Comparison of Radau and Lobatto Methods With a Novel Adaptive Time Stepping for Fluid Dynamics Problems

Tadas Paskevicius

Bachelor's thesis
2023:K26

Lund University

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Abstract

Fully implicit Runge-Kutta methods play an important role in the numerical integration of stiff differential equations. Radau IIA is the most commonly used fully implicit method due to its good theoretical performance. However, Radau IA and Lobatto IIIC might also be suitable for different problems. These three methods are constructed using simplified order conditions that ensures specific integration properties. The theoretical performance is derived from various stability and order concepts. Specifically, Lobatto IIIC has a lower order, and Radau IA has a lower stiff order than Radau IIA.

A new adaptive time-stepping method is proposed and used. The order of convergence and properties of the adaptive time-stepping are examined for three distinct problems: The linear test equation (ODE), a combustion equation (ODE), and an advection-diffusion equation (PDE). The compressible Euler equations (PDE) are also assessed, albeit to a lesser extent due to computational constraints. Both the advection-diffusion equation and the Euler equations pertain to fluid dynamics, which is a primary area of interest for implicit Runge-Kutta methods. For the two last problems DUNE library [2] is used for spatial discretization. Numerical we see that Radau IIA best overall performing, Lobatto IIIC being reasonable good performing and Radau IA being not as good when the problem is stiff.

Acknowledgement

I would like to thank my supervisors, Robert Klöfkorn and Viktor Linders, as well as my examiner, Tony Stillfjord. I am also grateful for the support of my family and my dear friends.

Contents

1 Introduction

Fully implicit Runge-Kutta methods form a family of techniques specifically designed to tackle stiff initial value problems (IVP). Stiff IVPs often pose analytical challenges, sometimes rendering them intractable or impossible to solve in a closed form. Furthermore, there are instances where finding an analytic solution is not even desirable for downstream applications. In such cases, the objective becomes discretizing the problem in time and computing a discrete solution. One effective approach for solving these problems is through the utilization of Runge-Kutta methods. They are a family of single-step methods employed for the solution of IVPs. The underlying principle involves taking discrete steps along the direction of the ordinary differential equation (ODE) and, upon landing at each step, initiating another step in the direction of the ODE at that specific point. The sequence of pints traced by the method is the numerical solution to the problem.

Notably, the simplest member of this family is the Euler method, which was first published by Euler himself in 1768. Subsequently, Runge and Kutta extended the Euler method in their respective papers in 1895 and 1901, resulting in improved accuracy. Building upon this and further developments in the field, Kuntzmann and Butcher independently introduced implicit Runge-Kutta methods in 1961 and 1964, respectively. These implicit methods offered even higher levels of accuracy. In addition they are particularly valuable when dealing with stiff IVPs, as they can handle the stiff systems without the use of very small time steps [6].

In recent times it has been proven that some implicit Runge-Kutta methods are equivalent to continuous and discontinuous Galerkin methods [20, 3]. This means that Runge-Kutta theory can be applied to Galerkin methods and vice versa. The allure of Galerkin methods is the potential of parallelization of the solver [9], which is of interest in the field of partial differential equation solvers. Galerkin methods are of particular use in computational fluid dynamics (CFD), and it is in this context that we wish to investigate the properties of certain implicit Runge-Kutta methods.

In this thesis, we explore three such Runge-Kutta methods for solving initial value problems (IVPs): Radau IA, Radau IIA, and Lobatto IIIC. We will delve into their construction and properties. Additionally, we will introduce a novel embedded method for adaptive time stepping with a feedback loop. To evaluate the efficacy of these methods, we will test their performance within the realm of fluid dynamics problems. According to the foundational theory of Runge-Kutta methods, Radau IIA is expected to outperform the others in terms of order and stiff stability. Lobatto IIIC, while stiffly stable, is anticipated to be of a lower order. On the other hand, Radau IA, despite being of the same order as Radau IIA, not stiffly stable and may experience order reduction for stiff problems. The goal of this thesis is to investigate whether these hypotheses hold in the context of fluid dynamics problems.

Section 2, give some background in Runge-Kutta methods and ODEs, section 3 delves into specifics of Lobatto IIIC, Radau IA and Radau IIA. Stability and order concepts are examined in Section 4. Next, Section 5 covers implementation details with focus on adaptive time stepping. Section 6 showcases experimental results from various test cases with focus on computational fluid dynamics. A summary and conclusion is given in Section 7. Finally, we discuss some additional observations and outlooks in Section 8.

2 Background

In this section we will give some background on stiffness in problems, introduce the Runge-Kutta method and lightly touch on error and order.

2.1 Ordinary differential equations (ODEs)

In this thesis we will consider systems of first order ordinary differential equations. While the family of differential equations is much larger they can be reduced and/or solved with analytical or numerical methods to a system of first order ODEs such that an IVP solver can also be used. For example, spatially discretized PDEs, as in problems 6.3 and 6.4.

Definition 1 (First order system of ODEs)**.**

$$\frac{dy}{dt} = f(t, y), \tag{1}$$

where

- $y(t)$ *is a vector in* $\mathbb{R}^n$ *(i.e.,* $y = [y_1, y_2, \ldots, y_n]^T$*),*
- $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ *is a given vector-valued function.*

The system is subject to the initial condition

$$y(t_0) = y_0, \tag{2}$$

where y_0 *is a given vector in* $\mathbb{R}^n$*.*

If the IVP fulfills the conditions stated in the Picard-Lindelöf theorem, it guarantees the existence and uniqueness of a solution.

Theorem 1 (Picard-Lindelöf)**.** *Let* $f : [t_0 - a, t_0 + a] \times \mathbb{R}^n \to \mathbb{R}^n$ *be continuous and satisfy a Lipschitz condition in* y*. If* (t_0, y_0) *is a point in the domain, then there exists an interval* $I = (t_0 - \delta, t_0 + \delta)$ *and a unique function* $y(t) : I \to \mathbb{R}^n$ *that solves the initial value problem in definition 1.*

2.2 Stiffness

Stiffness is a property of certain ODEs characterized by the fact that using explicit time stepping solvers results in instability unless prohibitively small step sizes are used, making the method computationally expensive. There are unfortunately no good definitions for stiffness for a given problem. The main issue with defining stiffness is that it is dependent on the solver, step size and the ODE. However, stiffness arises from the fact that the neighboring solution curves of the IVPs differ significantly from the one being computed, meaning that the inherent inaccuracy of discrete methods may lead to catastrophic deviation from the intended solution . In other words, very small step sizes may be needed to stay on the correct solution curve.

An example of a stiff ODE is an equation modeling combustion:

$$\frac{dy}{dt} = y^2 - y^3. \tag{3}$$

Here, the solution y represents the radius of a sphere of a combustion reaction in the range normalized to the range $(y(t_0), 1)$ where $y(t_0) < 1$. Physically, the rate of change of the combustion radius is proportional to the difference between the surface area and volume of the sphere [7]. The stiffness of the problem arises from the fact that the combustion acceleration is highly dependent on the initial radius y_0. This can be seen in Figure 1, which shows the vector field of (3) with five solution curves with different initial values. The solution curves are very close to each other in the beginning but quickly diverge. Another sources of stiffness could also be oscillations in the problem [16]. Stiffness can loosely be characterized as the presence of multiple time scales in the problem [10, 12].

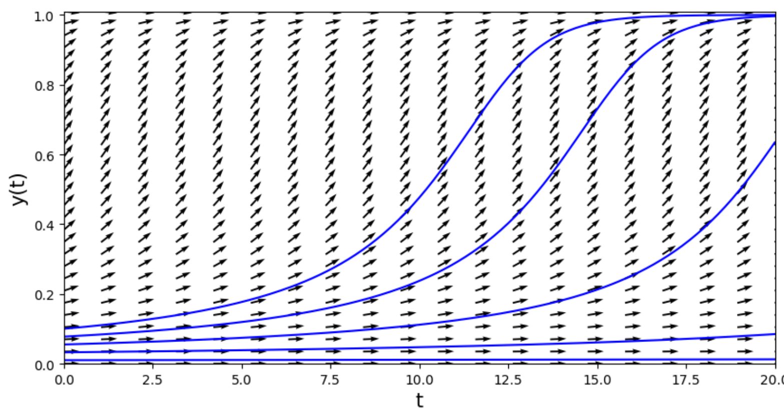


Figure 1: Vector field of the combustion problem (3) with five solution curves with the initial values $y_0 \in \{0.01, 0.0325, 0.055, 0.0775, 0.1\}$.

2.3 Runge-Kutta methods

Runge-Kutta methods are a family of one step methods for numerically integrating. This is done by sampling the function $f(t, y)$ at multiple time points

$t_n + c_i h$, where $h > 0$ is the time increment. These sampled values k_i are called *stage derivatives*, and are given by

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{s} a_{ij} k_j). \tag{4}$$

Here, c_i are constants for time nodes and a_{ij} weights of the stage derivatives. The stage derivatives are then combined to form the approximation at the next time step with the weights b_i as follows:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i. \tag{5}$$

The Runge-Kutta method can alternatively be rephrased for ease of equation manipulation as

$$g_i = y_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_i h, g_i), \tag{6}$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i f(t_n + c_i h, g_i). \tag{7}$$

Here, $y_{n+1} \approx y(t_n + h)$. In fact, Runge-Kutta methods can be viewed as approximations to a Taylor series expansion of $y(t + h)$ around $y(t)$.

A way to represent Runge-Kutta methods is with a Butcher tableau, which is a compact way to represent the constants c_i, b_i and a_{ij} as vectors and matrix respectively.

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

where A is a matrix consisting of a_{ij} coefficients, b is a vector consisting of b_i coefficients and c is a vector consisting of c_i coefficients in (4) and (5). There are mainly three types of Runge-Kutta methods: explicit, implicit, and semi-implicit. The explicit methods have the property that the matrix A is lower triangular, which means that the stage derivatives k_i can be computed sequentially. In contrast, the implicit methods have a matrix A with non-zero coefficients outside the lower triangular part, so the stage derivatives must be solved for simultaneously. The semi-implicit methods are a hybrid approach where some stage derivatives are solved sequentially, while others are solved for simultaneously.

The simplest Runge-Kutta method is the explicit Euler method,

$$y_{n+1} = y_n + h f(t_n, y_n).$$

Its Butcher tableau is given by $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$

2.4 Local, global truncation error and order

Numerical methods introduce discretization errors. The *local error* is the error introduced in a single time step $\tau_h^{n+1} = ||y(t_{n+1}) - y_{n+1}||$. The *global error* is the accumulated error at the final time $e_h^{\text{final}} = ||y(t_{\text{final}}) - y_{\text{final}}||$. The order of a method is the rate at which the global error decreases with the step size h,

$$e_h^{\text{final}} \leq Ch^p. \tag{8}$$

Here, p is the order of the method and C is a constant independent of h such that the inequality holds for sufficiently small h. An alternative expression for this is $e_h^{\text{final}} = \mathcal{O}(h^p)$.

A method is said to be *consistent* if $\lim_{h\to 0} \frac{\tau_h^n}{h} = 0$. It is said to be *convergent* if $\lim_{h\to 0} e_h^{\text{final}} = 0$. For a method to be convergent it is necessary for it to be consistent and stable, the latter of which will be discussed in Section 3

3 Radau IA, Radau IIA, Lobatto IIIC

The Runge-Kutta methods discussed in this thesis are the three-stage Radau IA, Radau IIA and Lobatto IIIC methods. They are fully implicit methods, hence the system of stage equations (4) must be solved in each time step. However, they offer advantageous stability properties for stiff problems compared to explicit methods. In this section we will introduce these methods and discuss their construction.

3.1 Order Conditions

The Runge-Kutta methods can be derived from the simplified order conditions set out by J.C Butcher in [5] which stems from Butcher group or also know as Butcher trees. :

$$B(p) : \sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} \qquad q = 1, 2, \ldots, p \tag{9}$$

$$C(\eta) : \sum_{j=1}^{s} a_{ij} c_j^{q-1} = \frac{c_i^q}{q} \qquad i = 1, 2, \ldots, s \quad q = 1, 2, \ldots, \eta \tag{10}$$

$$D(r) : \sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \frac{b_j(1 - c_j^q)}{q} \qquad j = 1, 2, \ldots, s \quad q = 1, 2, \ldots, r \tag{11}$$

The order conditions for a s-stage method $B(p)$ and $C(\eta)$ are equivalent to a quadrature rule that integrates a polynomial of order p and η exactly, respectively. The order of a Runge-Kutta method given the order conditions is satisfying by the following result from [5]:

Theorem 2. *If the coefficients b_i, c_i, and a_{ij} of a Runge-Kutta method satisfy $B(p)$, $C(\eta)$, $D(r)$ with $p \leq \eta + r + 1$ and $p \leq 2r + 2$, then the method is of order p.*

3.2 Construction of the methods

The c_i coefficients for the methods discussed are based on the roots of the right shifted Legendre polynomials with $x \in [0, 1]$. These polynomials P_s are expressed by Rodrigues' formula

$$P_s(x) = \frac{1}{s!} \frac{d^s}{dx^s} x^s (x-1)^s.$$

Using the roots of these polynomials as c_i coefficients result in Gaussian quadrature rules, which have the highest possible order $2s$. However, the methods considered here will specify at least one of c_1 and c_s to coincide with the boundaries of the interval $[0, 1]$. As we will see, this results in advantageous stability properties.

3.2.1 Radau IA

For Radau IA, the coefficients c_i are given by the roots of

$$P_s(x) + P_{s-1}(x) = \frac{d^{s-1}}{dx^{s-1}} x^s (x-1)^{s-1}. \tag{12}$$

This determines the collocation point $c_1 = 0$ resulting in a quadrature of order $2s-1$. Additionally, the coefficients b_i and a_{ij} are uniquely defined by satisfying $B(2s-1)$ and $D(s)$. For $s = 3$, this results in the Butcher tableau:

$$\begin{array}{c|ccc}
0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\
\frac{3}{5} - \frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45} + \frac{7\sqrt{6}}{360} & \frac{11}{45} - \frac{43\sqrt{6}}{360} \\
\frac{3}{5} + \frac{\sqrt{6}}{10} & \frac{1}{9} & \frac{11}{45} + \frac{43\sqrt{6}}{360} & \frac{11}{45} - \frac{7\sqrt{6}}{360} \\
\hline
 & \frac{1}{9} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{4}{9} - \frac{\sqrt{6}}{36}
\end{array}$$

3.2.2 Radau IIA

For Radau IIA, the coefficients c_i coefficients are given by the roots of

$$P_s(x) - P_{s-1}(x) = \frac{d^{s-1}}{dx^{s-1}} x^{s-1} (x-1)^s. \tag{13}$$

This determines the collocation point $c_s = 1$ resulting in a quadrature of order $2s-1$. Additionally, the coefficients b_i and a_{ij} are uniquely defined by satisfying $B(2s-1)$ and $C(s)$. For $s = 3$, this results in the Butcher tableau

$$\begin{array}{c|ccc}
\frac{2}{5} - \frac{\sqrt{6}}{10} & \frac{11}{45} - \frac{7\sqrt{6}}{360} & \frac{37}{225} - \frac{169\sqrt{6}}{1800} & -\frac{2}{225} + \frac{\sqrt{6}}{75} \\
\frac{2}{5} + \frac{\sqrt{6}}{10} & \frac{37}{225} + \frac{169\sqrt{6}}{1800} & \frac{11}{45} + \frac{7\sqrt{6}}{360} & -\frac{2}{225} - \frac{\sqrt{6}}{75} \\
1 & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \\
\hline
 & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9}
\end{array}$$

3.2.3 Lobatto IIIC

For Lobatto IIIC, the coefficients c_i are given by the roots of

$$P_s(x) - P_{s-2}(x) = \frac{d^{s-2}}{dx^{s-2}} x^{s-1}(x-1)^{s-1}. \tag{14}$$

This determines the collocation points $c_1 = 0$ and $c_s = 1$ giving a quadrature of two orders lower than the Gauss method at $2s-2$. Additionally, the coefficients b_i are derived from $B(2s-2)$ and a_{ij} coefficients are derived from $C(s-1)$ and $a_{i1} = b_1$. For $s = 3$, this results in the Butcher tableau

$$\begin{array}{c|ccc} 0 & 1/6 & -1/3 & 1/6 \\ 1/2 & 1/6 & 5/12 & -1/12 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \end{array}$$

4 Stability

In this section we will introduce different stability concepts. It is important that the methods are stable so that the solution converges and does not explode in size. There are a multitude of stability concepts pertaining to Runge-Kutta methods. Here, we discuss a few of them with particular relevance to the numerical experiments in Section 6

4.1 A-stability

A-stability is based on how the method behaves on the linear test equation

$$\begin{aligned} y' &= \lambda y, \\ y(0) &= 1. \end{aligned} \tag{15}$$

Here λ is complex valued and the analytical solution is

$$y(t) = e^{\lambda t}.$$

The linear test equation has the property that if $\mathrm{Re}(\lambda) < 0$ then the solution will converge to zero as $t \to \infty$. A-stability captures this property for a numerical method. If a Runge-Kutta method is applied to the linear test equation, the solution at consecutive time steps satisfy [10]

$$y_{n+1} = R(h\lambda)y_n \equiv (1 + h\lambda b^T (I - h\lambda A)^{-1}\vec{e})y_n.$$

Here $\vec{e}$ is a vector of ones and $R(\cdot)$ is known as the stability function of the Runge-Kutta method.

Thus, the numerical solution satisfies $y_n \to 0$ as $n \to \infty$ if and only if $|R(h\lambda)| < 1$. The method is said to be *absolutely stable* for those values of $h\lambda$ that satisfies this criterion. This leads to the following definition:

Definition 2. *A method is said to be A-stable if it is absolutely stable for all* $\mathrm{Re}(h\lambda) < 0$.

Radau IA, Radau IIA and Lobatto IIIC are all A-stable. In fact, they all are L-stable, which is a stronger stability concept.

4.2 L-stability

For stiff problems, A-stability might not be enough to ensure stability for stiff problems. Thus, L-stability (or strong A-stability) is introduced [10]:

Definition 3. *A Runge-Kutta method is said to be L-stable if it is A-stable and has the additional property that*

$$\lim_{h\to\infty} R(h\lambda) = 0.$$

As mentioned, Radau IA, Radau IIA and Lobatto IIIC are all L-stable. This suggests that these methods may be suitable for stiff problems as it dampens rapidly decaying transients.

4.3 S-stability, strong S-stability and stiff order

Both A- and L-stability are defined for the linear test problem (15). In an effort to find a stability concept that more accurately predicts the behaviour of Runge-Kutta methods for nonlinear problems, a generalization of the test equation was introduced in [17]:

$$y' = g'(t) + \lambda(y - g(t)). \tag{16}$$

Here, $g(t)$ is an arbitrary bounded function. The solution is given by $y(t) = g(t)$. Here a component of stiffness in the problem is introduced by λ. Note that (16) reduces to the test equation (15) when $g = 0$.

Generalization of A- and L-stability based on (16) were introduced in [17]. They are respectively known as S- and strong S-stability:

Definition 4. *A Runge-Kutta method is S-stable if for all h in an interval $(0, h_0)$ and $\mathrm{Re}(\lambda) < 0$ it holds that*

$$\left| \frac{y_{n+1} - g(t_{n+1})}{y_n - g(t_n)} \right| < 1.$$

It is strongly S-stable if, additionally,

$$\lim_{\lambda\to-\infty} \frac{y_{n+1} - g(t_{n+1})}{y_n - g(t_n)} = 0.$$

Note that S- and strong S-stability reduces to A- and L-stability respectively when $g(t) = 0$.

Associated with these stability concepts are the notion of *stiff order* of accuracy. The stiff order differs from the classical order of the Runge-Kutta method in that it is derived from the stiff test equation (16) and that one considers the limits $h \to 0$ and $\mathrm{Re}(\lambda) \to -\infty$ simultaneously [14]. The stiff order can be interpreted as the rate that the solution converges to $g(t)$ as the step size is reduced and the stiffness is increased. The local truncation error is in these limits understood to scale as follows:

$$\tau_h^n = y_{n,h} - g(t_n) = \mathcal{O}(h^{q+1}(\lambda h)^{-r}).$$

Here the pair (q, r) is the stiff order of the method. It is desirable for both q and r to be as large as possible.

Radau IA, Radau IIA and Lobatto IIIC are all S-stable. Radau IIA and Lobatto IIIC are additionally strongly S-stable, whereas Radau IA is not. As a consequence of this and the order conditions (9)–(11), it turns out that the stiff orders of these methods are different. Radau IA, Radau IIA and Lobatto IIIC have the respective stiff orders $(s-1, 0)$, $(s-1, 1)$ and $(s-2, 1)$ [17]. This suggests that the three methods may potentially perform differently when applied to stiff problems, with Radau IIA having an advantage due to its higher stiff accuracy.

4.4 B-stability

A-, L-, S-stability are linear stability concepts. However, for nonlinear problems, the notion of B-stability can be applied. B-stability is based on the concept of a dissipative problem where some energy of the problem is lost such that the solution eventually enters an absorbing set [10]. This means that independently of the initial value, the solution will converge to a solution set.

Definition 5. *A Runge-Kutta method is said to be B-stable if for a nonlinear problem* $\frac{dy}{dt} = f(t, y)$ *satisfying the contractive property*

$$\langle f(t, y(t)) - f(t, \hat{y}(t)) \,,\, y(t) - \hat{y}(t) \rangle \leq 0,$$

the numerical solution satisfies

$$\|y_{n+1} - \hat{y}_{n+1}\| \;\leq\; \|y_n - \hat{y}_n\|.$$

Here, $\langle \cdot, \cdot \rangle$ *denotes an inner product and* $\| \cdot \|$ *its induced norm.*

An algebraic criterion for B-stability was given by Burrage and Butcher [4] and independently by Crouzeix [8]:

Definition 6. *A Runge-Kutta method is said to be algebraically stable if the matrices*

$$B := \mathrm{diag}(b_1, \ldots, b_s)$$

and

$$M := BA + A^T B - bb^T$$

are positive semi-definite.

Radau IA, Radau IIA and Lobatto IIIC are all algebraically stable and consequently B-stable.[4, Thm 2.2]

4.5 Summary

The stability properties of the Radau IA, Radau IIA, and Lobatto IIIC methods are summarized in the following table:

Table 1: Comparison of s-stage Radau IA, Radau IIA, and Lobatto IIIC methods.

	Radau IA	**Radau IIA**	**Lobatto IIIC**
Order	$2s-1$	$2s-1$	$2s-2$
A-stability	Yes	Yes	Yes
L-stability	Yes	Yes	Yes
S-stability	Yes	Yes	Yes
Strong S-stability	No	Yes	Yes
Stiff order	$(s-1,0)$	$(s-1,1)$	$(s-2,1)$
B-stability	Yes	Yes	Yes

The primary distinction among the methods Lobatto IIIC, Radau IA, and Radau IIA lies in their respective orders and strong stability properties. Theoretically, Radau IIA possesses superior properties for stiff problems, closely followed by Lobatto IIIC, albeit at a reduced order. On the other hand, while Radau IA demonstrates good order properties, it may be prone to order reduction due to its lack of strong S-stability and its inferior stiff stability order. In the realm of computational fluid dynamics, particularly when addressing stiffness, Radau IIA appears to be a promising candidate among these methods. It is one of the goals of this thesis to investigate how the methods behave in practice.

5 Implementation

The methods are implemented in to Assimulo [1], a unified framework for ODE solvers. In this section we will go through the practical aspects of implementing the implicit Runge-Kutta methods. In addition we introduce a novel adaptive time step controller.

5.1 Adaptive time step

Numerical integration benefits significantly from the use of adaptive time steps. Instead of utilizing a fixed step size throughout the computation, adaptive time stepping dynamically adjust the step size depending on an estimate of the local error. This approach is essential because it allows for improved utilization of computational resources. The aim is to make the step size as large as possible without compromising the desired accuracy, governed by user defined tolerance.

The dynamic step size adjustment is done with the aid of an *embedded* method. The error is approximated as the difference between two numerical solutions with different orders of accuracy. Based on this error estimate, the step size is then adjusted accordingly [10]. The local estimated error is

$$\tau_h^{*n+1} = y^*_{n+1} - y_{n+1}. \tag{17}$$

Here, y_{n+1} denotes the numerical solution at time t^{n+1} obtained using the Butcher tableaus in Section 3, and y^*_{n+1} is the numerical solution computed with a method of lower order staring from y_n. In practice, this method will utilize the same Butcher tableau as the original method, but with the b-coefficients replaced. To impose the user defined tolerance, the estimated error is scaled by the tolerance, yielding a relative error instead:

$$err_{n+1} = \left\| \frac{\tau_h^{*n+1}}{tol} \right\|_2. \tag{18}$$

This returns the ratio between the estimated error and tolerance which is the sum of an absolute and a relative tolerance,

$$tol = Atol + Rtol \cdot \max(|y_n|, |y_{n+1}|).$$

The relative tolerance scales with the magnitude of the solution The step size is then adjusted by a proportional controller, defined trough the relation

$$h_{n+1} = h_n \cdot fac_{n+1} \cdot err_{n+1}^{-1/p}. \tag{19}$$

Here, p is a constant set to one higher than the order of the embedded method, since that is the order of the local estimated error. and fac_{n+1} is a safety factor The safety factor is needed since the error is just an estimation and might be over optimistic, leading to a failure in the numerical method. This controller tries to ensure that the local truncation $\tau^{n+1} < Ch^p_{n+1}$ is constant by varying h.

5.2 Embeded error estimation

The estimation of errors requires selecting a method that has a lower order than the method used for the solution. Since function calls can be costly, there is a preference to reuse the stage derivatives from the primary method. Such methods are termed embedded methods. In [18], it is suggested that the b_i coefficients for an embedded method can be formulated by ensuring that the order condition $B(p-1)$ is satisfied. This is achieved by setting the right-hand side of the final order condition to be 0 instead of $\frac{1}{p}$ in (9). In this thesis which is the noval part of this thesis adaptive time stepping, we alternatively perturb the last order condition with a small constant value $a \neq 0$. This embedded method will also satisfy the order condition $B(p-1)$. This can be expressed in vector form

$$b^* = V_s^{-1}\vec{e}_H = \begin{bmatrix} 1 & 1 & \dots & 1 \\ c_1 & c_2 & \dots & c_s \\ c_1^2 & c_2^2 & \dots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{s-1} & c_2^{s-1} & \dots & c_s^{s-1} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \frac{1}{2} \\ \vdots \\ \frac{1}{s-1} \\ \frac{1}{s-a} \end{bmatrix}, \tag{20}$$

where b^* contains the quadrature weights of the embedded method. For the embedded 3 stage Lobatto IIIC method the b^* coefficients are,

$$b^* = \begin{pmatrix} -\frac{1}{2} + \frac{2}{3-a} \\ 2 - \frac{4}{3-a} \\ -\frac{1}{2} + \frac{2}{3-a} \end{pmatrix}. \tag{21}$$

The embedded methods for Radau IA and IIA are computed similarly. An embedded method constructed in this way will at most have order $s-1$ and consequently so will the error estimator. Note that if $a = 0$, the b coefficients for the original Runge-Kutta method are recovered. In other words, $b^* \to b$ as $a \to 0$. Hence, the local estimated error

$$\tau_h^{*n+1} = y_{n+1}^* - y_{n+1} = h\sum_{i=1}^{s}(b_i - b_i^*)k_i,$$

will tend to zero as $a \to 0$. Therefore, we can make the error estimator more or less pessimistic depending on the choice of a. This allows us to add a feedback loop in the error estimator. We base this loop on the previous step size (19), resulting in a nested radical of previous estimated relative errors:

$$a = \alpha \cdot h_n^{1/p} = \alpha \cdot (fac_n \cdot err_n^{-1/p} \cdot h_{n-1})^{1/p} = \dots \tag{22}$$

Here, α is a small constant, which throughout is chosen to be $\alpha = 0.01$. The 1/p in the exponential in the feedback loop ensures that the feedback is not to extreme.

5.3 Nonlinear solver

To take a step with implicit Runge-Kutta methods, a nonlinear system of equations must be solved. Iterative solvers are typically employed for this task. Among these, Newton's method is prevalent because of its speed and the availability of optimization tricks for implicit Runge-Kutta methods. Newton's method requires the computation of the Jacobian of the IVP. If computing the Jacobian is not feasible, one can resort to Jacobian-free methods like Newton-Krylov methods. However, in this thesis, we adopted the implicit Runge-Kutta optimized simplified Newton method [10]. Further details on implementation and theory can be found in [13].

The nonlinear Newton solver consists of three components: Firstly, a *simplified Newton step* is used in which the Jacobian is reused over multiple time steps. Secondly, leveraging the simplified Newton method allows each iteration's linear system to be solved with a single LU-decomposition. Thirdly, and a decomposition of each linear system into two smaller systems utilizing an eigen-decomposition.

For any given step, if the step size changes or a new Jacobian is needed, a new LU decomposition is done. The Newton iteration continues until the solution for the implicit Runge-Kutta method converges within a predefined number of iterations. If convergence is not achieved, a new Jacobian is called, and the step restarts. Otherwise, the estimated error is calculated. If the estimated error is too large, the step size is reduced and the step recalculated; if not, the step is accepted. When determining the step size for the subsequent step, if the new step size is nearly identical to the previous one, the older step size is retained for the next step.

6 Experiments and results

In this section, we test the performance of Lobatto IIIC, Radau IA and Radau IIA, particularly in the realm of fluid dynamics problems, and examine how the numerical results align with theoretical predictions. Additionally, we aim to assess the performance of the new adaptive time-stepping technique.

Four experiments will be performed on:

- The linear test equation,
- The combustion equation,
- The advection diffusion equation,
- The compressible Euler equations.

The purpose of the linear test equation is to verify the classical order of the three methods. We will also use this equation to verify that the new error estimator (20)–(22) becomes progressively more optimistic as the value of a is decreased. We also use this problem to find suitable values for the constant α in (22). Next, the combustion equation is used to test convergence for a stiff problem with known solution. We also use this problem to benchmark the adaptive time stepping. The advection diffusion problem is used as an example of a partial differential equation with controllable stiffness. It is used to compare the methods on a first example of a PDE. Finally, the compressible Euler equations are used to test the solvers on a system of fully non-linear partial differential equations.

6.1 Linear test equation

To test that the implementation of Lobatto IIIC, Radau IA and Radau IIA have the correct orders of convergence, we solve the linear test equation (15). To this end, the error is measured at the final time $t = 1$ for a sequence of different step sizes. Let $e_h^{final} = |y(t_{final}) - y_{final}|$ denote the error when using step size h. From (8) it follows that

$$\frac{e_h}{e_{\frac{h}{2}}} = 2^p.$$

Therefore, the *experimental order of convergence* (EOC) can be computed as

$$\mathrm{EOC} = \log_2 \frac{e_h}{e_{\frac{h}{2}}} = \frac{\log \frac{e_h}{e_{\frac{h}{2}}}}{\log 2}. \tag{23}$$

For this problem, the error is computed with respect to the analytic solution, $y(t) = e^{\lambda t}$.

In the following experiment, we use $\lambda = 1$ and set the final time to $t_{final} = 1$. The errors of the three Runge-Kutta methods are shown in Fig. 2. All three methods display the order of convergence expected from Table 1.

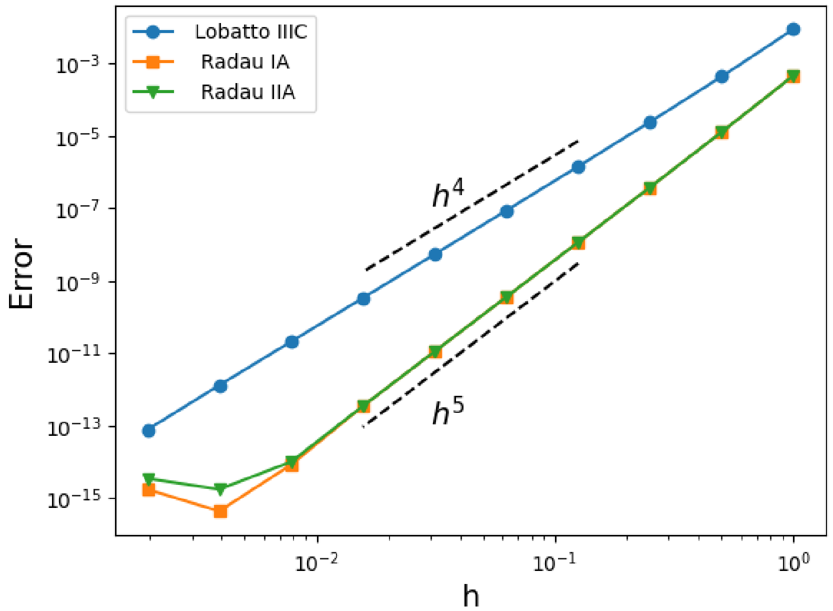


Figure 2: Error vs step size for the linear test equation (15) at time $t_{final} = 1$.

Next, we turn our attention to adaptive time stepping. The embedded method in [18] with $a = \inf$ in (20) are pessimistic error estimators, resulting in an unnecessary number of steps wasting computational time for a desired accuracy. This can can be seen for the linear test equation (15) with $t_{final} = 1$

in Table 2 and 3, where the resulting number of steps, function calls, Jacobin evaluations, LU decomposition and error at the final time is compiled.

Table 2: Method comparison for the linear test equation (15) with $a = \inf$. All tolerances are set to 10^{-6}.

Method	Lobatto IIIC	Radau IA	Radau IIA
Steps	53	53	52
Function Calls	213	198	213
Jacobian Calls	1	1	1
LU Decompositions	5	5	4
Error	7.82×10^{-10}	1.00×10^{-12}	1.05×10^{-12}

The embedded method with the feedback loop used in this thesis with $\alpha = 0.01$ in (22) results in errors closer in magnitude to the tolerance (10^{-6}), thereby using fewer steps, keeping computational cost down; see Table 3.

Table 3: Method comparison for the linear test equation (15) with $\alpha = 0.01$ and feedback loop. All tolerances are set to 10^{-6}.

Method	Lobatto IIIC	Radau IA	Radau IIA
Steps	8	8	8
Function Calls	42	30	36
Jacobian Calls	1	1	1
LU Decompositions	5	4	4
Error	3.61×10^{-5}	4.14×10^{-8}	4.14×10^{-8}

To see the effects of different a coefficients on the error estimation without the feedback loop on the linear test equation we measure the difference of the local error compared with the error estimation after a single step. To this end, we introduce a *discrepancy*, d_h, between the estimated error and the true local error:

$$d_h = \tau_h^* - \tau_h = ||y_1^* - y_1| - |y(h) - y_1||.$$

In Fig. 3 we see d_h at different step sizes for different a values. We expect to see that the error estimation in this experiment have order 3 since the order of the embedded method is 2 and we are measuring the local error, which typically is one order higher than than global error.

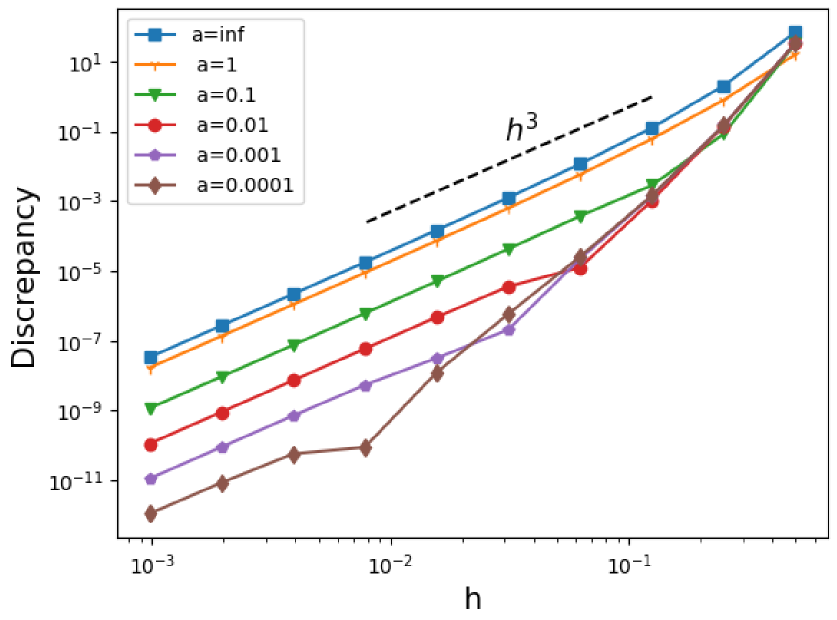


Figure 3: Discrepancy between the error estimate and the local error for the embedded Lobatto IIIC with different a vs step size for (15) after one time step.

We see in Fig. 3 that the error estimation becomes less pessimistic for smaller a as error estimation becomes better for the linear test equation (15) with smaller a. It is worth noting that in the Fig. 3 the discrepancy scales with the local error order of Lobatto IIIC for large h and the expected local error order of the embedded method asymptotically.

With the linear test equation we have observed that implemented Runge-Kutta methods have the expected order compared to the Table 1. We have also shown that the pessimism of the error estimation can be controlled with the novel error estimator described in Section 5.

6.2 Combustion equation

Next, we want to see how the three methods behave and how the adaptive time stepping works in the context of stiff nonlinear problems. The combustion equation is a simple example of such a problem and has controllable stiffness by the initial value. The equation is given by

$$\frac{dy}{dt} = y^2 - y^3, \quad y(0) = 0.01.$$

Here, y denotes the radius of a combustion ball. The differential equation reflects the relationship between the ball's surface area and its volume. A noteworthy feature of this equation is that it possesses a closed-form solution [7]:

$$y(t) = \frac{1}{1 + W(ue^{u-t})}, \quad u = \frac{1}{y(0)} - 1.$$

It is thus straightforward to compute the error. Here, W represents the Lambert W function. The solution $y(t)$ is shown in Fig. 4. Note that the solution converges to $\lim_{t\to\infty} y(t) = 1$ regardless of the initial value.

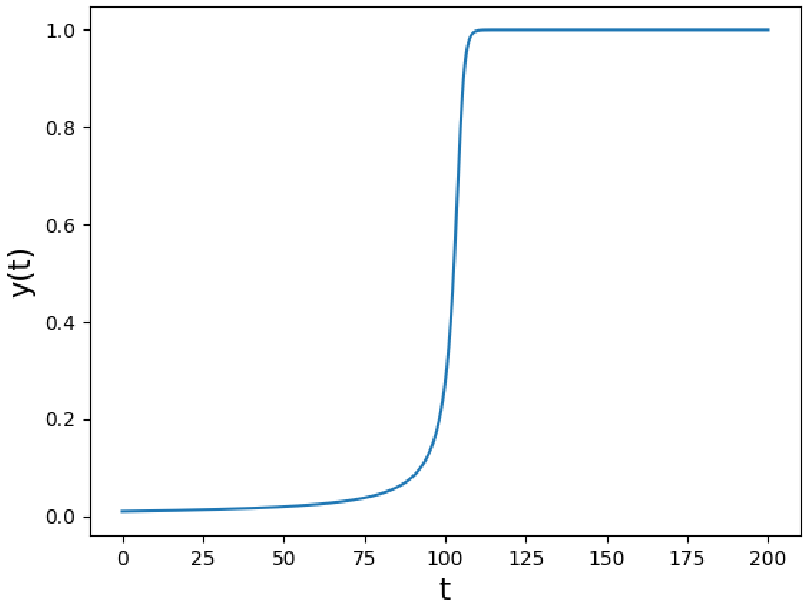
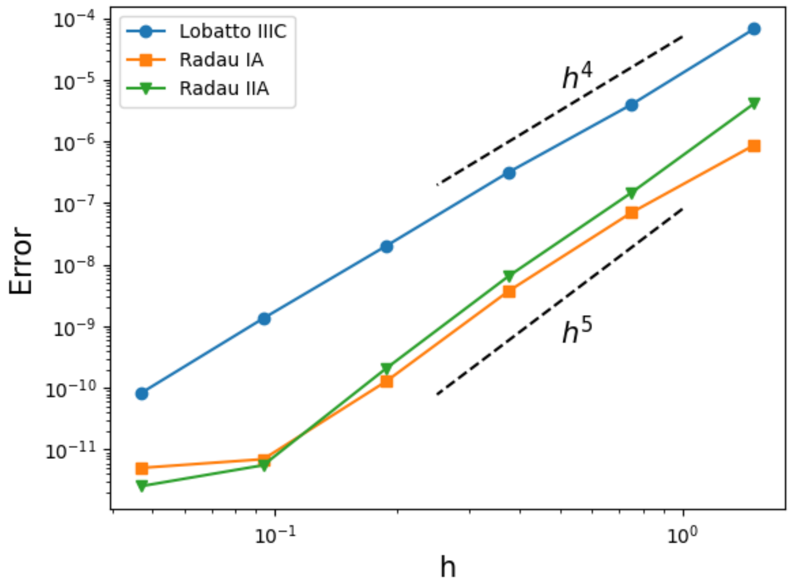


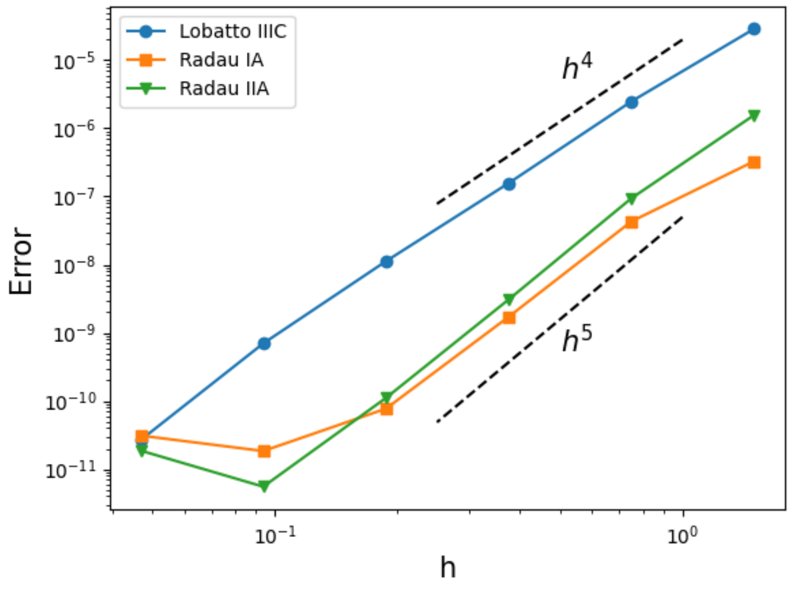
Figure 4: Plot of the combustion equation (3), $y(0) = 0.01$.

In this experiment, the the error is computed at $t_{final}/2$ as this is a transition point and a particular stiff point in the problem however the solver is run until t_{final}. Fig. 5 shows the convergence of the three Runge-Kutta methods for fixed h using two different initial values. In Fig. 5b, the stiffness is more pronounced than in Fig. 5a. For both cases, Lobatto IIIC and Radau IIA has the classical order of convergence seen in Table 1. Radau IA, while having the smallest error, displays a slight order reduction in the non-stiff case. This suggests that the behaviour of Radau IA may be less predictable for stiff problems than for the other methods. A plausible explanation for this observation is that Radau IA is not stiffly accurate.

Next, we solve the combustion equation with adaptive time stepping. We compare the feedback loop dynamic step size control with the standard control



(a) $t = 100$, $y(0) = 0.01$.



(b) $t = 200$, $y(0) = 0.005$.

Figure 5: Error vs step size for the combustion equation (3).

with $a = \inf$. Here, all tolerances are set to 10^{-6} and $\alpha = 0.01$; see (22). We consider the case $y(0) = 0.01$, i.e. the less stiff case and solve using Radau IIA.

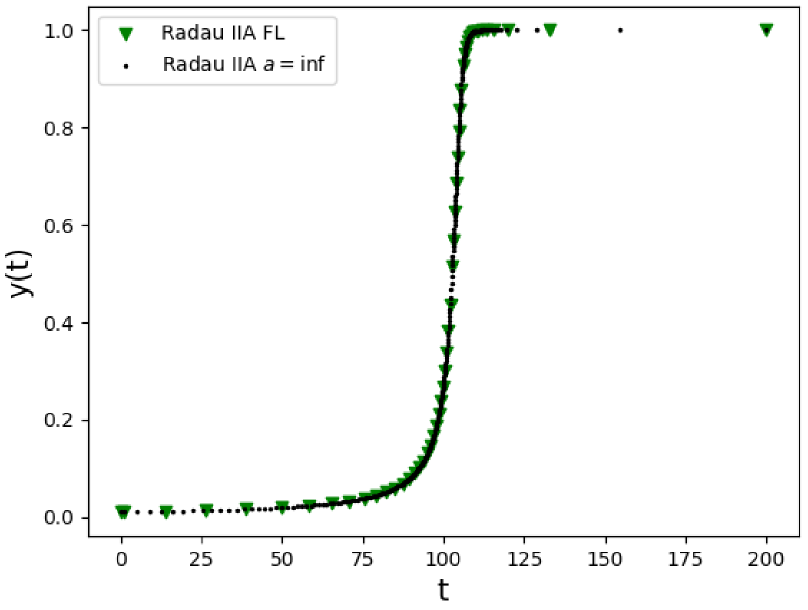


Figure 6: Scatter plot of Combustion equation (3), $y(0) = 0.01$ with adaptive time stepping.

In Fig. 6 we see that the adaptive time stepping have smaller step sizes when the combustion equation undergoes rapid change, and eases afterwards. Note that considerably fewer steps are taken with the feedback loop dynamic step size control than the standard control (FL) compared to the case where $a = \inf$. This is further observed in Tables 4 and 5, which shows run statistics for all three Runge-Kutta methods with the two error control strategies. As for the linear test equation, the errors with the feedback loop control shows errors closer in magnitude to the chosen tolerance.

With this experiment we have seen that the lower order of Lobatto IIIC is penalising its accuracy compared with the other methods. We have also seen that Radau IA displays an unpredictable behaviour for stiff problems, possibly due to its lower stiff order. It is experiencing order reduction. Radau IIA Lobatto IIIC meets the theoretical expectations from Table 1.

Table 4: Method comparison for the combustion equation $y_0 = 0.01$ with adaptive time stepping with feedback loop.

Method	Lobatto IIIC	Radau IA	Radau IIA
Steps	53	51	52
Function Calls	1641	1398	1476
Jacobian Calls	2	3	2
LU Decompositions	45	46	42
Error	4.85×10^{-6}	1.94×10^{-7}	1.31×10^{-7}

Table 5: Method comparison for the combustion equation $y_0 = 0.01$ with adaptive time stepping, $a = \text{inf}$.

Method	Lobatto IIIC	Radau IA	Radau IIA
Steps	299	297	298
Function Calls	4890	4536	4623
Jacobian Calls	2	2	2
LU Decompositions	100	101	100
Error	4.31×10^{-9}	7.30×10^{-11}	3.33×10^{-12}

6.3 Advection diffusion equation

We move to a fluid dynamic problem where space is also discretized. We want to see how the methods behave and how the adaptive time stepping performs with a fluid dynamics problem. As an introductory case, we consider a rotating pulse subject to the advection diffusion equation. A noteworthy feature of this case is that it is linear. The rotating pulse advection diffusion equation is given by [20]

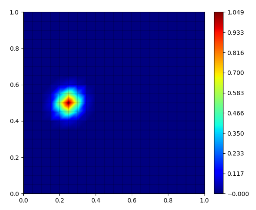
$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) - \nabla \cdot (vc). \tag{24}$$

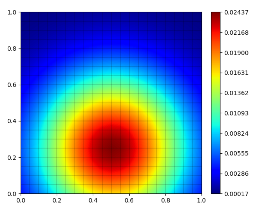
Here, c is the concentration function, D is the diffusion coefficient and v is the velocity vector field. For $v = (-4y, 4x)$, an analytic solution is given by the Gaussian pulse

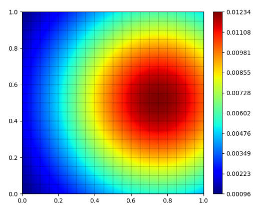
$$c(x, y, t) = \frac{2\sigma^2}{2\sigma^2 + 4Dt} e^{-\frac{(\hat{x}-x_c)^2 + (\hat{y}-y_c)^2}{2\sigma^2+4Dt}},$$

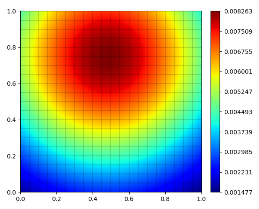
where $\hat{x} = x\cos(4t) + y\sin(4t)$ and $\hat{y} = -x\sin(4t) + y\cos(4t)$. Here, (x_c, y_c) determines the offset from the center of the pulse and σ controls its width. Throughout, we use the values $x_c = 0.25$, $x_y = 0$ and $2\sigma^2 = 0.004$. This solution can be used to extract boundary and initial conditions.

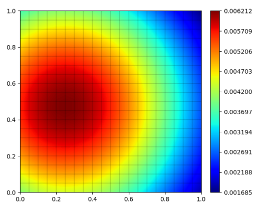
The space discretization utilizes Dune [2], an open-source software framework for solving partial differential equations (PDE). Here, we use the DUNE-FEM module [2], employing a third-order discontinuous Galerkin method. For details about the problem and the implementation, see [15]. Fig. 7 shows the solution on a uniform 20x20 spatial grid at different times.


(a) $t = 0$


(b) $t = 0.4$


(c) $t = 0.8$


(d) $t = 1.2$


(e) $t = 1.6$

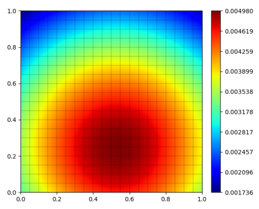
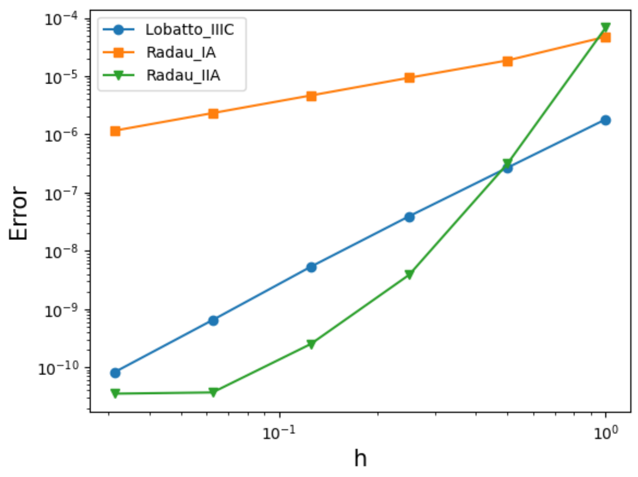

(f) $t = 2$

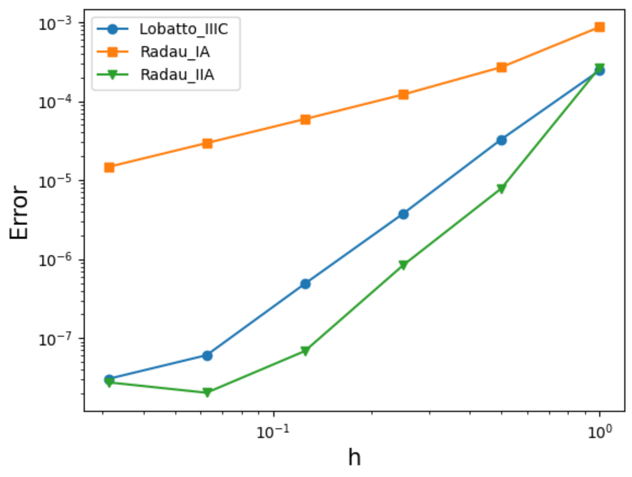
Figure 7: A counterclockwise rotating pulse solution to the advection diffusion equation at different times with grid size 20x20 and end time at 2 with $D = 1$.

The existence of an analytical solution enables direct error computation. This is accomplished by taking the 2-norm of the difference between the numerical and analytical solutions at the final time step. The rotating pulse advection diffusion problem is run with three distinct diffusion factors, $D \in \{1, 0.1, 0.01\}$. The error relative to the step size is depicted in Fig. 8, and the corresponding EOC, given by (23), is listed in Table 6.

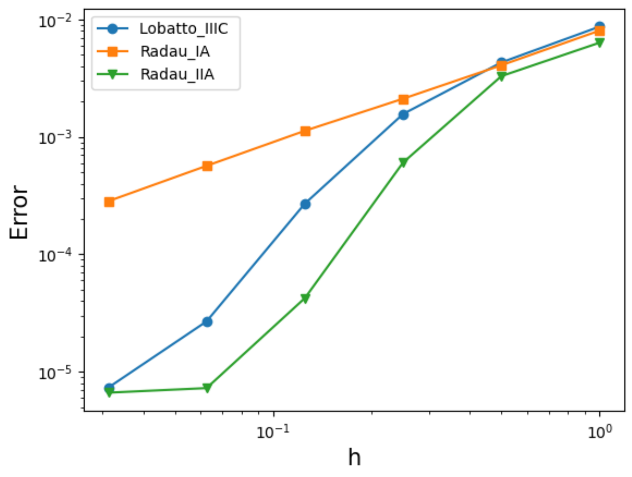
Notably, the methods achieve a lower EOC compared to the theoretical classical order presented in Table 1. This deviation is likely due to the interplay of spatial discretization-induced errors and order reductions from the Runge-Kutta methods. Radau IA underperforms relative to the other methods, and its order is even lower than what the stiff order predicts. For all three values of D, the observed order is asymptotically one. Both Lobatto IIIC and Radau IIA do not meet their theoretical classical orders. With $D = 1$ and $D = 0.1$, Lobatto IIIC appears to converge with an order close to three. For Radau IIA, the case is less clear. Notably, under certain conditions involving a high diffusion coefficient and step size, Radau IIA achieves higher EOC than its classical order, however not in the asymptotic regime. While Lobatto IIIC theoretically should have been significantly inferior to Radau IIA due to its lower order, its performance is only marginally worse in EOC terms. This implies that Lobatto IIIC as well might be well-suited for fluid dynamics challenges.



(a) $D = 1$



(b) $D = 0.1$.



(c) $D = 0.01$

Figure 8: Error vs step size for the advection diffusion equation (24) with grid size 20x20 at $t_{final} = 2$.

Table 6: Experimental order of convergence.

Method / h	D	0.5	0.25	0.125	0.0625	0.03125
Lobatto IIIC	1	2.7610	2.7701	2.8792	3.0442	2.9822
	0.1	2.8952	3.1293	2.9473	3.0051	0.9962
	0.01	1.0138	1.4591	2.5371	3.3319	1.8741
Radau IA	1	1.3510	0.9887	1.0103	1.0064	1.0033
	0.1	1.6883	1.1518	1.0324	1.0097	1.0038
	0.01	0.9817	0.9466	0.9080	0.9914	1.0013
Radau IIA	1	7.7704	6.3376	3.9453	2.7652	0.0700
	0.1	5.0859	3.2338	3.6096	1.7527	-0.4375
	0.01	0.9471	2.4442	3.8383	2.5454	0.1290

Next, we solve the rotating pulse advection diffusion equation with dynamic step size control, with all tolerances set to 10^{-6}. Based on the two previous experiments, the problem is assessed using $\alpha = 0.01$; see (22). The results are shown in Table 7.

We observe that the dynamic step size control increases the number of steps as the problem (24) as the error grows for smaller diffusion coefficients seen in Fig 6. Radau IA performed the poorest among the methods, achieving the lowest accuracy. Lobatto IIIC attaines lower accuracy than Radau IIA, but it also takes fewer steps. However, the error per step is higher than for Radau IIA. Radau IIA achieves the highest accuracy of all the methods.

Table 7: Method Comparison for the Advection-Diffusion equation with feed back loop adaptive time step method

	Diffusion Factor		
Method	1	0.1	0.01
	Lobatto IIIC		
Steps	16	17	33
Function Calls	111	108	174
Jacobian Calls	1	1	15
LU Decompositions	23	23	22
L2-error	4.52×10^{-8}	1.14×10^{-5}	7.16×10^{-5}
	Radau IA		
Steps	17	19	35
Function Calls	117	126	195
Jacobian Calls	1	1	21
LU Decompositions	25	26	20
L2-error	3.02×10^{-5}	6.83×10^{-5}	2.41×10^{-4}
	Radau IIA		
Steps	18	19	35
Function Calls	123	126	195
Jacobian Calls	1	1	21
LU Decompositions	26	25	20
L2-error	2.95×10^{-8}	1.32×10^{-6}	1.177×10^{-5}

6.4 Euler equation

The final problem in this thesis is the isentropic vortex problem with an analytical solution. [19] . It is a non linear fluid dynamic problem closer related to real world scenarios. The isentropic vortex problem is governed by the compressible Euler equations, a system of nonlinear PDEs describing the conservation of mass, momentum and energy of a gas in the absence of internal friction. In 2D, it is given by

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ (\rho E + p)u \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (\rho E + p)v \end{pmatrix}_y = 0.$$

Here ρ is the density, u and v are the velocity components in the horizontal and vertical directions, E is the total energy per unit mass and p is the pressure. The pressure is connected to the remaining variables through the equation of state,

$$p = (\gamma - 1)(\rho E - \frac{\rho}{2}(u^2 + v^2)),$$

where $\gamma = 1.4$ is a constant.

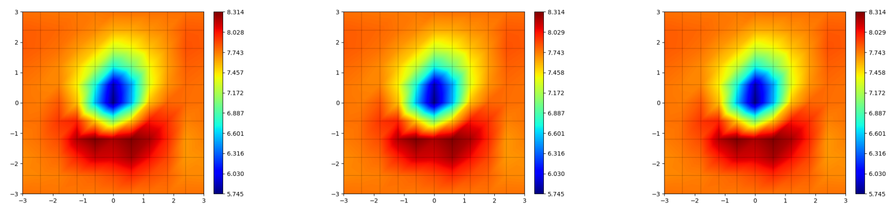
The isentropic vortex problem describes a vortex, initially centered at $(x, y) = (-1, 0)$, moving horizontally with a background flow of unit speed. The initial conditions are given by [20]

$$\begin{aligned} \rho &= \left(1 - S^2(\gamma - 1)M^2 \frac{e^f}{8\pi^2}\right)^{\frac{1}{\gamma-1}}, \\ u &= 1 - S\frac{y}{2\pi}e^{\left(\frac{f^2}{2}\right)}, \\ v &= S\frac{x}{2\pi}e^{\left(\frac{f^2}{2}\right)}, \\ p &= \frac{\rho^\gamma}{\gamma M^2}, \end{aligned}$$

where $S = 5$ is the vortex strength, $M = 0.5$ is the Mach number and $f = 1 - (x + 1)^2 - y^2$.

The space discretization uses the DUNE-FEM [2] module with a Lobatto Legrande spatial discretization on a 10x10 grid as larger grid size does not finish within a day on weaker hardware. In time, the adaptive time stepping is used with all tolerances set to 10^{-6}. This discretization is computationally heavy so no experimental order of convergence will be computed. Instead, a qualitative comparison is made among the three methods.

In Fig. 9 the numerical solution for the density at time $t = 1$ is shown. To the eye, they look indistinguishable. It should be noted that this test case is not particularly stiff, hence all three methods may be expected to perform well. Additionally, since the spatial grid is very coarse, this may account for the dominant contribution to the numerical errors resulting from spatial discretisation error.



(a) Lobatto IIIC (b) Radau IA (c) Radau IIA

Figure 9: Numerical solution for the density of the isentropic vortex problem at time $t = 1$.

The error of the density component is computed at the final time step. The results are shown in Table 8. All methods require the same number of steps. Lobatto IIIC attains the lowest error but requires the most function and Jacobian calls. Radau IIA displays a comparable error with fewer function and Jacobian calls. Meanwhile, Radau IA has the highest error but also the fewest function calls.

Table 8: Method comparison for the isentropic vortex problem

Method	Lobatto IIIC	Radau IA	Radau IIA
Steps	22	22	22
Function Calls	723	645	663
Jacobian Calls	2	1	1
LU Decomposition	13	10	10
L2-error in density	0.03808	0.038223	0.03809

7 Summary and conclusion

In this thesis, we introduced fully implicit Runge-Kutta methods: Lobatto IIIC, Radau IA, and Radau IA. We chose to describe their construction in terms of simplified order conditions. Various stability and order concepts are explored, forming the basis for our expected performance of the methods. A novel adaptive time-stepping approach has also been introduced. The Runge-Kutta methods were comparesd on four test cases, two IVPs and two fluid dynamics PDE problems.

It is for good reason that Radau IIA is the go to method for fully implicit Runge-Kutta applications. It has good order and stability properties for stiff problems. Lobatto IIIC also work well in the context of fluid dynamics problems and has robust performance, however the lower order is penalizing. The order reduction of Radau IA for stiff problems makes it undesirable in some situations.

The stability and order concepts established in Section 4 seem to be valid for initial value problems. However, when the methods are working in conjunction with spatial discretizations, the concepts become more of a guideline. Lobatto IIIC, in the context of fluid dynamic problems, can not easily be written off, and might be a good candidate for particular problems. The novel approach to adaptive time stepping shows good performance for an embedded method, however many aspects of it has been left untouched in this thesis.

8 Further observations and outlook

The bachelor's project has touched on parts that this thesis does not cover. However, these might be of interest to the reader:

The construction of the Runge-Kutta methods in the thesis originated from the simplified order conditions. However, they can equivalently be derived from collocation and discontinuous collocation of polynomials, as is detailed in [11]. The equivalency between collocation Runge-Kutta methods and Galerkin methods [20], while an interesting topic, is not instrumental to the thesis and therefore not elaborated upon.

We have considered the use of Newton-Krylov methods as a replacement for the simplified Newton methods. However, for the problems chosen it was not necessary and proved to be more computationally intensive. Additionally, we also implemented the methods using a full Newton method without various optimizations to determine if the results concurred with the main implementation. This was observed to be the case, modulo larger round-off errors.

Other test problems, such as the Brusselator or the Van der Pol oscillator, were also examined. However, they lacked analytical solutions, making error analysis difficult. In this thesis we tested the methods on one stiff linear PDE and one non-stiff nonlinear PDE. The testing of the methods could also be extended to non linear stiff fluid dynamics problems, which is the intended application of these methods.

The properties and behaviours of the new adaptive time stepping methods needs to be further studied if to be used beyond the few test cases considered herein. While the experiments conducted in the thesis suggests that the method is robust, there is currently no supporting theory.

In this thesis we used an equidistant grid for spatial discretization. The methods could have a different behaviour on an irregular or adaptive spatial discretization. This is relevant in 3D applications as adaptivity can considerably reduce the computational cost of numerically solving PDEs.

References

[1] Christian Andersson, Claus Führer, and Johan Åkesson. Assimulo: A unified framework for ode solvers. *Mathematics and Computers in Simulation*, 116:26–43, 2015.

[2] Peter Bastian, Markus Blatt, Andreas Dedner, Nils-Arne Dreier, Christian Engwer, René Fritze, Carsten Gräser, Christoph Grüninger, Dominic Kempf, Robert Klöfkorn, Mario Ohlberger, and Oliver Sander. The DUNE framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, 81:75–112, 2021. Development and Application of Open-source Software for Problems with Numerical PDEs.

[3] P. D. Boom and D. W. Zingg. High-order implicit time-marching methods based on generalized summation-by-parts operators. *SIAM Journal on Scientific Computing*, 37(6):A2682–A2709, 2015.

[4] Kevin Burrage and J. C. Butcher. Stability criteria for implicit Runge–Kutta methods. *SIAM Journal on Numerical Analysis*, 16(1):46–57, 1979.

[5] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* Wiley, 1987.

[6] J.C. Butcher. A history of Runge-Kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[7] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, 1996.

[8] Michel Crouzeix. Sur la B-stabilité des méthodes de Runge-Kutta. *Numerische Mathematik*, 32(1):75–82, 1979.

[9] Cory V. Frontin, Gage S. Walters, Freddie D. Witherden, Carl W. Lee, David M. Williams, and David L. Darmofal. Foundations of space-time finite element methods: Polytopes, interpolation, and integration. *Applied Numerical Mathematics*, 166:92–113, 2021.

[10] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II.* Springer Series in Computational Mathematics. Springer Berlin, Heidelberg, 2 edition, 1996. Ch. IV.3, IV.5, IV.8 IV.12 IV.15.

[11] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Geometric Numerical Integration.* Springer Series in Computational Mathematics. Springer Berlin, Heidelberg, 2 edition, 2006. Ch. 2.

[12] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem.* Wiley, April 1992. Print version. Ch.5-7.

[13] Lehsten, Edmund Aristid. Implementation of 3 stage Lobatto IIIC into the Assimulo package, 2021. Bachelor Theses.

[14] Viktor Linders, Jan Nordström, and Steven H. Frankel. Properties of Runge-Kutta-summation-by-parts methods. *Journal of Computational Physics*, 419, 2020.

[15] Lund University. Lund University GitLab Repository. https://gitlab.maths.lu.se/dune/spacetimelobattocode/.

[16] Linda R. Petzold, Laurent O. Jay, and Jeng Yen. Numerical solution of highly oscillatory ordinary differential equations. *Acta Numerica*, 6:437–483, 1997.

[17] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125):145 – 162, 1974.

[18] Joachim Rang. Adaptive timestep control for fully implicit Runge-Kutta methods of higher order. 2014. https://api.semanticscholar.org/CorpusID:125163075.

[19] Chi-Wang Shu. *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, pages 325–432. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[20] Lea Miko Versbach, Viktor Linders, Robert Klöfkorn, and Philipp Birken. Theoretical and practical aspects of space-time DG-SEM implementations. *The SMAI Journal of computational mathematics*, 9:61–93, 2023.

Bachelor’s Theses in Mathematical Sciences 2023:K26
ISSN 1654-6229

LUNFNA-4051-2023

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/