{p}c_{j-1,i}+\sum_{i=0}^{p}\left(\int_{-1}^{1}P_iP_k' d\xi-1\right)c_{ji}\right],\quad k=0,1,...,p. \tag{2.4}$$

Introducing a local vector of degrees of freedoms $\mathbf{c}_{j-1,i}=[c_{j-1,0},...,c_{j-1,p}]^T$, gives

$$\begin{aligned}\sum_{i=0}^{p}c_{j-1,i}&=[1,1,...,1][c_{j-1,0},c_{j-1,1},...,c_{j-1,p}]^T,\\&=[1,1,...,1]\mathbf{c}_{j-1},\\&=\mathbf{1}\mathbf{c}_{j-1}.\end{aligned}$$

The second sum in (2.4) can be treated similarly and (2.4) can be expressed in a vector notation as

$$\dot{c}_{jk}=\alpha\frac{2k+1}{\Delta x}\left((-1)^k[1,1,...,1]\mathbf{c}_{j-1}+\left[\int_{-1}^{1}P_0P_k' d\xi-1,...,\int_{-1}^{1}P_pP_k' d\xi-1\right]\mathbf{c}_j\right),\quad k=0,1,...,p. \tag{2.5}$$

The integrals involved in (2.5) can be written in an $n\times n$ square matrix A_n as

$$A_n=\begin{bmatrix}\int_{-1}^{1}P_0P_0' d\xi-1 & ... & \int_{-1}^{1}P_{n-1}P_0' d\xi-1\\ 3\left(\int_{-1}^{1}P_0P_1' d\xi-1\right) & ... & 3\left(\int_{-1}^{1}P_{n-1}P_1' d\xi-1\right)\\ \vdots & & \vdots\\ (2n-1)\left(\int_{-1}^{1}P_0P_{n-1}' d\xi-1\right) & ... & (2n-1)\left(\int_{-1}^{1}P_{n-1}P_{n-1}' d\xi-1\right)\end{bmatrix}, \tag{2.6}$$

where $n = p + 1$. Also, A_n can be written in a compact form as

$$A_n = (a_{ij}) = \left((2i-1)(\int_{-1}^{1} P_{j-1}P'_{i-1}d\xi - 1)\right). \tag{2.7}$$

Note that derivatives of the Legendre polynomials satisfy [1],

$$(2k+1)P_k = P'_{k+1} - P'_{k-1}, \tag{2.8}$$

and

$$P'_{k+1} = (2k+1)P_k + (2(k-2)+1)P_{k-2} + (2(k-4)+1)P_{k-4} + ... \tag{2.9}$$

Using derivatives (2.9) of the Legendre polynomials to simplify the entries in A_n, the integrals in A_n can be written as

Case 1: when $k \leq i$

$$\begin{aligned}\int_{-1}^{1} P_i P'_k d\xi &= \int_{-1}^{1} P_i[(2(k-1)+1)P_{k-1} + (2(k-3)+1)P_{k-3} + (2(k-5)+1)P_{k-5} + ...]d\xi \\ &= (2(k-1)+1)\int_{-1}^{1} P_i P_{k-1}d\xi + (2(k-3)+1)\int_{-1}^{1} P_i P_{k-3}d\xi \\ &+ (2(k-5)+1)\int_{-1}^{1} P_i P_{k-5}d\xi + ...\end{aligned} \tag{2.10}$$

Since $k \leq i$, and using the orthogonality property of Legendre polynomials (1.15) in (2.10) the integrals $\int_{-1}^{1} P_i P_{k-1}d\xi$, $\int_{-1}^{1} P_i P_{k-3}d\xi$, $\int_{-1}^{1} P_i P_{k-5}d\xi$ and so on are equal to zero. Hence, the above expression yields

$$\int_{-1}^{1} P_i P'_k d\xi = 0, \quad k \leq i. \tag{2.11}$$

Case 2: when $k > i$ and $(k-i) = odd$. For some m, $k-(2m+1) = i$. One of the integrals involved in the expression (2.10) is non-zero. Using the orthogonality property of Legendre polynomials (1.15) and $\delta_{i,k-(2m+1)} = 1$, we can evaluate the expression (2.10) as

$$\begin{aligned}\int_{-1}^{1} P_i P'_k d\xi &= (2(k-1)+1)\int_{-1}^{1} P_i P_{k-1}d\xi + (2(k-3)+1)\int_{-1}^{1} P_i P_{k-3}d\xi \\ &+ (2(k-5)+1)\int_{-1}^{1} P_i P_{k-5}d\xi + ... \\ &= \int_{-1}^{1} P_i P_{k-(2m+1)}d\xi \\ &= \int_{-1}^{1} (P_i)^2 d\xi = (2i+1)\frac{2}{(2i+1)} = 2.\end{aligned} \tag{2.12}$$

Case 3: when $k > i$, $(k-i) = even$, using the orthogonality of Legendre polynomials (1.15), all integrals involved in the expression (2.10) are equal to zero. In general, the entries in A_n can be summarized as

$$\int_{-1}^{1} P_i P_k' d\xi = \begin{cases} 0, & k \leq i, \\ 2, & k > i, \text{and } (k-i) \equiv 1 (mod 2), \\ 0, & k > i, \text{and } (k-i) \equiv 0 (mod 2). \end{cases} \tag{2.13}$$

Hence, A_n can be expressed as

$$A_n = - \begin{bmatrix} a_1 & a_1 & a_1 & ... & a_1 & a_1 \\ -a_2 & a_2 & a_2 & ... & a_2 & a_2 \\ a_3 & -a_3 & a_3 & ... & a_3 & a_3 \\ -a_4 & a_4 & -a_4 & ... & a_4 & a_4 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ (-1)^{n-2} a_{n-1} & (-1)^{n-3} a_{n-1} & (-1)^{n-4} a_{n-1} & ... & a_{n-1} & a_{n-1} \\ (-1)^{n-1} a_n & (-1)^{n-2} a_n & (-1)^{n-3} a_n & ... & -a_n & a_n \end{bmatrix}, \tag{2.14}$$

where $a_i = 2i - 1, i = 1, 2, ..., n$. For instance,

$$A_4 = - \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & 3 & 3 & 3 \\ 5 & -5 & 5 & 5 \\ 7 & 7 & -7 & 7 \end{bmatrix}.$$

In addition, a square matrix D_n is defined as

$$D_n = \mathbf{r}_n [1, 1, ..., 1],$$

where $\mathbf{r}_n = [1, -3, ..., (-1)^{n-1}(2n-1)]^T$. The matrix D_n has the form

$$D_n = \begin{bmatrix} 1 & ... & 1 \\ -3 & ... & -3 \\ \vdots & & \vdots \\ (-1)^{n-1}(2n-1) & ... & (-1)^{n-1}(2n-1) \end{bmatrix}. \tag{2.15}$$

For example,

$$D_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -3 & -3 & -3 \\ 5 & 5 & 5 & 5 \\ -7 & -7 & -7 & -7 \end{bmatrix}.$$

On each element j, with $k = 0, 1, ..., p$ in (2.5), we have

$$(2k+1)\left[\int_{-1}^{1} P_0 P_k' d\xi - 1, ..., \int_{-1}^{1} P_p P_k' d\xi - 1\right]\mathbf{c}_j = A_n \mathbf{c}_j, \tag{2.16}$$

where $\mathbf{c}_j = [c_{j0}, c_{j1}, ..., c_{jp}]^T$. Also, $(2k+1)(-1)^k[1, 1, ..., 1]\mathbf{c}_{j-1} = D_n \mathbf{c}_{j-1}$. Thus, (2.5) can be written as

$$\begin{bmatrix} \dot{c}_{j0} \\ \dot{c}_{j1} \\ \vdots \\ \dot{c}_{jp} \end{bmatrix} = \frac{\alpha}{\Delta x}\left(D_n \begin{bmatrix} c_{j-1,0} \\ c_{j-1,1} \\ \vdots \\ c_{j-1,p} \end{bmatrix} + A_n \begin{bmatrix} c_{j,0} \\ c_{j,1} \\ \vdots \\ c_{j,p} \end{bmatrix} \right). \tag{2.17}$$

Using the periodicity of boundary conditions $\mathbf{c}_0 = \mathbf{c}_N$, the above expression (2.17) yields

$$\dot{\mathbf{c}}_1 = \frac{\alpha}{\Delta x}\left(D_n \mathbf{c}_N + A_n \mathbf{c}_1 \right). \tag{2.18}$$

A block matrix L with periodic boundary conditions is obtained by inserting A_n and D_n as block matrices of $n \times n$ dimension

$$L = \begin{bmatrix} A_n & 0 & 0 & ... & 0 & 0 & D_n \\ D_n & A_n & 0 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0 & D_n & A_n \end{bmatrix}. \tag{2.19}$$

Hence, (2.5) can be written as a system of ODEs

$$\dot{\mathbf{c}} = \frac{\alpha}{\Delta x} L \mathbf{c}. \tag{2.20}$$

2.2 The characteristic polynomial

To find the eigenvalues of matrix L, an expression is derived in this section where λ is an eigenvalue of matrix L that satisfies

$$\begin{bmatrix} A_n & 0 & 0 & ... & 0 & 0 & D_n \\ D_n & A_n & 0 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0 & D_n & A_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix}, \tag{2.21}$$

where $\mathbf{v}^T = [\mathbf{v}_1^T, \mathbf{v}_2^T, ..., \mathbf{v}_N^T]$ is the corresponding eigenvector of matrix L. Also, components of vector $\mathbf{v}^T$ are $\mathbf{v}_j$, $j = 1, 2, ..., N$, and are column vectors of length $n = p + 1$. Thus, (2.21) can be written in an alternate form as

$$D_n \mathbf{v}_{j-1} + A_n \mathbf{v}_j = \lambda \mathbf{v}_j, \quad j = 1, 2, ..., N, \tag{2.22}$$

where $\mathbf{v}_0 = \mathbf{v}_N$, using periodic boundary conditions. Also, combining $\mathbf{v}_j$ on right side and replacing the value of D_n in (2.22), the above expression can be written as

$$\mathbf{r}_n[1, 1, ..., 1]\mathbf{v}_{j-1} = (\lambda \boldsymbol{I} - A_n)\mathbf{v}_j. \tag{2.23}$$

We define $S_j = [1, 1, ..., 1]\mathbf{v}_j$, then (2.23) can be written as

$$S_{j-1}\mathbf{r}_n = (\lambda \boldsymbol{I} - A_n)\mathbf{v}_j. \tag{2.24}$$

Multiplying (2.24) by $[1, 1, ..., 1](\lambda \boldsymbol{I} - A_n)^{-1}$ on both sides, we get

$$S_j = S_{j-1}[1, 1, ..., 1](\lambda \boldsymbol{I} - A_n)^{-1}\mathbf{r}_n. \tag{2.25}$$

We introduce a new expression $f_n(\lambda)$ by

$$f_n(\lambda) = [1, 1, ..., 1](\lambda \boldsymbol{I} \ - A_n)^{-1}\mathbf{r}_n. \tag{2.26}$$

Hence, (2.25) yields a recursive formula

$$\begin{aligned} S_j &= f_n(\lambda) S_{j-1}, && (2.27)\\ &= f_n(\lambda)(f_n(\lambda) S_{j-2}), && (2.28)\\ &= f_n^2(\lambda) S_{j-2}, && (2.29)\\ &\vdots && (2.30)\\ S_N &= f_n^N(\lambda) S_1. && (2.31) \end{aligned}$$

Since the boundary conditions are periodic, we get $S_N = f_n^N(\lambda) S_N$. Thus, we obtain

$$f_n^N(\lambda) = 1. \tag{2.32}$$

Thus, the eigenvalues of matrix L are the roots of the equations

$$f_n(\lambda) = \omega_j, \quad \omega_j = e^{\frac{2\pi i}{N} j}, \quad j = 0, 1, 2, ..., N-1. \tag{2.33}$$

2.3 Padé approximant

We can derive an analytical expression for $f_n(\lambda)$ using Padé approximants which are approximants using rationals.

Polynomials have elegant approximation properties to approximate functions. However they do not go along asymptotes and may produce oscillations. Rational functions are more diverse and applicable as they are ratios of polynomials. Rational functions might be bounded, possess singularities, and be free of oscillations. The Padé approximant gives us a better function approximation than its Taylor's series expansion.

Definition [1]: Assume that the Taylor series expansion of $f(z)$ is given by

$$f(z) = \sum_{i=0}^{\infty} c_i z^i.$$

Then, a Padé approximant of $f(z)$ is a rational function $R_{[\frac{L}{M}]}(z)$

$$R_{[\frac{L}{M}]}(z) = \frac{P_L(z)}{Q_M(z)} = \frac{a_0 + a_1 z + ... + a_L z^L}{b_0 + b_1 z + ... + b_M z^M}, \tag{2.34}$$

where $P_L(z)$ and $Q_M(z)$ are polynomials in z of degree at most L and M respectively, and the Padé approximant satisfies

$$\sum_{i=0}^{\infty} c_i z^i = \frac{a_0 + a_1 z + ... + a_L z^L}{b_0 + b_1 z + ... + b_M z^M} + O(z^{L+M+1}). \tag{2.35}$$

It is customary to choose $Q(0) = 1$, so that the Padé approximant is unique. The coefficients $a_0, a_1, ..., a_L$ and $b_0, b_1, ..., b_M$ can be computed from $c_0, c_1, ...,$ when a_0 is fixed. The approximants can be presented in a table called the Padé table. The Padé approximatnts of e^z for $p, q \in Z^+$ are given by the formula as in [4],

$$[p/q]_{exp(z)} = \frac{{}_1F_1(-p, -p-q, z)}{{}_1F_1(-q, -p-q, -z)}, \tag{2.36}$$

where ${}_1F_1$ is the confluent hyper-geometric function defined by the series [1],

$$ {}_1F_1(a,b,z) = 1 + \frac{a}{b}z + \frac{a}{b}\frac{a+1}{b+1}\frac{z^2}{2!} + \frac{a}{b}\frac{a+1}{b+1}\frac{a+2}{b+2}\frac{z^3}{3!} + ... \tag{2.37} $$

Also, ${}_1F_1(a,b,z)$ is a finite sum which is a polynomial of degree $|a|$, if a, b are negative integers and $b \leq a$. The main result is stated in the following theorem, see [26].

Theorem 1. [26] If $\mathbf{A_n}$ is an $n \times n$ matrix given by (2.14), and $f_n(z) = (1, ..., 1)(z\boldsymbol{I} - \mathbf{A}_n)^{-1}\mathbf{r}_n$, where $\mathbf{r}_n = [1, -3, ..., (-1)^{n-1}(2n-1)]^T$, then

$$ f_n(z) = \frac{{}_1F_1(-n+1, -2n+1, -z)}{{}_1F_1(-n, -2n+1, z)}, \tag{2.38} $$

which is the $[n-1/n]$ Padé approximant of e^{-z}.

2.4 Spectrum of the DG method

The spectral values of the discontinuous Galerkin spatial discretization matrix L are given by (2.33) and can be computed using MATLAB software. We have presented the eigenvalues of matrix L for $p = 1, 2, ..., 6$ with $N = 20$ in Figure 2.1.

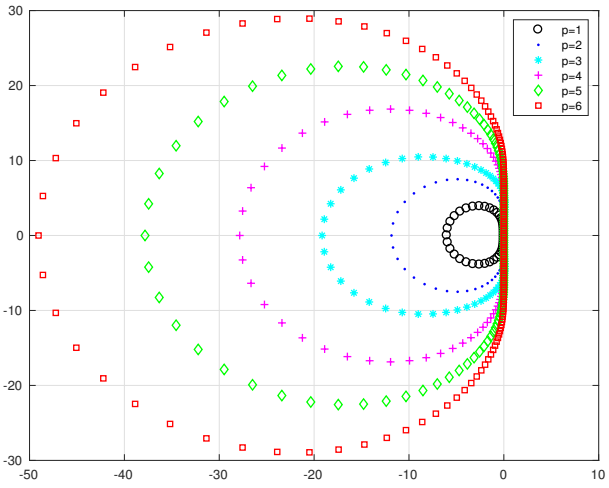

Figure 2.1: Eigenvalues of L for $p = 1, 2, ..., 6$ with $N = 20$.

It is apparent from Figure 2.1 that the size of spectrum increases with p. On the other hand, the curves produced by (2.33) are approaching and straighten near the imaginary axis as p grows. The leftmost eigenvalues are responsible for the decrease in CFL number with increasing value of p in Figure 2.1. The compression of spectral curves near the imaginary axis forces the absolute stability region of time integration scheme to posses a large enough part of the imaginary axis. The eigenvalues of matrix L can be utilized to measure the interval of absolute stability region. Real eigenvalues of matrix L for $p = 1, 2, ..., 6$ are presented in Table 2.1.

p	1	2	3
z	-6	-11.8424	-19.1569
p	4	5	6
z	-27.8419	-37.8247	-49.0518

Table 2.1: Real eigenvalues of L for $p = 1, 2, ..., 6$.

Chapter 3

Integration in time of semi-discrete schemes

Runge-Kutta methods are one of the most popular ODE solvers for solution of initial value problems. These methods were proposed by renowned mathematicians Carle Runge and Martin Kutta about 1900. Some advantages of RK methods are: easy to implement as they do not need derivatives of right side of a differential equation and easy to change the step-size adaptively.

3.1 Introduction to Runge-Kutta methods

Let us consider the initial value problem

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}), \quad t > t_0, \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}_0 \in \mathbb{R}^m, \tag{3.1}$$

where $\mathbf{y} = (y_1, y_2, ..., y_m)^T$ is a vector and f is a continuous differentiable function. To move forward in time, the step-size h is defined as $h = h_n = t_{n+1} - t_n$. An s-stage Runge-Kutta method for the system of ODEs (3.1) is defined as

$$\mathbf{Y}_i = \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_j h, \mathbf{Y}_j), \quad i = 1, 2, ..., s, \tag{3.2}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i f(t_n + c_i h, \mathbf{Y}_i), \tag{3.3}$$

where $\mathbf{Y}_i$'s are intermediate approximations to the solution at times $t_n + c_i h$ referred to as stages. Runge-Kutta method can be expressed using a short and well-known notation as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{3.4}$$

where $\mathbf{k}_i$'s are called nodes and defined by

$$\mathbf{k}_i = f(t_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j), \quad i = 1, 2, ..., s. \tag{3.5}$$

For example, one of the most common members of the Runge-Kutta family known as RK4 can be expressed as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \tag{3.6}$$

where

$$\begin{aligned}
\mathbf{k}_1 &= f(t_n, \mathbf{y}_n), \\
\mathbf{k}_2 &= f(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1), \\
\mathbf{k}_3 &= f(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2), \\
\mathbf{k}_4 &= f(t_n + h, \mathbf{y}_n + h\mathbf{k}_3).
\end{aligned}$$

The Runge-Kutta methods with the coefficients a_{ij}, b_i, and c_i $(i = 1, 2, ..., s)$ can be written in a compact form using a Butcher tableau [6] as

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

Table 3.1: Butcher tableau for a general s-stage Runge-Kutta method.

Also, the coefficients of an s-stage Rune-Kutta method can be presented in matrix form and denoted by $C = [c_i]^T$, $A = [a_{ij}]$, and $B = [b_i]^T$ in a Butcher tableau.

3.1.1 Explicit Runge-Kutta methods

A special class of Runge-Kutta methods when the matrix $A = [a_{ij}]$ is strictly lower triangular: $a_{ij} = 0$, $j \geq i, j = 1, ..., s$, is called explicit Runge-Kutta methods.

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \ddots & & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array} .$$

Table 3.2: Butcher tableau for an explicit Runge-Kutta method.

We will only consider explicit Runge-Kutta methods here. A few more examples of explicit Runge-Kutta methods are presented.

3.1.2 One-parameter family of Runge-Kutta methods

Let us derive an explicit method with $s = 2$, i.e., a two-stages method. The Taylor's series expansion of (3.1) yields

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{y}' + \frac{1}{2}h^2\mathbf{y}'' + O(h^3). \tag{3.7}$$

With the notation $f(t_n, \mathbf{y}_n) = f_n$, (3.7) becomes

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hf_n + \frac{1}{2}h^2\left(\left(\frac{\partial f_n}{\partial t}\right) + \left(\frac{\partial f_n}{\partial \mathbf{y}}\right)f_n\right) + O(h^3), \tag{3.8}$$

where $\mathbf{y}'' = \frac{d}{dt}(f(t_n, \mathbf{y}_n)) = \frac{\partial f_n}{\partial t} + \frac{\partial f_n}{\partial \mathbf{y}}f_n$. When $s = 2$, (3.4) becomes

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hb_1\mathbf{k}_1 + hb_2\mathbf{k}_2. \tag{3.9}$$

Since $c_1 = 0$ for an explicit method and using (3.5), we obtain

$$\mathbf{k}_1 = f(t_n, \mathbf{y}_n) \quad = f_n, \tag{3.10}$$

$$\mathbf{k}_2 = f(t_n + hc_2, \mathbf{y}_n + a_{21}\mathbf{k}_1). \tag{3.11}$$

The Taylor series expansion of $\mathbf{k}_2$ with respect to t gives

$$\begin{aligned}\mathbf{k}_2 &= f(t_n, \mathbf{y}_n + ha_{21}\mathbf{k}_1) + hc_2\frac{\partial}{\partial t}f(t_n, \mathbf{y}_n + ha_{21}\mathbf{k}_1) + O(h^2)\\ &= f_n + hc_2(\frac{\partial f_n}{\partial t}) + ha_{21}(\frac{\partial f_n}{\partial \mathbf{y}})f_n + O(h^2).\end{aligned} \tag{3.12}$$

Thus, (3.9) becomes

$$\mathbf{y}_{n+1} = \mathbf{y}_n + hb_1 f_n + hb_2\left(f_n + hc_2\left(\frac{\partial f_n}{\partial t}\right) + ha_{21}\left(\frac{\partial f_n}{\partial \mathbf{y}}\right)f_n\right) + O(h^3). \tag{3.13}$$

We can now equate the coefficients in (3.8) and (3.13) to get the values of unknowns b_1, b_2, a_{21} and c_2

$$[hf_n]: \quad b_1 + b_2 = 1, \tag{3.14}$$

$$[h^2\left(\frac{\partial f_n}{\partial t}\right)]: \quad b_2c_2 = \frac{1}{2}, \tag{3.15}$$

$$[h^2\left(\frac{\partial f_n}{\partial \mathbf{y}}\right)f_n]: \quad b_2a_{21} = \frac{1}{2}. \tag{3.16}$$

This is a system with three equations in four unknowns. We can solve this underdetermined system in terms of b_2 to present a one parameter family of explicit two-stages Runge-Kutta method as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h[(1 - b_2)\mathbf{k}_1 + b_2\mathbf{k}_2], \tag{3.17}$$

$$\mathbf{k}_1 = f(t_n, \mathbf{y}_n), \tag{3.18}$$

$$\mathbf{k}_2 = f(t_n + \frac{h}{2b_2}, \mathbf{y}_n + \frac{h}{2b_2}\mathbf{k}_1). \tag{3.19}$$

Well-known two-stages methods are obtained, when $b_2 = \frac{1}{2\alpha}$, where α is a parameter, see [33]. For example, we get, Heun's method with $\alpha = 1$, and Ralston's method with $\alpha = 2/3$. With $\alpha = 0$, (3.17) - (3.19) becomes the first-order Euler's method.

0	0	0
α	α	0
	$1 - \frac{1}{2\alpha}$	$\frac{1}{2\alpha}$

Table 3.3: One-parameter family of two-stages explicit Runge-Kutta methods.

A one-parameter family of three-stages Runge-Kutta methods is given by Butcher tableau as

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

Table 3.4: A one parameter family of three-stages Runge-Kutta methods.

3.2 Consistency

Any explicit method can be written as one-step method using (3.3),

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\psi(t_n, \mathbf{y}_n, h), \tag{3.20}$$

where ψ satisfies a Lipschitz condition in $\mathbf{y}$. The original RK scheme (3.3) can be obtained by replacing f by ψ. For a given Runge-Kutta method, the order of consistency can be found as taking the the Taylor series expansion of the method and subtracting the Taylor expansion from the exact solution. The order of consistency is defined by the first power of h with a non-zero coefficient such that the derivatives exist and are bounded.

Definition 1. [25] For each $\mathbf{y} \in \mathbb{R}^m, t > 0$, denoted by $\eta = \eta(\xi)$ the unique solution to the initial value problem

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

with initial condition $(\mathbf{y}, t)$, then

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

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

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

uniformly for all $\mathbf{y} \in \mathbb{R}^m, t > 0$, and it is called consistency of order p if

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

for all $\mathbf{y} \in \mathbb{R}^m$, $t > 0$, $\forall\ h > 0$, and some constant K.

3.3 Convergence and consistency

The necessary conditions on the coefficients of the Runge-Kutta method to be consistent of order p are given in [11].
Let A be the $s \times s$ matrix whose entries are the coefficients of the Butcher tableau that defines the Runge-Kutta method. Let $\mathbf{Y} = (Y_1, Y_2, ..., Y_{\mathbf{s}})^T$ be the matrix of intermediate stages, $\mathbf{b} = (b_1, b_2, ..., b_{\mathbf{s}})^T$ be the vector of weights, C =diag$(c_1, c_2, ..., c_s)$ be the diagonal matrix with the c_j coefficients on its diagonal, and $\mathbf{1} = (1, 1, ..., 1)^T$. For scheme (3.3) to be consistent of order p, the following condition must be satisfied

$$\mathbf{b}^T A^k C^{l-1} \mathbf{1} = \frac{(l-1)!}{(l+k)!}, \quad 1 \leq l + k \leq p, \tag{3.21}$$

for each $l, 1 \leq l \leq p$, and the order condition for $k = 1, 2, ..., p - l$. Setting $l = 1$ and $k \leftarrow k + 1$ in (3.21), we get

$$\mathbf{b}^T A^{k-1} \mathbf{1} = \frac{1}{k!}, \quad k = 1, 2, ..., p. \tag{3.22}$$

It can be seen in [6] that the order conditions are used to prove that if an explicit s-stage Runge-Kutta method has order p, then $\mathbf{s} \geq p$. It is also proven that if an explicit s-stage Runge-Kutta method has order $p \geq 5$, then $s > p$. This implies that the only explicit Runge-Kutta methods with $\mathbf{s} = p$ have order $p = 1, 2, 3, 4$.
The highest possible order of accuracy for a method with a given number of stages [3], is given below in the table

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 3.5: Highest attainable order per number of stages.

The necessary and sufficient condition for consistency and convergence is stated in theorem, see [25].

Theorem 1. [25] Assume that the function ψ describing the single-step (Runge-Kutta) method is continuous in all variables and satisfies the Lipschitz condition in the first variable; i.e.,

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

for all $\mathbf{y}, \mathbf{w} \in \mathbb{R}^m$, $t > 0$, 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.

3.4 Local error of Runge-Kutta methods

The error made by a numerical method in one time-step is called the local error. It is important to have an estimate of the local error of solutions at each time-step for the accuracy and stability of a Runge-Kutta method. The local error is defined as the difference between the numerical solution $\mathbf{y}_{n+1}$ and the exact solution $\tilde{\mathbf{y}}_{n+1}$ of the initial value problem at time t_{n+1}

$$T_{n+1} = ||\tilde{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}||. \tag{3.23}$$

In [15], an accurate local error bound for a Runge-Kutta method of order p is stated. Also, if a method is order p, and $f \in C^p$, then

$$||T_{n+1}|| = ||\tilde{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}|| \leq Ch^{p+1}, \tag{3.24}$$

for some constant C.

3.5 Global error

The numerical solution must converge to the exact solution when a numerical method is applied to an ODE system. The error in the approximated solution made by a numerical method after final time-step is known as the global error and defined as

$$\mathbf{e}_n = \tilde{\mathbf{y}}_n - \mathbf{y}(t_n), \tag{3.25}$$

where $\mathbf{y}(t_n)$ is the approximate solution at final time-step. Since the the order of local error is $O(h)^{p+1}$, so the global error is $O(h)^p$.
The following theorem from [15] states to bound the global error at final time t_f based on the local error approximation.

Theorem 2. [15] Let U be the neighborhood of $\{(\tilde{\mathbf{y}}(t), t) : t_0 \leq t \leq t_f\}$ where $\tilde{\mathbf{y}}(t)$ is the exact solution of (3.1). Suppose that in U

$$||\frac{\partial f}{\partial y}|| \leq M, \tag{3.26}$$

and that the local error estimates $||T_n|| \leq Ch_{n-1}^{p+1}$ are valid in U. Then the global error (3.25) can be estimated by

$$||\mathbf{e}_n|| \leq h^p \frac{C'}{M}(exp(M(t_f - t_0)) - 1), \tag{3.27}$$

where $h = \max h_i$,

$$C' = \begin{cases} C & M \geq 0 \\ Cexp(-Mh) & M < 0, \end{cases}$$

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

3.6 Absolute stability regions

Consider the scalar test equation to investigate the regions of absolute stability for explicit Runge-Kutta methods

$$y' = \lambda y, \tag{3.28}$$

where λ is a complex constant. Using initial condition $y(0) = c$, where $c > 0$, then the exact solution of (3.28) is

$$y(t_n) = ce^{\lambda t_n}. \tag{3.29}$$

The absolute value of solution will grow exponentially in time i.e., $|y(t_{n+1})| > |y(t_n)|$, when $Re(\lambda) > 0$. Thus, the given problem is unstable. Meanwhile, the solution will oscillate for all times, if $Re(\lambda) = 0$. It implies that the difference between the solution curves will remain the same. Also, when $Re(\lambda) < 0$, the absolute value of the solution $y(t)$ will decay exponentially in time that results in an absolute stability condition

$$|y(t_{n+1})| < |y(t_n)|, \quad n = 1, 2, ... \tag{3.30}$$

The region of the complex z-plane that provides an area $z = \lambda \Delta t$ for a numerical solution to satisfy (3.30) is known as the region of absolute stability. We rewrite the inner stages of Runge-Kutta method (3.2) to find the absolute stability region for test equation (3.28)

$$\begin{aligned} \mathbf{Y} =& \mathbf{y}_n \mathbf{1} + h\lambda A \mathbf{Y}, \\ \mathbf{Y} =& \mathbf{y}_n (\boldsymbol{I} - zA)^{-1} \mathbf{1}. \end{aligned} \tag{3.31}$$

Rewriting (3.3) as

$$\begin{aligned} y_{n+1} =& y_n + h\lambda \mathbf{b}^T \mathbf{Y}, \\ =& y_n + y_n z \mathbf{b}^T (\boldsymbol{I} - zA)^{-1} \mathbf{1}, \\ =& y_n (1 + z \mathbf{b}^T (\boldsymbol{I} - zA)^{-1} \mathbf{1}), \end{aligned} \tag{3.32}$$

Thus, we have

$$y_{n+1} = y_n (1 + z \mathbf{b}^T (\boldsymbol{I} + \sum_{i=1}^{\infty} z^i A^i) \mathbf{1}), \tag{3.33}$$

where $\mathbf{b}^T = [b_j], j = 1, ..., s$, is the matrix defined in Butcher tableau 3.1. Replacing (3.22) in (3.33), and for explicit Runge-Kutta method $A^j = 0$, for all $j \geq s$, we get

$$y_{n+1} = \left(1 + z + \frac{z^2}{2} + ... + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \mathbf{b}^T A^{j-1} \mathbf{1}\right) y_n. \tag{3.34}$$

For $|y_{n+1}| \leq |y_n|$, we have the following stability requirement

$$|1 + z + \frac{z^2}{2} + ... + \frac{z^p}{p!} + \sum_{j=p+1}^{s} z^j \mathbf{b}^T A^{j-1} \mathbf{1}| \leq 1, \tag{3.35}$$

which is a polynomial in the complex plane z. The Runge-Kutta method will be stable when $z = \lambda \Delta t$ lies in the absolute stability region of complex plane z. Next, we present the MATLAB algorithm to generate the absolute stability regions of classical RK schemes.

Algorithm 1 Absolute stability region

$X_0 = -5$ % x range and number of points
$X_1 = 5$
$X_N = 101$
$Y_0 = -5$ % y range and number of points
$Y_1 = 5$
$Y_N = 101$
$x_n = linspace(X_0, X_1, X_N)$ % construct mesh
$y_n = linspace(Y_0, Y_1, Y_N)$
$[x, y] = meshgrid(x_n, y_n)$
$z = x + i * y$ % compute z
$Euler = abs(1 + z)$ % Euler's amplification factor
$RK2 = abs(1 + z + 1/2 * z^2)$ % RK2 amplification factor
$RK3 = abs(1 + z + 1/2 * z^2 + 1/6 * z^3)$ % RK3 amplification factor
$RK4 = abs(1 + z + 0.5 * z.^2 + 1/6 * z.^3 + 1/24 * z.^4)$ % RK4 amplification factor
$contour(x, y, Euler, [1, 1], 'r-')$ % plot contours
hold on
$contour(x, y, RK2, [1, 1], 'k-')$ % plot contours
$contour(x, y, RK3, [1, 1], 'c-')$ % plot contours
$contour(x, y, RK4, [1, 1], 'b-')$ % plot contours

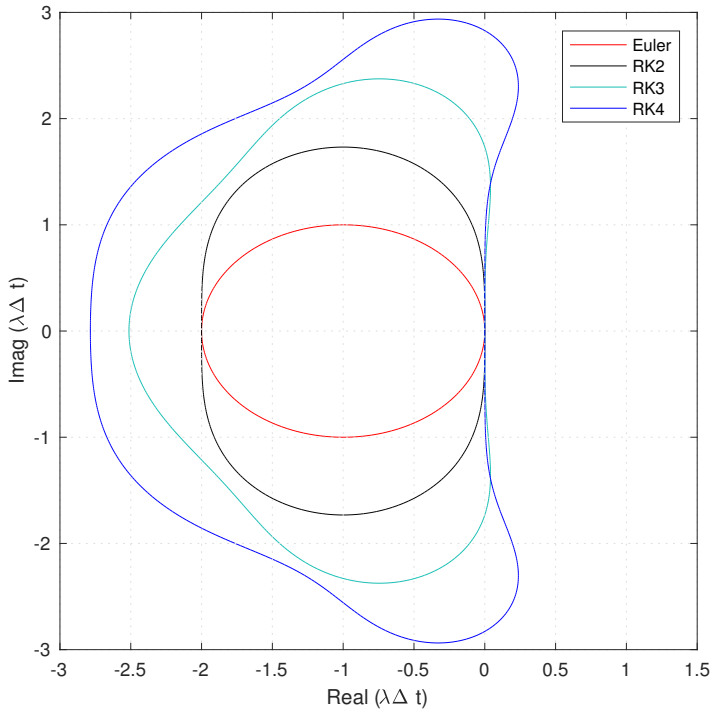

Figure 3.1: Boundaries of absolute stability regions of classical explicit RK methods.

Name	*Stages*	Interval of absolute stability
Euler's method	1	$(-2, 0)$
RK2	2	$(-2, 0)$
RK3	3	$(-2.51, 0)$
RK4	4	$(-2.78, 0)$

Table 3.6: Summary of classical RK schemes.

Note that Runge-Kutta methods with $s = p$ have the same regions of absolute stability. When $s \geq p$, the absolute stability region is constructed by the coefficients of a Runge-Kutta method using (3.34). Wider stability regions can be constructed by adding more stages into a Runge-Kutta method and using the coefficients of Butcher tableau.

3.7 Stability of systems of ODEs

Consider a system of ordinary differential equations

$$\frac{d\mathbf{y}}{dt} = A\mathbf{y}, \tag{3.36}$$

where A is a constant, square and diagonalizable $m \times m$ matrix. We can write the general solution as

$$\mathbf{y}(t) = \sum_{i=1}^{m} C_i e^{\lambda_i t} \mathbf{v}_i,$$

where $\lambda_1, \lambda_2, ..., \lambda_m$ are the eigenvalues, $\mathbf{v}_i = v_1, v_2, ..., v_m$ are the corresponding eigenvectors, and $C_1, C_2, ..., C_m$ are coefficients. The stability of system (3.36) is determined by the eigenvalues of matrix A. The origin is said to be stable, if all the eigenvalues of A lie in the left half plane. Also, the origin is asymptotically stable when all the eigenvalues of matrix A have negative real part.

Let $\Lambda = \text{diag}\{\lambda_1, \lambda_2, ..., \lambda_m\}$, be the diagonal matrix consisting of eigenvalues of A. If P be the matrix of eigenvectors of A such that

$$P^{-1}AP = \Lambda, \tag{3.37}$$

where P is a non-singular matrix. Let us assume the change of dependent variables, $\mathbf{w} = P^{-1}\mathbf{y}$ and multiply (3.36) by P^{-1}, we get $P\mathbf{w} = \mathbf{y}$. Thus, a system of differential equations is obtained

$$\frac{d\mathbf{w}}{dt} = \Lambda\mathbf{w}. \tag{3.38}$$

The system (3.38) can be written as m independent scalar equations. Each $\lambda_i \Delta t, \quad i = 1, 2, ..., m$ must lie within the absolute stability region to establish the stability of Runge-Kutta methods. Thus, the absolute stability condition becomes

$$|\mathbf{w}_n| \leq |\mathbf{w}_{n-1}| \leq ... \leq |\mathbf{w}_0|. \tag{3.39}$$

Also, in variable $\mathbf{y}$ (3.39) yields

$$|\mathbf{y}_n| \leq \|P\||\mathbf{w}_n| \leq ... \leq \|P\||\mathbf{w}_0| \leq \|P\|\|P^{-1}\||\mathbf{y}_0|, \tag{3.40}$$

where $\|.\|$ is the specific matrix norm. The constant $\|P\|\|P^{-1}\|$ is denoted by

$$cond(P) = \|P\|\|P^{-1}\|, \tag{3.41}$$

and it is called the condition number of the eigenvector of matrix P. Hence the stability bound is obtained as

$$|\mathbf{y}_n| \leq cond(P)|c|, \quad n = 1, 2, ..., \quad c > 0. \tag{3.42}$$

In general, there is no restriction on the size of the stability constant, $cond(P)$. It may depend on the size m of the ODE system and is independent of n. The eigenvalues of matrix A can be used to estimate a stable time-step if the $cond(P)$ is not large.

Chapter 4

Efficient explicit Runge-Kutta schemes for the discontinuous Galerkin method

After applying the DG method to (2.1), we obtain a semi-discrete scheme, which is a system of ordinary differential equations (2.20). This ODE system requires a higher-order numerical ODE solver for the time integration, where accuracy, computational cost and memory storage are mandatory performance indicators. In this chapter, it is observed that the classical RK schemes have higher computational cost and memory requirements than 2N-storage schemes.

Efficient explicit low-storage Runge-Kutta time integration schemes associated with the discontinuous Galerkin method are discussed in detail in this chapter. Low-storage schemes are also known as 2N-storage schemes, where N is the dimension of the given system. Relevant 2N-storage high-order Runge-Kutta methods up to 8 stages reported in literature are reviewed. The CFL numbers with these methods are discussed in this chapter. Numerical results are presented for solution of (2.20) with number of solvers and comparing their efficiency.

4.1 Time discretization

Consider a system of ODEs (2.20) resulting from the DG method applied to one-dimensional advection equation (2.1)

$$\frac{d\tilde{\mathbf{c}}}{dt} = \frac{\alpha}{\Delta x} L\tilde{\mathbf{c}}, \tag{4.1}$$

where the degrees of freedoms, vector $\mathbf{c}$ is renamed $\tilde{\mathbf{c}}$ so it is not confused with the coefficients of Butcher tableau. High-order explicit 2N-storage Rung-Kutta schemes can be used to solve (4.1). Williamson and Fyfe [40], and Carpenter [9] presented the idea of 2N-storage Rung-Kutta methods. This 2N-storage technique allows us to store the useful information in two storage registers in computer. An s-stage Runge-Kutta method in 2N-storage format can be as

$$\begin{aligned} d\tilde{\mathbf{c}}^{(i)} &= A_i d\tilde{\mathbf{c}}^{(i-1)} + \Delta t L(t_n + c_i \Delta t, \tilde{\mathbf{c}}^{(i-1)}), \\ \tilde{\mathbf{c}}^{(i)} &= \tilde{\mathbf{c}}^{(i-1)} + B_i d\tilde{\mathbf{c}}^{(i)}, \quad i = 1, ..., s. \end{aligned}$$

where A_i, B_i and c_i are coefficients of a 2N-storage scheme. Note, $A_1 = 0$ for an explicit scheme. The values of vectors $d\tilde{\mathbf{c}}$ and $\tilde{\mathbf{c}}$ must be stored in two registers to form the 2N-storage algorithm. Using Williamson's technique [40], the algorithm takes the form

Algorithm 2 2N-storage algorithm

$K_2 = 0$,
$K_1 = \tilde{\mathbf{c}}^n$,
for $i = 1 : s$ do
$K_2 = A_i K_2 + \Delta t L(t + \Delta t c_i, K_1)$,
$K_1 = K_1 + B_i K_2$,
end
$\tilde{\mathbf{c}}^{n+1} = K_1$.

4.2 Relation to Butcher tableau

The coefficients of classical RK schemes a_{ij}, b_i and c_i, and 2N-storage coefficients A_i and B_i are related to each other as described in [40] by

$$B_i = a_{i+1,i}, \quad i = 1, 2, ..., s-1, \tag{4.2}$$

$$B_s = b_s, \tag{4.3}$$

$$A_i = \frac{b_{i-1} - B_{i-1}}{b_i}, \quad i = 2, ..., s, b_j \neq 0, \tag{4.4}$$

$$A_i = \frac{a_{i+1,i-1} - c_i}{B_i}, \quad i = 2, ..., s, b_j = 0. \tag{4.5}$$

In addition, the relation to classical RK coefficients and 2N-storage coefficients is presented using Butcher tableau in Appendix A. A brief introduction of these 2N-storage schemes has been presented next.

4.2.1 LDD46 scheme [21]

An explicit six-stages, fourth order Runge-Kutta scheme LDD46, has been proposed by M. Calvo, J.M. Franco and L. Rández in [21]. This 2N-storage scheme has been constructed to minimize the dissipation and dispersion errors when solving wave propagation problems while maintaining large stability regions. The coefficients a_{ij} are expressed in the form of a lower triangular matrix $\mathbf{A} \in \mathbb{R}^{s \times s}$ and b_i as a vector $\mathbf{b} \in \mathbb{R}^s$.

$$\mathbf{A} = \begin{bmatrix} 0 & & & & \\ b_1 + \gamma_1 & 0 & & & \\ b_1 & b_2 + \gamma_2 & 0 & & \\ \vdots & \vdots & \ddots & \ddots & \\ b_1 & b_2 & ... & b_{s-1} + \gamma_{s-1} & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{s-1} \\ b_s \end{bmatrix}, \tag{4.6}$$

Coefficients of LDD46 scheme			
i	c_i	b_i	γ_i
1	0	0.10893125722541	0.17985400977138
2	0.28878526699679	0.13201701492152	0.14081893152111
3	0.38176720366804	0.38911623225517	0.08255631629428
4	0.71262082069639	-0.59203884581148	0.65804425034331
5	0.69606990893393	0.47385028714844	0.31862993413251
6	0.8305050587987157	0.48812405426094	

Table 4.1: LDD46 coefficients.

4.2.2 Carpenter's(5,4) scheme [9]

A five-stages fourth-order explicit Runge-Kutta method has been presented in [9] by Carpenter. Carpenter (5,4) scheme is more efficient, accurate and has a large allowable absolute stability domain than classical RK schemes. The coefficients of Carpenter(5,4) are reported in Table 4.2.

Four sets of coefficients for Carpenter(5,4) scheme				
Coeff	Solution 1	Solution 2	Solution 3	Solution 4
A1	0	0	0	0
A2	-0.4812317431372	-0.4801594388478	-0.4178904745	-0.7274361725534
A3	-1.049562606709	-1.4042471952	-1.192151694643	-1.906288083353
A4	-1.602529574275	-2.016477077503	-1.697784692471	-1.444507585809
A5	-1.778267193916	-1.056444269767	-1.514183444257	-1.365489400418
B1	9.7618354692056e-02	0.1028639988105	0.1496590219993	4.1717869324523e-02
B2	0.4122532929155	0.7408540575767	0.3792103129999	1.232835518522
B3	0.4402169639311	0.7426530946684	0.8229550293869	0.5242444514624
B4	1.426311463224	0.4694937902358	0.6994504559488	0.72129132239696
B5	0.1978760537318	0.1881733382888	0.1530572479681	0.2570977031703
c1	0	0	0	0
c2	9.7618354692056e-02	0.1028639988105	0.1496590219993	4.1717869324523e-02
c3	0.3114822768438	0.487989987833	0.3704009573644	0.377744236865
c4	0.5120100121666	0.6885177231562	0.6222557631345	0.6295990426348
c5	0.8971360011895	0.9023816453077	0.9582821306748	0.8503409780005

Table 4.2: Carpenter's (5,4) method.

4.2.3 HALE7 Schemes [2]

A high-accuracy large-step explicit Runge-Kutta scheme is abbreviated as HALE-RK. It is a six/seven stages scheme given in the 2N-storage format while maintaining forth-order accuracy and maximizing the absolute stability region. The coefficients for HALE7 scheme are given in Table 4.3.

Coefficients of HALE7 scheme			
i	A_i	B_i	c_i
1	0	0.117322146869	0
2	-0.647900745934	0.503270262127	0.117322146869
3	-2.704760863204	0.233663281658	0.294523230758
4	-0.460080550118	0.283419634625	0.305658622131
5	-0.500581787785	0.540367414023	0.582864148403
6	-1.906532255913	0.371499414620	0.858664273599
7	-1.45	0.136670099385	0.868664273599

Table 4.3: HALE7 coefficients.

4.2.4 ORK25-6 [5]

To optimize performance of Runge-Kutta schemes for solution of a system of ODEs, S. Pirozzoli and M. Bernardini have developed this method. It is a five-stage, 2nd-order 2N-storage scheme to solve hyperbolic problems having a large absolute stability region. The coefficients of ORK25-6 are presented in Table 4.4.

Coefficients of ORK25-6 scheme enforcing $c_i \in [0, 1]$.			
i	A_i	B_i	c_i
1	0	0.2	0
2	-1.0	0.83204	0.2
3	-1.55798	0.6	0.2
4	-1.0	0.35394	0.8
5	-0.45031	0.2	0.8

Table 4.4: ORK25-6 coefficients.

4.2.5 RKF84, RKC84 and RKC73 schemes

Toulorge and Desmet [38] have constructed three 2N-storage techniques: RKF84, RKC84 and RKC73 specifically to minimize the computational cost of the RKDG method for the solution of hyperbolic conservation laws. The coefficients of these 2N-storage schemes A_i, B_i and c_i are tabulated in Tables 4.5, 4.6 and 4.7. The first number in 2N-storage methods with names RKF84, RKC84 and RKC73 indicates the number of stages and the second number indicates the rate of convergence.

Coefficients for RKF84 scheme			
stages	A_i	B_i	c_i
1	0	0.080379336882736950	0
2	-0.5534431294501569	0.5388497458569843	0.08037936882736950
3	-0.01065987570203490	0.01974974409031960	0.3210064250338430
4	-0.5515812888932000	0.09911841297339970	0.3408501826604660
5	-1.885790377558741	0.7466920411064123	0.3850364824285470
6	-5.701295742793264	1.679584245618894	0.5040052477534100
7	2.113903965664793	0.2433728067008188	0.6578977561168540
8	-0.5339578826675280	0.1422730459001373	0.9484087623348481

Table 4.5: RKF84 coefficients.

Coefficients for RKC84 scheme			
stages	A_i	B_i	c_i
1	0	0.2165936736758085	0
2	-0.7212962482279240	0.1773950826411583	0.2165936736758085
3	-0.01077336571612980	0.01802538611623290	0.2660343487538170
4	-0.5162584698930970	0.08473476372541490	0.2840056122522720
5	-1.730100286632201	0.8129106974622483	0.3251266843788570
6	-5.200129304403076	1.903416030422760	0.4555149599187530
7	0.7837058945416420	0.1314841743399048	0.7713219317101170
8	-0.5445836094332190	0.2082583170674149	0.9199028964538660

Table 4.6: RKC84 coefficients.

Coefficients for RKC73 scheme			
stages	A_i	B_i	c_i
1	0	0.01197052673097840	0
2	-0.8083163874983830	0.8886897793820711	0.01197052673097840
3	-1.503407858773331	0.4578382089261419	0.1823177940361990
4	-1.053064525050744	0.5790045253338471	0.5082168062551849
5	-1.463149119280508	0.3160214638138484	0.6532031220148590
6	-0.6592881281087830	0.243525368264122	0.8534401385678250
7	-1.667891931891068	0.06771230959408840	0.9980466084623790

Table 4.7: RKC73 coefficients.

4.3 Absolute stability regions of 2N-storage RK schemes

When an RK scheme is applied to integrate the initial value problem, a complex amplification factor is generated as stated in (3.34). Then the absolute stability condition is obtained by (3.35) and the absolute stability region of an RK scheme is defined as the locus $S = \{z : |R(z)| \leq 1\}$. From [3], it can be seen that the absolute stability regions of 2N-storage RK schemes of order p with $s > 4$ can be constructed using (3.34), the coefficients of 2N-storage schemes and Algorithm 1. Also, the real value of absolute stability region of an RK scheme is used to define the interval of absolute stability region, where imaginary part of complex amplification factor is zero.

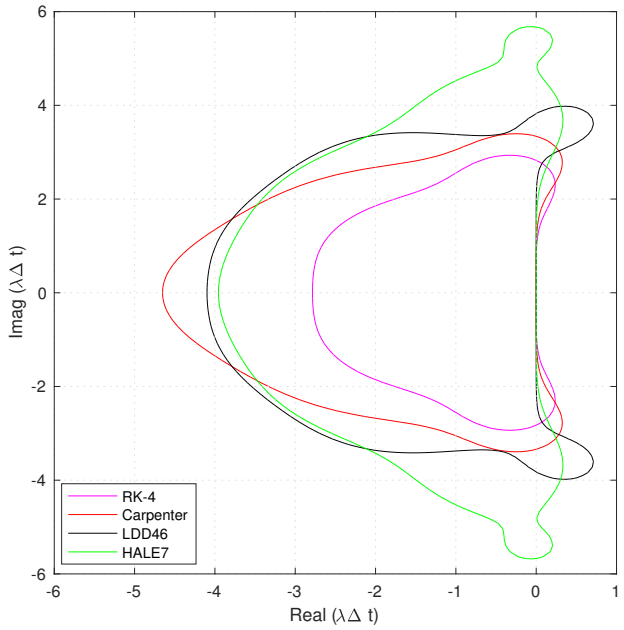

Figure 4.1: Absolute stability regions of 2N-storage schemes and classical RK4 scheme.

Name	*Order*	*Stages*	Interval of absolute stability
Carpenter(5,4)	4	5	$(-4.66, 0)$
LDD46	4	6	$(-4.1, 0)$
HALE7	4	7	$(-3.966, 0)$

Table 4.8: Summary of Crpenter(5,4), LDD46 and HALE7 schemes.

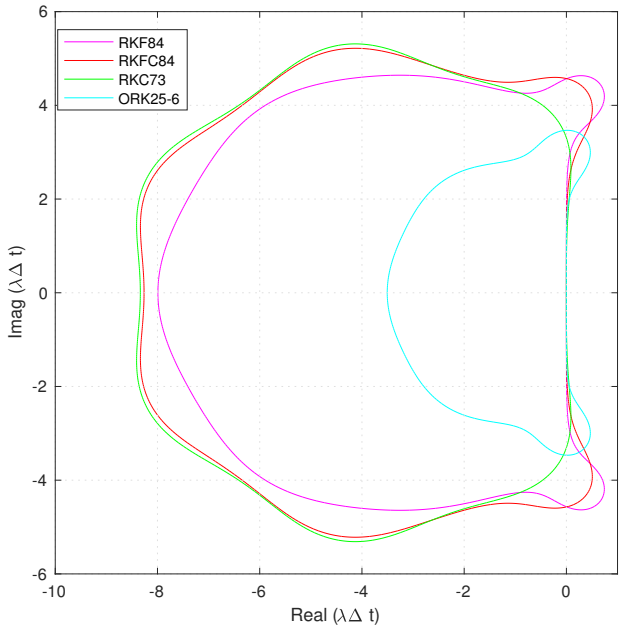

Figure 4.2: Absolute stability regions of RKC84, RKF84, RKC73 and ORK25-6 schemes.

Name	*Order*	*Stages*	Interval of absolute stability
ORK25-6	2	5	$(-3.5, 0)$
RKC73	3	7	$(-8.3, 0)$
RKC84	4	8	$(-8.2, 0)$
RKF84	4	8	$(-8.0, 0)$

Table 4.9: Summary of ORK25-6, RKC73, RKC84 and RKF84 schemes.

4.4 CFL condition for RK-DG pairings

For one-dimensional hyperbolic conservation laws, the stability condition (1.25) for fully discretized scheme involves the CFL number v. The CFL number v depends on the time integration technique and the spatial order of approximation p in the DG scheme. Some illustrations of RK-DG pairings are presented using Algorithm 1 and the spectral values of DG method with different orders of approximation p.

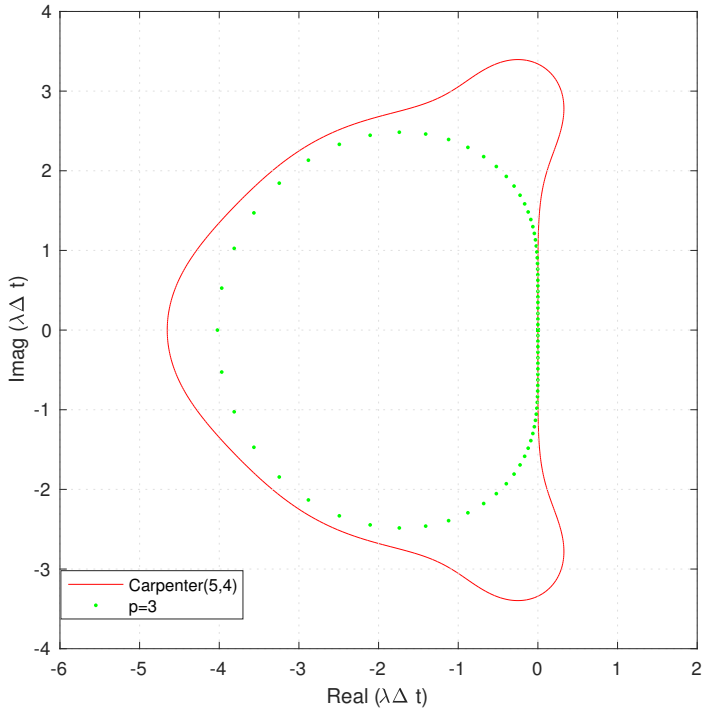

Figure 4.3: Absolute stability region of Carpenter(5,4) scheme and the spectrum of DG method with $p = 3$, scaled by a factor 0.22.

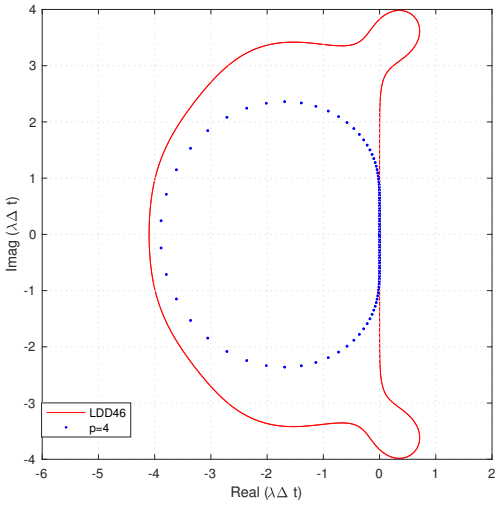

Figure 4.4: Absolute stability region of LDD46 scheme and the spectrum of DG method with $p = 4$, scaled by a factor 0.14.

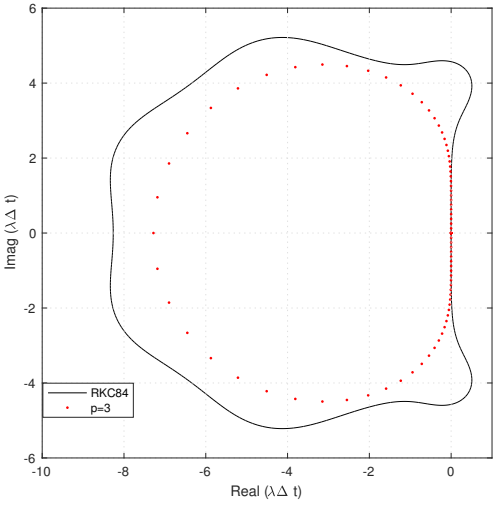

Figure 4.5: Absolute stability region of RKC84 scheme and the spectrum of DG method with $p = 3$, scaled by a factor of 0.38.

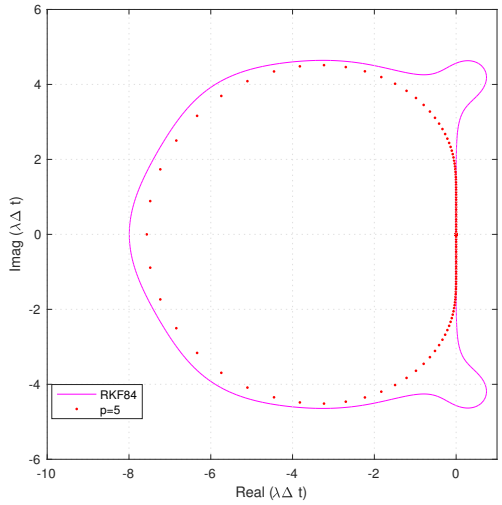

Figure 4.6: Absolute stability region of RKF84 scheme and the spectrum of DG method with $p = 5$, scaled by a factor 0.2.

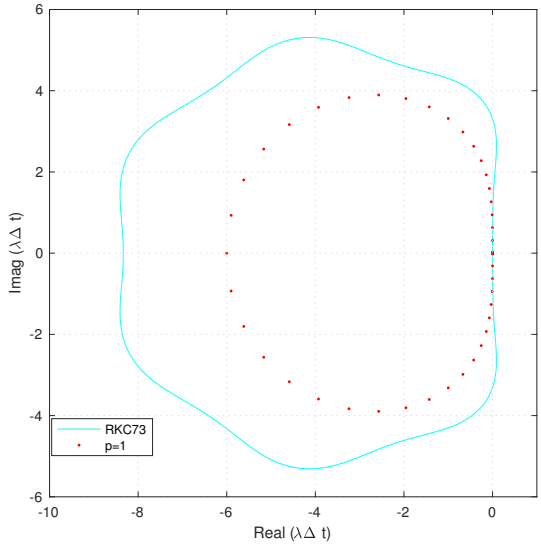

Figure 4.7: Absolute stability region of RKC73 scheme and the spectrum of DG method with $p = 1$.

The CFL numbers of RK-DG pairings for time integration schemes described in Tables 4.8 and 4.9 are presented in the Table 4.10 and 4.11. The CFL numbers of RK-DG pairings are calculated by taking the ratios of the largest real value of the absolute stability intervals of 2N-storage RK schemes and the eigenvalues of the DG method. When $p = 1$, CFL number for RKC73, RKF84 and RKC84 becomes 1, and stability condition results in $\Delta t = \frac{\Delta x}{\alpha}$ as described in Figure 4.7.

p	Carpenter(5,4)	LDD46	HALE7
1	0.67	0.67	0.65
2	0.34	0.34	0.33
3	0.22	0.21	0.20
4	0.15	0.14	0.14
5	0.11	0.10	0.104

Table 4.10: CFL numbers for Carpenter(5,4), LDD46 and HALE7 schemes with $p = 1, .., 5$.

p	ORK25-6	RKC73	RKF84	RKC84
1	0.58	1	1	1
2	0.29	0.62	0.61	0.64
3	0.18	0.41	0.38	0.4
4	0.125	0.29	0.26	0.29
5	0.09	0.22	0.20	0.21

Table 4.11: CFL numbers for ORK25-6, RKC73, RKF84 and RKC84 schemes with $p = 1, ..., 5$.

4.5 Order of approximation

The following result in [29] is stated for an priori error of discontinuous Galerkin method.

Proposition [29] Let u be the exact solution of (1.5a), u is smooth and U be the approximate solution of u using the discontinuous Galerkin method. Suppose that the basis functions consist of piecewise polynomials up to degree p. Then, the following L^2-norm error estimate holds

$$\|u - U\| \leq C(\Delta x)^{p+1}, \tag{4.7}$$

where Δx is the length of the element and C depends on u and its derivatives but is independent of Δx.

4.6 Cost measure

A cost measure can be used to compare and optimize RK schemes, see [32, 5]. To define a cost measure, assume that the computational cost is

1. proportional to the total number of elements, i.e. $\Omega/\Delta x$, where Ω is the size of the computational spatial domain.
2. proportional to the number of operations/nodes required by the spatial discretization scheme, say N_{op} ;
3. proportional to the number of Runge-Kutta stages, s;
4. proportional to the number of time steps, $n = T/\Delta t$.

Thus, we get

$$C = sN_{op}T\Omega\frac{1}{\Delta t \Delta x}. \tag{4.8}$$

4.7 Numerical results

In this section, the discontinuous Galerkin method is applied to one-dimensional hyperbolic conservation law (2.1). We compare the behavior of discontinuous Galerkin scheme paired with various 2N-storage RK schemes. Results for a linear advection equation are presented.

4.7.1 Solution of linear advection equation

A linear advection equation describes propagation of a variable quantity u along a wave moving with constant speed α. If we choose $\alpha = 1$, we get the following

$$u_t + u_x = 0, \quad x \in (0,1), t \geq 0, \tag{4.9}$$

$$u(x,0) = u_0(x). \tag{4.10}$$

In general, the exact solution of (4.9) is given by

$$u(x,t) = u_0(x - \alpha t). \tag{4.11}$$

We choose the initial condition for (4.9) to be a smooth periodic function

$$u_0(x,0) = sin(2\pi x), \quad x \in (0,1). \tag{4.12}$$

Hence, the exact solution is

$$u(x,t) = sin(2\pi(x-t)), \quad x \in [0,1], \quad t \geq 0. \tag{4.13}$$

The RKDG method is applied to (4.9) to produce an approximate solution. A uniform mesh is introduced consisting of N finite elements with a suitable time-step. The choice of space and time step may affect the stability of the numerical method. So the CFL condition for classical RK schemes (1.25) must be satisfied. The periodic boundary conditions, i.e., $u(0,t) = u(1,t)$ are used, and results are shown in the Table 4.12. The discrete L^2-norm is used to measure the error

$$||u - U||_{L^2} = \frac{\Delta x_j}{2}\left(\sum_j \int_{\Omega_j} |(u-U)|^2 d\xi\right)^{1/2}, \tag{4.14}$$

where Δx_j is the length of each element Ω_j, u is exact solution and U is approximated solution at each element Ω_j. Numerical quadrature (1.24) is used to solve the integral involved in (4.14). The rate of convergence is calculated by using the formula

$$r = log(\frac{error_{new}}{error_{old}}). \tag{4.15}$$

Table 4.12 shows the discrete L^2-norm error at $T = 4.0$ and the rates of convergence for the RKDG method with different order of approximations. The domain is divided into a sequence of meshes with $N = 10, 20, ..., 160$ elements. The rates of convergence are calculated using (4.15).

N	$RK2(p=1)$	*Rate*	$RK3(p=2)$	*Rate*	$RK4(p=3)$	*Rate*
10	$2.071e-01$		$2.934e-03$		$6.96e-05$	
20	$4.97e-02$	2.0606	$3.437e-04$	3.093	$4.379e-06$	3.990
40	$1.22e-02$	2.0218	$4.234e-05$	3.021	$2.745e-07$	3.995
80	$3.0e-03$	2.0073	$5.2800e-06$	3.003	$1.718e-08$	3.998
160	$8.0e-04$	2.0042	$6.600e-07$	3.000	$1.074e-09$	3.999

Table 4.12: L^2-norm error and rates of convergence of RK2, RK3 and RK4 methods, $T = 4$.

N	Carpenter(5,4)	Rate	LDD46	Rate	HALE7	Rate
10	$7.2568e-05$		$6.824e-05$		$6.8648e-05$	
20	$4.58e-06$	3.9858	$4.3072e-06$	3.9945	$4.31e-06$	3.994
40	$2.99e-07$	3.9988	$2.709e-07$	3.9979	$2.7e-07$	3.997
80	$2.0e-08$	3.9996	$1.69e-08$	3.999	$2.0e-08$	4.000
160	$1.0534e-09$	3.9990	$1.0527e-09$	4.000	$1.055e-09$	4.000

Table 4.13: L^2-norm error and rates of convergence of Carpenter(5,4), LDD46 and HALE7 with $p = 3$, at $T = 4$.

In Table 4.13, performance of 2N-storage schemes is presented when applied to (4.9). The error is calculated in L^2-norm and rates of convergence are computed. 2N-storage RK schemes have different number of stages $s = 5, 6, 7$, but they produce the same order of convergence as the classical 4th-order Runge-Kutta method. The order of spatial approximation is $p = 3$ in all 4th-order 2N-storage schemes and time-step Δt is computed using RK-DG pairings in Table 4.8. The Carpenter(5,4), LDD46 and HALE7 are efficient schemes for use with the DG method when comparing the rate of convergence and error to classical RK4 scheme.

A few more efficient 2N-storage schemes have been applied to (4.9). In Table 4.14, the performance of ORK25-6 and RKF84 schemes is reported. The L^2-norm error and numerical convergence rate is computed. It is observed that ORK25-6 with $p = 1$ is a 2nd-order accurate while RKF84 with $p = 3$ is a fourth-order scheme.

N	$ORK25-6(p=1)$	$Rate$	$RKF84(p=3)$	$Rate$
10	$2.0e-02$		$7.204e-05$	
20	$4.3e-03$	2.2090	$4.56e-06$	3.9816
40	$1.0e-03$	2.1364	$2.9e-07$	3.9928
80	$2.0e-04$	2.0553	$2.0e-08$	3.9958
160	$1.0e-04$	2.0244	$1.06e-09$	3.9980

Table 4.14: L^2-norm errors and rates of convergence of ORK25-6 with $p = 1$, and RKF84 with $p = 3$ at $T = 4$.

In Table 4.15, the similar results are presented for RK73 and RKC84 when applied to (4.9). The RKC73 with $p = 2$ is a 3rd-order and RKC84 with $p = 3$ is a 4th-order scheme.

N	$RKC73(p = 2)$	$Rate$	$RKC84(p = 3)$	$Rate$
10	$7.6e-03$		$9.871e-05$	
20	$9.2e-04$	3.040	$6.22e-06$	3.9885
40	$1.16e-05$	2.994	$3.7e-07$	4.0580
80	$1.44e-06$	3.001	$2.0e-08$	4.3201
160	$1.815e-07$	3.006	$1.08e-09$	3.9977

Table 4.15: L^2-norm errors and rates of convergence of RKC73 with $p = 2$, and RKC84 with $p = 3$ at $T = 4$.

4.8 Discussion

The criteria to investigate the performance of 2N-storage schemes paired with the DG method can be characterized by stability, accuracy and computational cost. The computational cost depends on number of parameters. The most important are the maximum time-step Δt and the CFL number. The larger time-step will result in small number of steps require to reach the final time T. The combination of spatial order of approximation p and 2N-storage schemes can be used to measure the effectiveness of RK-DG parings. Hence, certain tools are needed to investigate the aftermaths of this combination. It is also convenient to calculate the computation time of 2N-storage schemes.

In Table 4.16, results are reported when two classical RK schemes are applied to one-dimensional conservation law with smooth initial condition. For comparison, the number of elements in spatial domain $N = 100$ is kept fixed. The order of approximation p in DG method for RK2 and RK3 is $p = 1$. The order of approximation is kept same with two different classical RK schemes to illustrate the cost measure (4.8) here. The number of

stages s of $RK2$ and RK3 are $s = 2, 3$ respectively. The accuracy is measured by L^2-norm error. The cost is computed by the ratio of RK3 scheme runtime over RK2 computation time. It is shown that RK3 scheme has 3 stages and the cost is almost 1.5 compared with RK2 scheme using (4.8).

Schemes	Time Elapsed	Cost	L^2-Norm Error
RK2($p = 1$)	0.2564 Seconds	1	4.8e-02
RK3($p = 1$)	0.37562 Seconds	1.5	3.5822e-04

Table 4.16: Cost analysis of RK2 and RK3 with $p = 1$, $N = 100$, and $T = 10$.

In Table 4.17, cost analysis of high-order 2N-storage RK schemes is presented. The number of elements in spatial domain $N = 100$ is fixed. The order of approximation for RK4, LDD46, Carpenter(5,4), HALE7, RKC84 and RKF84 is $p = 3$.

Schemes	Elapsed Time	Cost	L^2-Norm Error
RK4	0.38 Seconds	1	6.9069e-9
Carpenter	0.33 Seconds	0.86	6.9217e-09
LDD46	0.43 Seconds	1.13	6.899e-09
HALE7	0.48 Seconds	1.26	6.008e-09
RKF84	0.27 Seconds	0.71	6.925e-09
RKC84	0.31 Seconds	0.81	8.3439e-09

Table 4.17: Cost analysis of RK4 and 2N-storage schemes with $p = 3$, $N = 100$ and $T = 1$.

The accuracy is measured by L^2-norm error (4.14) and it can be seen that the error

magnitude of 2N-storage RK schemes is almost similar to RK4. Since the number of stages of 2N-storage RK schemes are different so the computational cost varies in all schemes. The cost is estimated by the ratio of each 2N-storage RK scheme runtime over RK4 computation time. Note that the 4th-order 2N-storage schemes are less costly as compared to the classical 4th-order RK scheme although, 2N-storage schemes have more stages. The solution is approximated for one full period of time $T = 1$, Δt is computed with different values of CFL numbers using RK-DG pairings from Tables 4.10 and 4.11.

Schemes	Elapsed Time	Cost	L^2-Norm Error
RK4	2.062 Seconds	1	7.1183e-9
Carpenter	1.619 Seconds	0.78	7.4884e-09
LDD46	2.11 Seconds	1.02	6.909e-09
HALE7	2.453 Seconds	1.89	6.922e-09
RKF84	1.3839 Seconds	0.67	7.6127e-09
RKC84	1.360 Seconds	0.65	2.440e-08

Table 4.18: Cost analysis of 2N-storage schemes and RK4 with $p = 3$, $N = 100$ and $T = 5$.

In Table 4.18, the efficiency of 2N-storage schemes has been demonstrated when final time is $T = 5$. We can see that when the final time T is increased, the temporal error grows but final time T does not effect the efficiency of 2N-storage schemes. We see that Carpenter(5,4) and RKF84 schemes produce little increment in the L^2-norm error while RKC84 scheme produces larger error as compared to other 2N-storage schemes.

Schemes	Elapsed Time	Cost	L^2-Norm Error
RK4	20.305 Seconds	1	1.8977e-08
Carpenter	18.04 Seconds	0.88	3.0307e-08
LDD46	21.006 Seconds	1.0345	8.0518e-09
HALE7	23.94 Seconds	1.17	8.5912e-09
RKF84	14.84 Seconds	0.73	3.3251e-08
RKC84	14.24 Seconds	0.70	2.511e-07

Table 4.19: Cost analysis of 2N-storage schemes and RK4 with $p = 3$, $N = 100$ and $T = 50$.

From the above discussion, the benefits of efficient high-order 2N-storage schemes have been demonstrated in terms of accuracy and cost measures. Thus, accuracy and cost measure are the most important features of any numerical method.

Schemes	Elapsed Time	Cost	L^2-Norm Error
RK3	5.5 Seconds	1	2.7328e-05
RKC73	4.5 Seconds	0.80	1.002e-04

Table 4.20: Cost analysis of RK3 and RKC73 with $p = 2$, $N = 100$ and $T = 50$.

The computational cost and L^2-norm error of RK3 and RKC73 is presented in Table 4.20. It is observed that RKC73 cost is decreased by 20% to compute the solution but results in larger error.

2N-storage schemes provide more wider absolute stability regions and larger time steps but spatial and temporal discretization error also contribute in the efficiency of schemes. The computational efficiency of 2N-storage schemes depends on the element size, time step and the order of approximation p. Thus, to investigate the efficiency of 2N-storage schemes in more unique way, a scenario is considered as if we fixed the order of approximation p and compute the solution for a certain time.

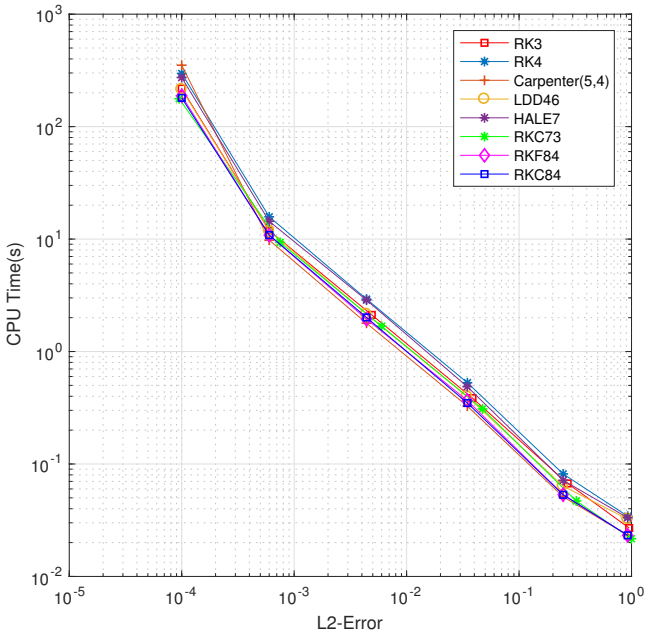

Figure 4.8: CPU-time and L^2-norm error analysis of the 2N-storage RK schemes for linear advection equation with $p = 1$ and $T = 100$.

In Fig 4.8, the order of approximation is $p = 1$ and runtime is T=100. The computational time is calculated versus L^2-norm error with refinement of mesh. We observe from the graph that the high-order 2N-storage schemes are more efficient.

In order to illustrate the efficiency of 2N-storage schemes, we have considered the higher order of approximation as $p = 2, 3$ with run time $T = 100$. From Figure 4.9, it is clear that when $p = 2$, 2N-storage schemes use less time to reach final time T. On the other hand, RKC73 is more efficient but produce large error.

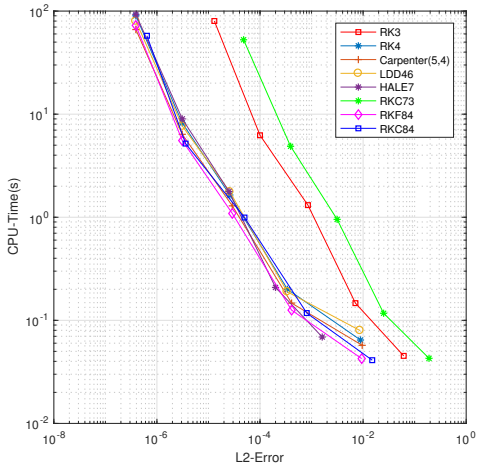

Figure 4.9: CPU time and L^2-norm error analysis of the 2N-storage RK schemes for linear advection equation with $p = 2$ and $T = 100$.

In addition, when $p = 3$ and run time is $T = 100$, Figure 4.10 shows that the high-order 2N-storage methods are the most efficient in terms of L^2 error and computational time. HALE7, Carpenter(5,4) and LDD46 are less efficient while RKF84 performs similar to RK4 when p=3.

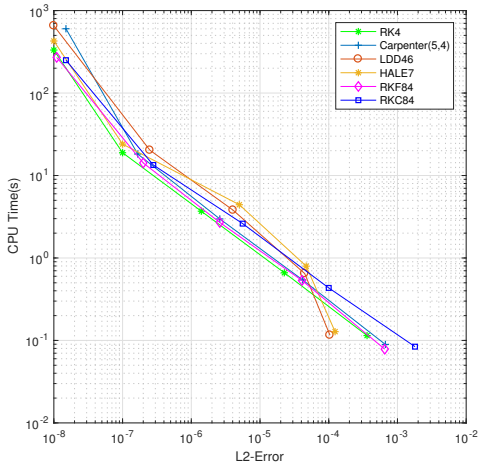

Figure 4.10: CPU-time and L^2-norm error analysis of the 2N-storage RK schemes for linear advection equation with $p = 3$ and $T = 100$.

4.9 Conclusion

We have presented the performance of high-order explicit 2N-storage schemes presented in literature combined with the DG spatial method. The high-order 2N-storage schemes with their main features are reported with classical Runge-Kutta methods for comparison. The computational efficiency has been obtained by combining the mesh size and the numerical method that gives solution in less time and required accuracy. General L^2-norm error and cost measures are used to attain the desired accuracy and computational cost effects.

We have examined that these efficient 2N-storage time integrators can be utilized with the DG spatial scheme. These 2N-storage schemes maintain the order of accuracy in L^2-norm. High-order 2N-storage time integrators are stable and do not produce instabilities during the change in order of approximation p.

The performance indicators of 2N-storage schemes have demonstrated the significant features in terms of stability, accuracy and cost computations. It is shown that 2N-storage schemes are suitable choice for the DG method while solving linear wave propagation problems with smooth initial profile.

During the optimization process, it is observed that the RKF84 is the most efficient and suitable schemes for use with the discontinuous Galerkin method. Better approximations are obtained while using RKF84 in the form of accuracy and less expansive in computational cost.

In addition, the coefficients of 2N-storage schemes have been provided in Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 for user. The CFL numbers with different order of approximations have been presented as well in Tables 4.10 and 4.11. Also, the relation to coefficient of 2N-storage schemes and Butcher tableau coefficients is presented in Appendix A.

4.10 Future work

Exploration of more efficient time integrators associated with the DG method while solving hyperbolic partial differential equations is one course of future work. Effective application of these schemes on time-dependent, non-linear and high dimensional problems is another fascinating field of study. High-order 2N-storage RK time integrators are more efficient than classical RK schemes so they can be used with other time-stepping methods such as implicit time integration and the local time-stepping.

References

[1] M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions.* Dover, New York, 1965.

[2] V. Allampalli, R. Hixon, M. Nallasamy, and S.D. Sawyer. High-accuracy large-step explicit Runge-Kutta (HALE-RK) schemes for computational aeroacoustic. *Journal of Computational Physics.*228, 3837-3850, 2009.

[3] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.* Volume 61. SIAM, 1998.

[4] G.A. Baker and P.R. Graves-Morris. *Padé Approximants.* Addison-Wesley, Reading, Mass, Don Mills, Ont, 1981.

[5] M. Bernardini and S. Pirozzoli. A general strategy for the optimization of Rung-Kutta schemes for wave propagation phenomena. *Journal of Computational Physics.* 228, 4182-4199, 2009.

[6] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations.* John Wiley Sons, Ltd., Chichester, England, 2003.

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

[8] M. Calvo, J.M. Franco, and L. Rández. Minimum storage Runge-Kutta schemes for computational acoustics. Journal of Computational Mathematics Appl.45, 535-545, 2003.

[9] M.H. Carpenter and C.A. Kennedy. Fourth-Order 2N-storage Runge-Kutta schemes, technical memorandum. *NASA-TM-109112, NASA Langley Research Center*, 1994.

[10] B. Cockburn and C.W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin methods for scalar conservation laws II: General framework. *Mathematics of Computation*, 52:411435, 1989.

[11] B. Cockburn and C.W. Shu. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing* 16.3:173-261, 2001.

[12] B. Cockburn and C.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): 1998.

[13] X. Feng, O. Karakashian, and Y. Xing. Recent developments in discontinuous Galerkin finite element methods for partial differential equations. *Springer*, 2012.

[14] R. Haberman. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*. Pearson Prentice Hall, Upper Saddle River, N.J, 2004.

[15] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer, Berlin Heidelberg, 2008.

[16] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49, 357-393, 1983.

[17] P. Henrici. *Applied and Computational Complex Analysis. Special Functions-Integral Transforms-asymptotic-continued fractions*. Volume 2. John Wiley & Sons, 1977.

[18] J.S. Hesthaven and T. Warbruton. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, texts in applied Mathematics Vol.54. Springer-Verlag, New York, USA, 2008.

[19] F.Q. Hu and H.L. Atkins. Eigen-solution analysis of the discontinuous Galerkin method with nonuniform grids. Journal of Computational Physics, 182, 516-545, 2002.

[20] F. Q. Hu, M. Hussaini, and P. Rasetarinera. An analysis of the discontinuous Galerkin method for wave propagation problems. Journal of Computational Physics.151(2), 921-946, 1999.

[21] F.Q. Hu, M.Y. Hussaini, and J.L. Manthey. Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics. Journal of Computational Physics.124, 177-191, 1996.

[22] C. Johnson and J. Pitkrata. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. Mathematics of computation,46, 1-26, 1986.

[23] S. Gottlieb J.S Hesthaven and D. Gottlieb. Spectral methods for time dependent problems. Cambridge University Press, Cambridge, 2007.

[24] A. Kanevsky, M. H. Carpenter, D. Gottlieb, and J. S. Hesthaven. Application of implicit-explicit high order Runge-Kutta methods to discontinuous-Galerkin schemes. Journal of Computational Physics, 225(2):, 2007.

[25] R. Kress. Numerical analysis. Volume 181 of Graduate Texts in Mathematics, Springer, 1998.

[26] L. Krivodonova and R. Qin. An analysis of the spectrum of the discontinuous Galerkin method. Journal of Applied Numerical Mathematics, Vol. 64., 1-18, 2013.

[27] D. Kuzmin. A guide to numerical methods for transport equations. 2010.

[28] R. J. LeVeque. Finite volume methods for hyperbolic problems. volume 31. Cambridge university press, 2002.

[29] X. Meng, C. W. Shu, Q. Zhang, and B. Wu. Super-convergence of discontinuous Galerkin methods for scalar nonlinear conservation laws in one space dimension. SIAM Journal on Numerical Analysis, vol. 50, no. 5, pp. 2336-2356, 2012.

[30] K. W. Morton and D. F. Mayers. Numerical solution of partial differential equations: An introduction. Cambridge University Press, 2005.

[31] T. Peterson. A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. SIAM ,Journal on Numerical Analysis,28, 133-140, 1991.

[32] S. Pirozzoli. Performance analysis and optimization of finite-difference schemes for wave propagation problems. Journal of Computational Physics. 222, 809-831, 2007.

[33] E. Süli and D. Mayers. An Introduction to Numerical Analysis. Cambridge University Press, ISBN 0-521-00794-1,2003.

[34] P. Le Saint and P. Raviart. On a finite element method for solving the neutron transport equation. Academic Press, New York, 1974.

[35] C.Wang Shu. Discontinuous Galerkin methods: general approach and stability. Division of Applied Mathematics, Brown University, RI 02912, USA.

[36] D. Stanescu and W.G. Habashi. 2N-storage low dissipation and dispersion Runge-Kutta schemes for computational acoustics. Journal of Computational Physics. 143, 674-681, 1998.

[37] E. Toro. *Riemann solvers and numerical methods for fluid dynamics.* Springer, 1999.

[38] T. Toulorge* and W. Desmet. Optimal Runge-Kutta schemes for discontinuous Galerkin space discretization applied to wave propagation problems. Journal of Computational Physics. 231., 2067-2091, 2012.

[39] J. H. Verner. Explicit Runge-Kutta methods with estimates of the local truncation error. SIAM Journal of Numerical Analysis, 15, 772-790, 1978.

[40] J.H. Williamson. Low-storage Runge-Kutta schemes. Journal of Computational Physics.35, 48-56, 1980.

[41] Q. Zhang and C.Wang Shu. Error estimates to smooth solutions of Runge-Kutta discontinuous Galerkin methods for scalar conservation laws. SIAM Journal on Numerical Analysis, 44, 1703-1720, 2004.

APPENDICES

Appendix A

A.1 The relation to classical RK variables and 2N-storage variables

The relationship between the original RK variables $a_{i,j}, b_i, c_i$ and the 2N-storage variables A_i, B_i up to eight stages are given in this section.

$a_{2,1} = B_1,$
$a_{3,1} = A_2B_2 + B_1,$
$a_{3,2} = B_2,$
$a_{4,1} = A_2(A_3B_3 + B_2) + B_1,$
$a_{4,2} = A_3B_3 + B_2,$
$a_{4,3} = B_3,$
$a_{5,1} = A_2(A_3(A_4B_4 + B_3) + B_2) + B_1,$
$a_{5,2} = A_3(A_4B_4 + B_3) + B_2,$
$a_{5,3} = A_4B_4 + B_3,$
$a_{5,4} = B_4,$
$a_{6,1} = A_2(A_3(A_4(A_5B_5 + B_4) + B_3) + B_2) + B_1,$
$a_{6,2} = A_3(A_4(A_5B_5 + B_4) + B_3) + B_2,$
$a_{6,3} = A_4(A_5B_5 + B_4) + B_3,$
$a_{6,4} = A_5B_5 + B_4,$
$a_{6,5} = B_5,$

$$
\begin{aligned}
a_{7,1} &= A_2(A_3(A_4(A_5(A_6B_6 + B_5) + B_4) + B_3) + B_2) + B_1,\\
a_{7,2} &= A_3(A_4(A_5(A_6B_6 + B_5) + B_4) + B_3) + B_2,\\
a_{7,3} &= A_4(A_5(A_6B_6 + B_5) + B_4) + B_3,\\
a_{7,4} &= A_5(A_6B_6 + B_5) + B_4,\\
a_{7,5} &= A_6B_6 + B_5,\\
a_{7,6} &= B_6,
\end{aligned}
$$

$$
\begin{aligned}
a_{8,1} &= A_2(A_3(A_4(A_5(A_6(A_7A_7 + B_6) + B_5) + B_4) + B_3) + B_2) + B_1,\\
a_{8,2} &= A_3(A_4(A_5(A_6(A_7B_7 + B_6) + B_5) + B_4) + B_3) + B_2,\\
a_{8,3} &= A_4(A_5(A_6(A_7B_7 + B_6) + B_5) + B_4) + B_3,\\
a_{8,4} &= A_5(A_6(A_7B_7 + B_6) + B_5) + B_4,\\
a_{8,5} &= A_6(A_7B_7 + B_6) + B_5,\\
a_{8,6} &= A_7B_7 + B_6,\\
a_{8,7} &= A_7,
\end{aligned}
$$

$$
\begin{aligned}
b_1 &= A_2(A_3(A_4(A_5(A_6(A_7(A_8B_8 + B_7) + B_6) + B_5) + B_4) + B_3) + B_2) + B_1,\\
b_2 &= A_3(A_4(A_5(A_6(A_7(A_8B_8 + B_7) + B_6) + B_5) + B_4) + B_3) + B_2,\\
b_3 &= A_4(A_5(A_6(A_7(A_8B_8 + B_7) + B_6) + B_5) + B_4) + B_3,\\
b_4 &= A_5(A_6(A_7(A_8B_8 + B_7) + B_6) + B_5) + B_4,\\
b_5 &= A_6(A_7(A_8B_8 + B_7) + B_6) + B_5,\\
b_6 &= A_7(A_8B_8 + B_7) + B_6,\\
b_7 &= A_8B_8 + B_7,\\
b_8 &= B_8,
\end{aligned}
$$

$$
\begin{aligned}
c_1 &= 0,\\
c_2 &= B_1,\\
c_3 &= B_1 + B_2(A_2+1),\\
c_4 &= B_1 + B_2(A_2+1) + B_3(A_3(A_2+1)+1),\\
c_5 &= B_1 + B_2(A_2+1) + B_3(A_3(A_2+1)+1) + B_4(A_4(A_3(A_2+1)+1)+1),\\
c_6 &= B_1 + B_2(A_2+1) + B_3(A_3(A_2+1)+1) + B_4(A_4(A_3(A_2+1)+1)+1)\\
&\quad + B_5(A_5(A_4(A_3(A_2+1)+1)+1)+1),\\
c_7 &= B_1 + B_2(A_2+1) + B_3(A_3(A_2+1)+1) + B_4(A_4(A_3(A_2+1)+1)+1)\\
&\quad + B_5(A_5(A_4(A_3(A_2+1)+1)+1)+1)\\
&\quad + B_6(A_6(A_5(A_4(A_3(A_2+1)+1)+1)+1)+1),\\
c_8 &= B_1 + B_2(A_2+1) + B_3(A_3(A_2+1)+1) + B_4(A_4(A_3(A_2+1)+1)+1)\\
&\quad + B_5(A_5(A_4(A_3(A_2+1)+1)+1)+1)\\
&\quad + B_6(A_6(A_5(A_4(A_3(A_2+1)+1)+1)+1)+1)\\
&\quad + B_7(A_6(A_7(A_6(A_5(A_4(A_3(A_2+1)+1)+1)+1)+1)+1)+1).
\end{aligned}
$$