,

Towards New High-Order Operator Splitting Time-Integration Methods

A dissertation submitted to the
College of Graduate and Postdoctoral Studies
in partial pulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Jessica Cervi

Permission to Use

In presenting this dissertation in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this dissertation in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my dissertation work or, in their absence, by the Head of the Department or the Dean of the College in which my dissertation work was done. It is understood that any copying or publication or use of this dissertation or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my dissertation.

Disclaimer

Reference in this dissertation to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this dissertation in whole or part should be addressed to:

Head of the Department of Mathematics and Statistics
142 McLean Hall
106 Wiggins Road
University of Saskatchewan
Saskatoon, SK S7N 5E6 Canada

OR
Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

Operator splitting (OS) methods represent a powerful strategy to solve an extensive range of mathematical models in the form of differential equations. They have a long and celebrated history, having been successfully used for well over half a century to provide efficient numerical solutions to challenging problems. In fact, OS methods are often the only viable way to solve many problems in practice.

The simplest, and perhaps, most well-known OS methods are Lie–Trotter–Godunov and the Strang–Marchuk methods. They compute a numerical solution that is first-, and second-order accurate in time, respectively. OS methods can be derived by imposing order conditions using the Campbell–Baker–Hausdorff formula. It follows that, by setting the appropriate order conditions, it is possible to derive OS methods of any desired order. An important observation regarding OS methods with order higher than two is that, according to the Sheng–Suzuki theorem, at least one of their defining coefficients must be negative. Therefore, the time integration with OS methods of order higher than two has not been considered suitable to solve deterministic parabolic problems, because the necessary backward time integration would cause instabilities.

Throughout this thesis, we focus our attention on high-order (i.e., order higher than two) OS methods. We successfully assess the convergence properties of some higher-order OS methods on diffusion-reaction problems describing cardiac electrophysiology and on an advection-diffusion-reaction problem describing chemical combustion. Furthermore, we compare the efficiency performance of higher-order methods to second-order methods. For all the cases considered, we confirm an improved efficiency performance compared to methods of lower order.

Next, we observe how, when using OS and Runge–Kutta type methods to advance the time integration, we can construct a unique extended Butcher tableau with a similar structure to the ones describing Generalized Additive Runge–Kutta (GARK) methods. We define a combination of methods to be OS-GARK methods. We apply linear stability analysis to OS-GARK methods; this allows us to conveniently analyze the stability properties of any combination of OS and Runge–Kutta methods. Doing so, we are able to perform an eigenvalue analysis to understand and improve numerically unstable solutions.

Acknowledgements

First of all, I want to thank my supervisor, Professor Raymond J. Spiteri, for his continuous support and deep knowledge throughout the course of this degree. Most of the work presented in this thesis would not have been possible without his guidance.

It has also been a privilege to have such a distinguished committee. In particular, I want to thank Professor Higueras, Professor Ko, Professor Patrick, and Professor Szmigielski for their valuable comments that helped me improve the quality of this thesis. I also appreciate the help and support I received from the faculty, students, and staff at the Departments of Mathematics and Statistics.

Thanks to my parents, Ombretta and Silvano Cervi, and all my family back home. It has not been easy being so far from home; probably more so for them than for me. Their continuous support for my education, sense of encouragement, and wisdom have helped me more than they will ever be able to understand. I will never be able to express my gratitude with words.

Last, I want to thank you all the friends I have made since I moved here. The list of names would be to long to write here, but you are all in my mind, always. They showed me love, friendship, and support. Some have seen the best and worse of me, but none of them has ever stopped supporting me. They always encouraged me to move forward, *one day at a time*. I consider myself truly lucky and blessed to have them in my life.

Contents

List of Tables

List of Figures

List of Abbreviations

OS	Operator Splitting
IC	Initial Condition
BC	Boundary Condition
PDE	Partial Differential Equation
CBH	Campbell-Baker–Hausdorff
G1	Godunov method
ADI	Alternating Direction Implicit
SM2	Strang–Marchuk method
R3	Ruth method
Y4	Yoshida method
RK	Runge–Kutta
M4	McLachlan method
DS4	Descombes method
S3	Sornborger method
B4	Blanes method
AKS3	Auzinger–Ketchenson–Suzuki method
C3	Chambers third-order method
C4	Chambers fourth-order method
AKS3C	Auzinger–Ketchenson–Suzuki method with complex coefficients
RK	Runge–Kutta
SDIRK	Singly Diagonal Implicit Runge–Kutta method
ARK	Additive Runge–Kutta method
GARK	Generalized Additive Runge–Kutta method
ADR	Advection-Diffusion-Reaction
BE	Backward Euler
FE	Forward Euler
ERK3	Explicit third-order Runge–Kutta method
ERK4	Explicit fourth-order Runge–Kutta method
SDIRK2O3	Third-order SDIRK method with two stages
SDIRK3O4	Fourth-order SDIRK method with three stages

1 Introduction

Operator splitting (OS) methods represent a powerful strategy to solve an exceptionally broad range of mathematical models in the form of differential equations. They have a long and celebrated history, having been successfully used for well over half a century to provide efficient numerical solutions to challenging problems, and they remain an active area of research effort; see, e.g., [Yan71, Mar90, HV03, GMS06, GOY16] and references therein, for thorough expositions of methods and applications. In particular, compelling applications of OS methods come from diverse areas of science and engineering, including direct numerical simulation of particulate and free-surface flows [GPH$^+$01], quantum mechanics [BJM02, Tha08], reactive flows [OB05], real-time simulation of fuel cells [KIFS10], chemotaxis [TSL00, GV02], and cardiac simulations [CS18a, CS18b, STZ16, ZMSS14], to name only a few. In fact, OS methods are often the only viable way to solve many problems in practice.

The simplest, first, and, perhaps most well-known OS methods is the Lie–Trotter–Godunov method [LE70, Tro59, God59]. The Lie–Trotter–Godunov method splits the original problem into two sub-problems, performs an integration on each operator at each time-step, and computes a numerical solution that approximates the exact solution to a degree which is first-order accurate. Another well-known OS method is the one discovered independently by Strang [Str68] and Marchuk [Mar71]. This method was derived based on a symmetrization principle by carrying out two half steps of the Lie–Trotter–Godunov method with reversed sequence. The Strang–Marchuk method computes a numerical solution that is second-order accurate.

Although the Lie–Trotter–Godunov and the Strang–Marchuk methods are the most famous splitting methods, it is possible to derive OS method of any desired order. To do so, it is necessary to impose order conditions to guarantee the proper order of convergence. Such order conditions can be derived by approximating the product of the sub-flows of the sub-problems obtained via OS using the Campbell–Baker–Hausdorff formula to get a series of commutators in powers of the time-step [HLW06]. This way, the Campbell–Baker–Hausdorff formula can be used to derive order conditions to obtain OS methods of any order. One important observation is that, when deriving an OS method of order higher than two, at least one of the defining coefficients that characterize an OS method must be negative. This result was proven independently by Sheng [She89] and Suzuki [Suz91] in 1989 and 1991, respectively, and, later, by Goldman and Kaper [GK96], who showed that each operator must have a negative coefficient.

Both the Lie–Trotter–Godunov and the Strang OS methods do not have negative coefficients. For this reason, these methods are considered stable for solving problems where the differential equation is well-posed for forward time integration. In fact, the presence of negative coefficients translates into a need for backward

time integration, which is unstable for solving deterministic parabolic systems.

Throughout this thesis, we focus our attention on high-order (i.e., order higher than two) OS methods. We start by assessing the convergence properties of some higher-order OS methods on one- and three-dimensional reaction-diffusion problems describing cardiac electrophysiology via the bidomain [SLC06] and monodomain [Tun78] models. In all the problems, expected rates of convergence of high-order OS methods are successfully demonstrated on different examples with a variety of initial conditions and cell models, including the Niederer benchmark problem [NKB$^+$11]. Having established the proper order of convergence of high-order OS methods, we focus our attention on their efficiency performance. Results show that using higher-order OS methods reduces the CPU time needed to complete the simulations by 20% to 40% depending on the problem considered. We successfully conduct similar experiments to verify the convergence and the efficiency of higher-order OS on an advection-diffusion-reaction problem describing chemical combustion [BCP11]. The expected order of convergence and a better efficiency performance are achieved on the combustion problem as well.

We observe that such extended Butcher tableaux have a similar structure to the ones describing Generalized Additive Runge–Kutta (GARK) methods. This observation leads to the conclusion that the combination of OS and Runge–Kutta methods used for the time integration can be written as a GARK method scaled by the coefficients of the OS method chosen. We define these methods to be OS-GARK methods.

We study the stability of the OS-GARK method by applying linear stability analysis. Doing so, we derive a stability function that can be written as the product of the stability functions of the Runge–Kutta methods used in the integration scaled by the coefficients of the OS methods. With this result, we can conveniently analyze the stability of any combination of OS and Runge–Kutta methods used.

Being able to plot the stability region results particularly useful in cases where the solution of a split problem is unstable. In fact, it is sometime possible to understand numerically unstable solutions by obtaining the eigenvalues of the Jacobian and by verifying whether they are captured by stability region or not. We apply this stability theory to a variety of problems with stable and unstable solutions, including the combustion problem and the Brusselator equations [PL68].

1.1 Contributions of the Thesis

1. We assessed the convergence of OS methods with order higher than two on problems describing cardiac electrophysiology in one and three dimensions using a variety of initial conditions and cell models. We demonstrated gains in efficiency between 15% and 30% in terms of CPU time of such high-order methods on the same set of cardiac problems. Similar results were obtained when solving an advection-diffusion-reaction problem describing chemical combustion.

2. We showed how, by choosing an OS method and Runge–Kutta type integrators, it is possible to construct an extended Butcher tableau with the same structure as the extended Butcher tableau

describing GARK methods. It follows that the time integration performed using an OS method together with Runge–Kutta integrators can be written as a GARK method with the coefficients scaled by the coefficients of the OS method. We defined such methods to be OS-GARK methods.

3. We performed a linear stability analysis of OS-GARK methods to show how it is possible to derive a stability function for OS-GARK methods that can be simplified as the product of the stability functions of the integrators scaled by the coefficients of the OS method.

4. We used the presented stability theory to justify stable and unstable behaviours in the solution of various problems, such as the Brusselator problem and a chemical combustion problem.

   Results about the convergence and the gains in efficiency of OS methods with order higher than two can be found in [CS18a]. In [CS18a], we demonstrate that it is possible to obtain stable results from third-order OS methods applied to the bidomain and monodomain models describing cardiac electrophysiology. Furthermore, the authors demonstrate the accompanying gains in efficiency obtained using higher-order OS compared to lower-order methods on a number of illustrative problems.

   Further empirical results about the stability of OS of order four are presented in [CS18b]. In [CS18b], we demonstrate the stability of OS methods of up to order four to solve the bidomain and monodomain models on several examples arising in the field of cardiovascular modeling.

   Finally, empirical results about the efficiency of OS methods of order four were presented in [CS19]. In [CS19], we compare the performance of fourth-order OS methods with real and complex to solve the Niederer problem [NKB$^+$11], a well-defined benchmark in cardiac tissue electrophysiology, as well as a variant with a stiffer cell model.

1.2 Structure of the Thesis

The remainder of this thesis is structured into four chapters. In chapter 2, we present a review of OS methods from a historical point of view. In chapter 3, we focus on the stability of OS methods. We present the state of the art regarding the stability of OS methods, our results about the construction of the extended Butcher tableau and we derive the stability function. In chapter 4, we present numerical experiments demonstrating the convergence, efficiency, and stability properties of high-order OS methods. Finally, in chapter 5, we describe our conclusions and possible directions for future work.

2 Literature Review of Operator Splitting Methods

This chapter provides the background information associated with the main concepts of OS methods. In section 2.1, we present the theoretical framework behind OS methods, including the issue of positivity preservation and the treatment of boundary conditions. In section 2.2, we present a historical review of OS methods from the first low-order methods to the more recent higher-order ones, including methods with complex coefficients.

2.1 Theoretical Background

OS methods have a broad range of applications. This can be understood by the fact that they are particularly useful when discretizing systems that are large, coupled, and non-linear. Simulating such systems typically poses a number of challenges. In order to cope with such challenges, a well-known classic strategy is *divide-and-conquer*. OS methods attempt to decouple the systems under consideration into smaller and simpler sub-systems that can potentially be solved more readily with specialized techniques.

In this chapter, we present a survey of OS methods from a largely historical perspective. The theory is illustrated in terms of a basic 2-additive splitting, but it is easily generalizable to N-additively split systems, and the notation is kept general in order to ease the interpretation to many possible applications.

It is useful to start our discussion about OS methods by presenting their general theoretical framework. Consider the following Cauchy problem

$$\frac{d\mathbf{y}}{dt} = \mathcal{A}(\mathbf{y}) = \mathcal{A}^{[1]}(\mathbf{y}) + \mathcal{A}^{[2]}(\mathbf{y}), \qquad \mathbf{y}(0) = \mathbf{y}_0, \tag{2.1}$$

where the (generally non-linear) operator $\mathcal{A}(\cdot)$ has been additively split into two terms $\mathcal{A}^{[1]}(\cdot)$ and $\mathcal{A}^{[2]}(\cdot)$ and $\mathbf{y}_0$ is the given initial condition (IC). The exact flow of (2.1) can be written as

$$\phi_t(\mathbf{y}_0) = e^{t\mathbf{D}}\mathrm{Id}(\mathbf{y}_0),$$

where $\mathrm{Id}(\mathbf{y}) = \mathbf{y}$ is the identity operator and $\mathbf{D}$ is the Lie derivative of $\mathcal{A}(\cdot)$, defined as

$$\mathbf{D}^{[i]} = \sum_j \mathcal{A}_j^{[i]}(\mathbf{y}) \frac{\partial}{\partial \mathbf{y}_j},$$

where $\mathcal{A}_j$ refers to the "component" j of $\mathcal{A}(\cdot)$; see, e.g., [HLW06] for more details.

Having split the operator $\mathcal{A}$ into two terms, it is possible to view the original problem as split into two separate sub-problems

$$\frac{d\mathbf{y}^{[1]}}{dt} = \mathcal{A}^{[1]}(\mathbf{y}^{[1]}), \qquad \frac{d\mathbf{y}^{[2]}}{dt} = \mathcal{A}^{[2]}(\mathbf{y}^{[2]}), \tag{2.2}$$

that are somehow more desirable to solve. The composition of the associated flows $\phi_t^{[l]},\ l = 1, 2$, of the sub-problems in (2.2) can be written as per the following lemma.

Lemma 2.1.1 (Gröbner 1960 [HLW06]). *Let $\phi_s^{[1]}$ and $\phi_t^{[2]}$ be the flows of the differential equations in* (2.2)*, respectively. For their composition we then have*

$$\left(\phi_t^{[2]} \circ \phi_s^{[1]}\right)(\mathbf{y}_0) = \exp(s\mathbf{D}^{[1]}) \exp(t\mathbf{D}^{[2]})\mathrm{Id}(\mathbf{y}_0), \tag{2.3}$$

where $\mathbf{D}^{[1]}$ and $\mathbf{D}^{[2]}$ are the Lie derivatives of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$, respectively, and where Id defines the identity.

Unless $\mathbf{D}^{[1]}$ and $\mathbf{D}^{[2]}$ commute, the composition $\exp\left(t\mathbf{D}^{[1]}\right)\exp(t\mathbf{D}^{[2]})$ is different from $\exp\left(t(\mathbf{D}^{[1]} + \mathbf{D}^{[2]})\right)$, which represents the exact flow ϕ_t of (2.1). The product of the exponentials in (2.3) can approximate the exact flow by using the celebrated Campbell–Baker–Hausdorff (CBH) formula [HLW06], which reads

$$\exp\left(t\mathbf{D}^{[1]}\right)\exp\left(t\mathbf{D}^{[2]}\right) = \exp\left[t(\mathbf{D}^{[1]} + \mathbf{D}^{[2]}) + \frac{1}{2}t^2\mathbf{D}_{12} + \frac{1}{12}t^3(\mathbf{D}_{112} + \mathbf{D}_{221}) + \dots\right], \tag{2.4}$$

where

$$\mathbf{D}_{kl\dots mn} \doteq [\mathbf{D}^{[k]}, [\mathbf{D}^{[l]}, \cdots [\mathbf{D}^{[m]}, \mathbf{D}^{[n]}] \cdots]],$$

and

$$[\mathbf{D}^{[m]}, \mathbf{D}^{[n]}] \doteq \mathbf{D}^{[m]}\mathbf{D}^{[n]} - \mathbf{D}^{[n]}\mathbf{D}^{[m]}.$$

Throughout this thesis, we focus our attention on the class of *fractional-step methods*; see, e.g., [LeV92]. To understand how to derive methods belonging to this class, we consider an evolution equation in the form of (2.1) and define the s-stage method $\mathcal{S}_{\Delta t}(\mathbf{y})$ with fractional steps $\boldsymbol{\alpha} = \left\{\left(\alpha_k^{[1]}, \alpha_k^{[2]}\right)\right\}_{k=1}^{s}$ in a recursive manner as

$$\mathcal{S}_{\Delta t}(\mathbf{y}) = \mathcal{S}_{\boldsymbol{\alpha}_s \Delta t} \circ \mathcal{S}_{\boldsymbol{\alpha}_{s-1}\Delta t} \circ \cdots \circ \mathcal{S}_{\boldsymbol{\alpha}_1 \Delta t}(\mathbf{y}), \tag{2.5}$$

so that

$$\mathcal{S}_{\Delta t} \approx \phi_{\Delta t} = \exp\left(\Delta t\left(\mathbf{D}^{[1]} + \mathbf{D}^{[2]}\right)\right)\mathrm{Id}(\mathbf{y}),$$

For convenience of notation, we define a sub-step of the OS method $(\boldsymbol{\alpha}_k \Delta t)$ as

$$(\boldsymbol{\alpha}_k \Delta t) \doteq e^{\alpha_k^{[2]}\Delta t\mathbf{D}^{[2]}} e^{\alpha_k^{[1]}\Delta t\mathbf{D}^{[1]}}, \quad k = 1, 2, \dots, s, \tag{2.6}$$

where, for coefficients $\alpha_k^{[l]}$, the index l runs over the operators and the index k runs over the stages of the method.

The solution information from the end point of each sub-flow with fractional step $\alpha_k^{[l]}$ is used as the initial condition for the start of the subsequent sub-flow. An example of a two-stage, two-additive OS method with positive sub-flows $\alpha_k^{[l]}\Delta t$, $\sum_{k=1}^{2}\alpha_k^{[l]} = 1$, $k, l = 1, 2$, to advance the numerical solution from time t_n to time $t_{n+1} = t_n + \sum_{k=1}^{2}\alpha_k^{[l]} = t_n + \Delta t,\ l = 1, 2$ is depicted in Figure 2.1. The solid lines denote sub-flows, and the dashed lines denote the transfer of solution information between sub-flows.

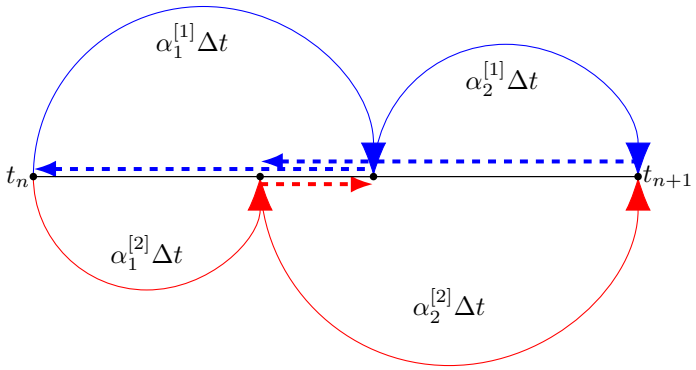

Figure 2.1: Schematic of a two-stage, 2-additive OS method. The solid lines denote the sub-flows, and the dashed lines denote the transfer of solution information between sub-flows.

We further denote the reversal of the default order in which the differential operators $\mathbf{D}^{[i]}$, $i = 1, 2$, are applied by

$$(\check{\boldsymbol{\alpha}}_k \Delta t) \doteq e^{\Delta t \mathbf{D}_k^{[1]}} e^{\Delta t \mathbf{D}_k^{[2]}}, \qquad k = 1, 2..., s. \tag{2.7}$$

Defining *abscissae*

$$t_{n,k+1}^{[l]} := t_{n,k}^{[l]} + \alpha_k^{[l]} \Delta t = t_n + \left(\Delta t \sum_{j=1}^{k} \alpha_j^{[l]} \right), \quad l = 1, 2, \quad k = 1, 2, \ldots, s, \tag{2.8}$$

for appropriate $\alpha_k^{[l]}$, an algorithm for a generic OS method applied to (2.2) is given in Algorithm 1, where it is assumed the sub-system with $\mathcal{A}^{[1]}$ is solved first. For the sub-steps that require solving the sub-system with the operator $\mathcal{A}^{[2]}$ first, the order of the solves and the associated indexing and temporary variables are reversed.

Algorithm 1 One step of an s-stage 2-additive OS for a generic Cauchy problem (2.1) with abscissae given by (2.8)

Require: $\alpha_0^{[l]} = 0$, $l = 1, 2$; $t = t_n$; $\mathbf{y}_n^{[1]} \approx \mathbf{y}(t_n)$.

1: **for** $k = 1$ to s **do**
2: Solve the sub-system
3: $\dot{\mathbf{y}}^{[1]} = \mathcal{A}^{[1]}(\mathbf{y}^{[1]})$ for $t_{n,k}^{[1]} \le t \le t_{n,k+1}^{[1]}$
4: subject to the local IC $\mathbf{y}^{[1]}(t_{n,k}^{[1]}) = \mathbf{y}_{n,k}^{[2]}$ and the prescribed BCs to obtain $\mathbf{y}_{n,k+1}^{[1]}$.
5: Solve the sub-system
6: $\dot{\mathbf{y}}^{[2]} = \mathcal{A}^{[2]}(\mathbf{y}^{[2]})$ for $t_{n,k}^{[2]} \le t \le t_{n,k+1}^{[2]}$
7: subject to the local IC $\mathbf{y}^{[2]}(t_{n,k}^{[2]}) = \mathbf{y}_{n,k+1}^{[1]}$ and the prescribed BCs to obtain $\mathbf{y}_{n,k+1}^{[2]}$.
8: **end for**
9: **return** $\mathbf{y}_{n+1} = \mathbf{y}_{n,s}^{[2]} \approx \mathbf{y}(t_{n+1})$.

The two sub-systems solved in Algorithm 1 can be thought of as advancing the first sub-flow $\mathcal{S}_{\boldsymbol{\alpha}_1 \Delta t}$ in (2.5), where we note that components of the sub-flows may not be coincident in time; e.g., $\alpha_1^{[2]} \neq \alpha_1^{[1]}$ in

Figure 2.1. The subsequent sub-flows in (2.5) can be executed in a similar fashion to complete the integration to time $t = t_{n+1}$. Of course, to ensure that the overall OS method attains the desired order p of accuracy in time, we need to approximate the flows of both sub-systems to at least order p.

2.1.1 Positivity Preservation

In this section, we describe some results regarding positivity preservation on certain classes of problems solved using OS.

Although the analysis and the treatment of positivity preservation is not studied in any other section of the thesis, positivity preservation is an important feature in numerous applications, specifically those that come from computational biology or chemistry, where the state variables may represent population sizes, densities, absolute temperatures, or concentrations; see, e.g. [EK01, Mur03, Dur10].

Many results are centred on the theme that the composition of positive operators is a positive operator [Kos80]. Therefore, exponential operator splitting methods preserve positivity provided that the defining sub-flows do so.

An example based on this theme is given in [HKO12]. In this study, the authors consider an inhomogeneous semi-linear parabolic problem of the form

$$\frac{\partial \mathbf{y}}{\partial t} = \mathcal{A}\mathbf{y} + f(\mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0. \tag{2.9}$$

Positivity is lost when trying to solve (2.9) by using unconditionally positive Runge–Kutta or multistep methods (see, [San01, BC78]) of order $p > 1$. As explained in [HKO12], the situation improves when using exponential Runge–Kutta or exponential multistep methods. In fact, positivity is lost when using methods of order $p > 2$ belonging to these classes. However, none of the these methods preserves positivity for semi-linear parabolic problem. This issue can be overcome by using exponential operator splitting methods (see [HKO12, RS11]). In fact, OS methods are used to make direct use of the split sub-flows to help preserve properties of the exact flow under numerical discretization because they are constructed to preserve the properties of the exact flow. An immediate consequence of this simple observation is the following result.

Proposition 2.1.2 (Hansen 2012, [HKO12])**.** *If the semigroups $e^{-t\mathcal{A}}$ and e^{-tf} generated by $\mathcal{A}$ and f are positive and solved exactly , then the (combined) (see e.g.,[HKO12]) Strang splitting* (2.24) *preserves the positivity of the numerical flow.*

A stronger result to preserve positivity when using high-order OS was presented in [RS11], where up to fourth-order accuracy was achieved by splitting the Vlasov–Poisson system using second- and fourth-order OS methods defined by Strang and Yoshida (see sections 2.2.2 and 2.2.3 of this thesis), and by solving the sub-flows using a semi-Lagrangian method of the appropriate order and such that the overall system preserves positivity after each time step.

Despite these results, a general theory about preserving positivity using OS methods is lacking. We

encourage the interested reader to refer to the literature (see e.g., [HKO12, San01, BC78, RS11] and references therein) when dealing with positivity preservation on particular problems.

2.1.2 Treatment of Boundary Conditions

In many applications, the solution of Cauchy problem (2.1) is the result of a method-of-lines discretization of a partial differential equation (PDE). When applying OS methods in such cases, it is important to consider the treatment of boundary conditions (BCs). In particular, it is shown in [CGA94] that the conventional way of imposing BCs at the intermediate stages by using the prescribed value of the boundary data inevitably reduces the accuracy to first order, independently of the order of accuracy of the difference operator. In [CGAD95], Carpenter and collaborators presented the *simultaneous approximation term* (SAT) as a general technique to impose correct intermediate boundary values for linear PDEs to attain the full order of accuracy. The SAT technique solves a linear combination of the BCs and the differential equations near the boundary. However, the approach presented in [CGAD95] was not suitable for non-linear PDEs solved with integrators of order higher than three. The solution to the non-linear case was given in [AGC96], where Abarbanel, Gottlieb, and Carpenter presented a variation of the linear SAT method. The idea behind the solution for the non-linear case comes from the fact that the linear SAT yields intermediate values for the boundary terms that have the required degree of accuracy. By using a linear combination of such terms computed with the linear SAT, the authors were able to recover a scheme that attained the required degree of accuracy for non-linear PDEs. We refer the interested reader to [AGC96] for more details and to [DRFHZ14] for a comprehensive review about SAT.

In [HV03], Hundsdorfer and Verwer point out that, when solving PDEs using an OS method with order $p > 2$, the unavoidable negative time-steps (see subsection 2.2.3) could limit the effectiveness of higher-order OS methods. In fact, as pointed out also in [Sor07], diffusion or reaction terms could lead to instabilities when using backward time integration. For this reason, the use of higher-order methods has been focused on conservative problems where BCs are not relevant such as the Schrödinger equation and Hamiltonian systems [HV03]. In general, difficulties with OS methods may arise for PDE problems where BCs are physical conditions for the entire problem and BCs for the sub-steps are missing. So, when taking a step using OS, these physical conditions may not be obvious, and one may have to reconstruct them for the specific splitting in consideration; see [HV03] for more details.

Other problems related to OS methods and the treatment of BCs arise when dealing with advection problems with Dirichlet BCs at the inflow boundary [HV03]. In particular, when using higher-order OS methods, we necessarily need to take backwards steps (see section 2.2), which require backward-in-time boundary values at the outflow of the boundary as well [HV03].

More recently, various authors came across the issue of order reduction when applying OS methods especially in the context of diffusion-reaction PDEs. In 2015, Einkemmer and Ostermann [EO15] considered the case where the BCs on the domain Ω were time-dependent and of inhomogeneous Dirichlet type. In those

cases, the OS procedure does not retain the correct order. To overcome this problem, the authors proposed to rewrite the problem in such a way that homogeneous boundary conditions can be imposed (see [EO15]).

In the following year, the same authors [EO16] proposed correction for a semi-linear diffusion-reaction problem with oblique Dirichlet BCs. The approach for this case is similar to the one adopted in [EO15], where the BCs need to be modified by a time-dependent boundary condition constructed such as to avoid order reduction (see [EO16]). It is useful to point out that a cheaper construction of this correction was presented in [BV19].

Despite these results, to the best of our knowledge, a general theory and analysis of BCs for splitting methods is lacking. As a general rule, the treatment of the boundaries should be as consistent as possible with the method used on the interior of the domain [HV03].

2.2 A Brief History of Operator Splitting Methods

According to [CHMM78], the first operator splitting method on record was introduced in 1875 by Lie [LE70] (reprinted 1888) in order to solve (2.1) in the case where $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are two $m \times m$ time-dependent matrices denoted now with $\bar{\mathcal{A}}^{[1]}$ and $\bar{\mathcal{A}}^{[2]}$, respectively. From the relation,

$$e^{(\bar{\mathcal{A}}^{[2]}+\bar{\mathcal{A}}^{[1]})t} = \lim_{n\to\infty} \left(e^{\bar{\mathcal{A}}^{[2]}\frac{t}{n}} e^{\bar{\mathcal{A}}^{[1]}\frac{t}{n}} \right)^n, \tag{2.10}$$

Lie suggested the following method to find an approximate solution of (2.1)

$$\mathbf{y}_{n+1} = e^{\Delta t \bar{\mathcal{A}}^{[2]}} e^{\Delta t \bar{\mathcal{A}}^{[1]}} \mathbf{y}_n, \quad \mathbf{y}(0) = \mathbf{y}_0, \quad n = 0, 1, 2, \ldots. \tag{2.11}$$

In 1958, Trotter made the first important contribution to (2.11) by extending the theory to the case where $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are unbounded operators as follows: [Tro59]

Theorem 2.2.1 (Trotter)**.** *Let $\mathcal{H}$ be a Hilbert space, not necessarily separable. If $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are self-adjoint operators in $\mathcal{H}$ such that $\mathcal{A}^{[1]} + \mathcal{A}^{[2]}$ is also self-adjoint in $\mathcal{H}$, then* (2.10) *is satisfied for each $t \in \mathbb{R}$ and for each $\mathbf{y} \in \mathcal{H}$.*

Especially for high-dimensional problems, computing the exponential of a matrix can be a non-trivial operation. Therefore, a matrix-free version of this method is usually used. Such equivalent version of the method in (2.11) was introduced in 1959 by Godunov [God59] while he was working with non-linear systems describing gas dynamics.

Godunov obtained a variation of the upwind method in which the local characteristics are not provided by diagonalizing the Jacobian matrix, but rather by solving a series of Riemann problems forward in time. In the Godunov method, the evolution from time t_n to time $t_{n+1} = t_n + \Delta t$ is computed by first assuming a piecewise constant distribution of the data over the spatial grid. Then, solutions can be found by solving a sequence of Riemann problems obtained by separating the contributions from the left and right states. The

numerical solution to the problem is then obtained by piecing together the contributions from the different solutions to the Riemann problems [LeV92].

The method, commonly named after *Lie–Trotter* [Tro59] or *Godunov* [God59], can be written in the following way. Starting from (2.2), first solve

$$\frac{d\mathbf{y}^{[1]}}{dt} = \mathcal{A}^{[1]}\mathbf{y}^{[1]}, \qquad \mathbf{y}^{[1]}(0) = \mathbf{y}_0,$$

for $t \in [0, \Delta t]$ to yield an approximate solution $\mathbf{y}^{[1]}_{\Delta t}$. Next, use $\mathbf{y}^{[1]}_{\Delta t}$ to compute

$$\frac{d\mathbf{y}^{[2]}}{dt} = \mathcal{A}^{[2]}\mathbf{y}^{[2]}, \qquad \mathbf{y}^{[2]}(0) = \mathbf{y}^{[1]}_{\Delta t},$$

for $t \in [0, \Delta t]$ to obtain $\mathbf{y}^{[2]}_{\Delta t}$. According to the notation introduced in (2.6), the Lie–Trotter–Godunov method can be written as

$$(\boldsymbol{\alpha}_1 \Delta t),$$

with $\boldsymbol{\alpha}_1 = \left(\alpha_1^{[1]}, \alpha_1^{[2]}\right) = (1, 1)$. A schematic of the method is displayed in Figure 2.2.

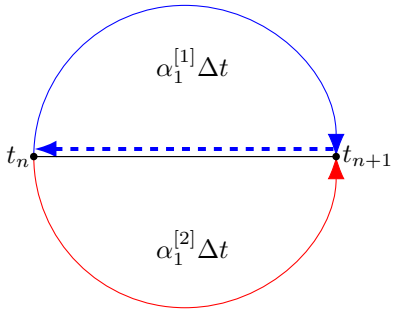

Figure 2.2: Schematic of the Lie–Trotter–Godunov method. The blue solid line denotes the flow of the first sub-system, the dashed line denotes the transfer of solution information between sub-flows, and the red line denotes the flow of the second sub-system.

The Lie–Trotter–Godunov method is perhaps the simplest and most well-known OS method. It computes an approximation to the exact solution of (2.1) that is first-order accurate in the sense that the local splitting error $\mathcal{L}$ is

$$\mathcal{L} = \|\mathbf{y}(\Delta t) - \mathbf{y}^{[2]}_{\Delta t}\| = \mathcal{O}\left((\Delta t)^2\right).$$

This result can be derived by comparing the Taylor series of the exact solution $\mathbf{y}(t)$ with the approximate solution $\mathbf{y}^{[2]}_{\Delta t}$ obtained via OS. We have

$$\mathbf{y}(\Delta t) = \mathbf{y}_0 + \Delta t \frac{d\mathbf{y}}{dt}\bigg|_{t=0} + \frac{(\Delta t)^2}{2}\frac{d^2\mathbf{y}}{dt^2}\bigg|_{t=0} + \mathcal{O}\left((\Delta t)^3\right).$$

From (2.1), by applying direct differentiation k times, we have

$$\frac{d^k\mathbf{y}}{dt^k} = \left(\mathcal{A}^{[1]} + \mathcal{A}^{[2]}\right)^k \mathbf{y},$$

which, when substituted into the above Taylor series, gives

$$\mathbf{y}(\Delta t) = \mathbf{y}_0 + \Delta t \left(\mathcal{A}^{[1]} + \mathcal{A}^{[2]}\right)\mathbf{y}_0 + \frac{(\Delta t)^2}{2}\left(\mathcal{A}^{[1]} + \mathcal{A}^{[2]}\right)^2 \mathbf{y}_0 + \mathcal{O}\left((\Delta t)^3\right).$$

The expansions of the approximate solutions $\mathbf{y}^{[1]}(\Delta t)$ and $\mathbf{y}^{[2]}(\Delta t)$ are, respectively

$$\mathbf{y}^{[1]}(\Delta t) = \mathbf{y}_0 + \Delta t \mathcal{A}^{[1]}\mathbf{y}_0 + \frac{(\Delta t)^2}{2}\left(\mathcal{A}^{[1]}\right)^2 \mathbf{y}_0 + \mathcal{O}\left((\Delta t)^3\right),$$

and

$$\mathbf{y}^{[2]}(\Delta t) = \mathbf{y}^{[1]}_{\Delta t} + \Delta t \mathcal{A}^{[2]}\mathbf{y}^{[1]}_{\Delta t} + \frac{(\Delta t)^2}{2}\left(\mathcal{A}^{[2]}\right)^2 \mathbf{y}^{[1]}_{\Delta t} + \mathcal{O}\left((\Delta t)^3\right),$$

respectively. Inserting the series expansion for $\mathbf{y}^{[1]}(\Delta t)$ into the expansion of $\mathbf{y}^{[2]}(\Delta t)$, the local splitting error becomes

$$\mathbf{y}(\Delta t) - \mathbf{y}^{[2]}(\Delta t) = \frac{(\Delta t)^2}{2}\left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\mathbf{y}_0 + \mathcal{O}\left((\Delta t)^3\right),$$

where $[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}] = \mathcal{A}^{[1]}\mathcal{A}^{[2]} - \mathcal{A}^{[2]}\mathcal{A}^{[1]}$ is, generally, a non-vanishing commutator. Because the error after one time step is proportional to $(\Delta t)^2$, the Lie–Trotter–Godunov OS method is first-order accurate. From now onwards, we denote the Lie–Trotter–Godunov scheme simply with G1.

Curiously enough, Godunov came up with his method by critically analyzing some suspicious conjectures in a report by Zhukov [God99] that was made available to him. Later on, Godunov discovered that Lax published a much more robust version of Zhukov's report a year before his thesis defense, but he did not read it until after. Godunov admitted that had he read Lax's paper a year earlier, the "Godunov Method" would have never been created [God99].

2.2.1 Alternating Implicit Direction Methods

Other methods that incorporate the OS paradigm are the so-called *Alternating Direction Implicit* (ADI) type methods. According to Usadi and Dawson [UD06], ADI methods were first introduced by Rachford, Douglas, and Peaceman in their founding publications [PR55, DR56] to solve parabolic partial differential equations in two spatial dimensions. ADI methods are *predictor-corrector*-type methods, where part of the difference operator is computed in an initial (predictor) step and another part is computed in a final (correction) step. In this approach, each individual operator employs a simple tridiagonal matrix. For this reason, ADI methods are memory-efficient, easy to parallelize, and they have been used in many applications such as astrophysics [SN92], bioengineering [SWL$^+$15], and economics [IT08].

The first variant of ADI methods that we describe here is the *Peaceman–Rachford* method. The predictor step consists of solving (2.1) for a time step $\Delta t/2$ using *forward Euler* for $\mathcal{A}^{[2]}$ and *backward Euler* for $\mathcal{A}^{[1]}$, i.e., solve

$$\mathbf{y}_{\Delta t/2} = \left(\mathrm{I} - \frac{\Delta t}{2}\mathcal{A}^{[1]}\right)^{-1}\left(\mathrm{I} + \frac{\Delta t}{2}\mathcal{A}^{[2]}\right)\mathbf{y}_0 \tag{2.12}$$

for $\mathbf{y}_{\Delta t/2}$, where I is the identity. Then, the roles of the operators are reversed and the corrector step completes the solution process for a time step by using *forward Euler* for $\mathcal{A}^{[1]}$ and *backward Euler* for $\mathcal{A}^{[2]}$, thus solving

$$\mathbf{y}_{\Delta t} = \left(\mathrm{I} - \frac{\Delta t}{2}\mathcal{A}^{[2]}\right)^{-1}\left(\mathrm{I} + \frac{\Delta t}{2}\mathcal{A}^{[1]}\right)\mathbf{y}_{\Delta t/2} \tag{2.13}$$

for $\mathbf{y}_{\Delta t}$. Combining (2.12) and (2.13), the Peaceman–Rachford method can be written as an OS scheme $\mathcal{S}^{\mathrm{PR}}_{\Delta t}$ as

$$\mathcal{S}^{\mathrm{PR}}_{\Delta t} = \left(\mathrm{I} - \frac{\Delta t}{2}\mathcal{A}^{[2]}\right)^{-1}\left(\mathrm{I} + \frac{\Delta t}{2}\mathcal{A}^{[1]}\right)\left(\mathrm{I} - \frac{\Delta t}{2}\mathcal{A}^{[1]}\right)^{-1}\left(\mathrm{I} + \frac{\Delta t}{2}\mathcal{A}^{[2]}\right)\mathbf{y}_0.$$

In general, the Peaceman–Rachford method is first-order accurate. However, when $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are linear and commute, this method becomes second-order accurate [GOY16].

An alternative to the Peaceman–Rachford method is the *Douglas–Rachford* method [DR56]. With an approximation $\mathbf{y}_0$ being known, the prediction step of the Douglas–Rachford method solves

$$\hat{\mathbf{y}}_{\Delta t} = \left(\mathrm{I} - \Delta t\mathcal{A}^{[1]}\right)^{-1}\left(\mathrm{I} + \Delta t\mathcal{A}^{[2]}\right)\mathbf{y}_0 \tag{2.14}$$

for $\hat{\mathbf{y}}_{\Delta t}$ Next, the correction step completes the integration process solving

$$\mathbf{y}_{\Delta t} = \left(\mathrm{I} - \Delta t\mathcal{A}^{[2]}\right)^{-1}\left(\mathbf{y}_0 + \Delta t\mathcal{A}^{[1]}(\hat{\mathbf{y}}_{\Delta t})\right) \tag{2.15}$$

for $\mathbf{y}_{\Delta t}$. Combining (2.14) and (2.15), the Douglas–Rachford method can be written as an OS scheme $\mathcal{S}^{\mathrm{DR}}_{\Delta t}$ as

$$\mathcal{S}^{\mathrm{DR}}_{\Delta t} = \left(\mathrm{I} - \Delta t\mathcal{A}^{[2]}\right)^{-1}\left[\mathbf{y}_0 + \Delta t\mathcal{A}^{[1]}\left(\left(\mathrm{I} - \Delta t\mathcal{A}^{[1]}\right)^{-1}\left(\mathrm{I} + \Delta t\mathcal{A}^{[2]}\right)\mathbf{y}_0\right)\right].$$

The Douglas–Rachford method, even in the case where the operators $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are linear and commute, is, at best, first-order accurate [LM79]. However, to improve accuracy, a variant of the method based on the second-order Crank–Nicolson method has been proposed in [DK01]. This alternative method, commonly referred to as *Douglas–Kim* method solves

$$\hat{\mathbf{y}}_{\Delta t} = \mathbf{y}_0 + \Delta t\mathcal{A}^{[1]}\left(\frac{\hat{\mathbf{y}}_{\Delta t} + \mathbf{y}_0}{2}\right) + \Delta t\mathcal{A}^{[2]}\left(\mathbf{y}_0\right). \tag{2.16}$$

Next, it completes the time-step by solving

$$\mathbf{y}_{\Delta t} = \mathbf{y}_0 + \Delta t\mathcal{A}^{[1]}\left(\frac{\hat{\mathbf{y}}_{\Delta t} + \mathbf{y}_0}{2}\right) + \Delta t\mathcal{A}^{[2]}\left(\frac{\mathbf{y}_{\Delta t} + \mathbf{y}_0}{2}\right). \tag{2.17}$$

Although the Douglas–Kim method is more accurate than other ADI methods in the sense that is second order accurate, it is also less stable compared to the classical ADI methods [UD06].

Both the Peaceman–Rachford and Douglas–Rachford methods are generalizable to the case where $\mathcal{A}$ is split into more than two operators. The details of such generalization go beyond the scope of this thesis, and, therefore, they are omitted here. Details can be found in [GW96] and chapter 2 of [GOY16], respectively.

We remark that, unlike in the Peaceman–Rachford method, the roles played by operators $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ in the Douglas–Rachford method are not interchangeable. This fact is confirmed by numerical experiments where, for the same Δt, the convergence rate depends on the choice of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ (see chapter 2 in [GOY16]). As a general rule, it is desirable to take $\mathcal{A}^{[2]}$ as the operator with the best continuity and monotonicity properties [GOY16, Wac13], additionally, many numerical experiments confirm that the Douglas–Rachford method is more robust and faster when solving problems where one of the operators is *non-smooth* [GOY16, Wac13]. This behaviour is consistent with the fact that the Douglas–Rachford method has better stability

properties than the Peaceman–Rachford method (see [GOY16] for more details). Like the Lie–Trotter–Godunov method, both the classic ADI methods are also part of the family of *additive operator splitting methods* [GOY16].

2.2.2 Second-Order Operator Splitting Methods

In some situations, first-order operator splitting methods may not be efficient enough to achieve a sufficiently accurate numerical solution. Accordingly, higher-order methods were sought, and second-order methods were proposed in the latter half of the twentieth century.

The first second-order operator splitting method that we consider was presented by Lax and Wendroff in 1964 [LW64] to solve the two-dimensional problem

$$\frac{\partial \mathbf{y}}{\partial t} = \bar{\mathcal{A}}^{[1]} \frac{\partial \mathbf{y}}{\partial x_1} + \bar{\mathcal{A}}^{[2]} \frac{\partial \mathbf{y}}{\partial x_2}, \qquad \mathbf{y}(0) = \mathbf{y}_0, \tag{2.18}$$

where t, x_1, and x_2 denote the derivative with respect to time and along spatial directions x_1 and x_2, respectively, and $\bar{\mathcal{A}}^{[1]}$ and $\bar{\mathcal{A}}^{[2]}$ are symmetric, constant matrices that do not necessarily commute. To advance the time integration, the authors chose to use a difference method of the form

$$\mathbf{y}_{\Delta t} = S^{\mathrm{LW2}}_{\Delta t} \mathbf{y}_0, \tag{2.19}$$

where $S^{\mathrm{LW2}}_{\Delta t}$ is a linear difference operator obtained from the Taylor series in time (see chapter 3 in [Wan18])

$$\mathbf{y}_{\Delta t} = \mathbf{y}_0 + \Delta t \left(\frac{d\mathbf{y}}{dt} \right)\bigg|_{t=0} + \frac{\Delta t^2}{2} \left(\frac{d^2\mathbf{y}}{dt^2} \right)\bigg|_{t=0} + \mathcal{O}\left((\Delta t)^2\right). \tag{2.20}$$

By rewriting the time derivatives using (2.18), we can write

$$\frac{\partial^2 \mathbf{y}}{\partial t^2} = \bar{\mathcal{A}}^{[1]} \frac{\partial^2 \mathbf{y}}{\partial x_1 \partial t} + \bar{\mathcal{A}}^{[2]} \frac{\partial^2 \mathbf{y}}{\partial x_2 \partial t} = \bar{\mathcal{A}}^{[1]} \frac{\partial^2 \mathbf{y}}{\partial t \partial x_1} + \bar{\mathcal{A}}^{[2]} \frac{\partial^2 \mathbf{y}}{\partial t \partial x_2}. \tag{2.21}$$

Inserting (2.18) and (2.21) into (2.20), and retaining the $\mathcal{O}\left((\Delta t)^2\right)$ term, (2.20) becomes

$$\begin{aligned} \mathbf{y}_{\Delta t} = \mathbf{y}_0 &+ \Delta t \left(\bar{\mathcal{A}}^{[1]} \frac{\partial \mathbf{y_0}}{\partial x_1} + \bar{\mathcal{A}}^{[2]} \frac{\partial \mathbf{y_0}}{\partial x_2} \right) \\ &+ \frac{(\Delta t)^2}{2} \left(\bar{\mathcal{A}}^{[1]^2} \frac{\partial^2 \mathbf{y_0}}{\partial x_1^2} + (\bar{\mathcal{A}}^{[1]} \bar{\mathcal{A}}^{[2]} + \bar{\mathcal{A}}^{[2]} \bar{\mathcal{A}}^{[1]}) \frac{\partial^2 \mathbf{y_0}}{\partial x_1 \partial x_2} + \bar{\mathcal{A}}^{[2]^2} \frac{\partial^2 \mathbf{y_0}}{\partial x_2^2} \right). \end{aligned}$$

By approximating the spatial derivatives of $\mathbf{y}_0$ using symmetric first- and second-order centred finite differences, the Lax–Wendroff method can be written as

$$\begin{aligned} \mathbf{y}_{\Delta t_{ij}} = \mathbf{y}_{0_{ij}} &+ \frac{\Delta t}{2} \left(\bar{\mathcal{A}}^{[1]} \frac{\mathbf{y}_{0_{ij-1}} - \mathbf{y}_{0_{ij+1}}}{\Delta x_1} + \bar{\mathcal{A}}^{[2]} \frac{\mathbf{y}_{0_{i-1j}} - \mathbf{y}_{0_{i+1j}}}{\Delta x_2} \right) \\ &+ \frac{(\Delta t)^2}{2} \left(\bar{\mathcal{A}}^{[1]^2} \frac{\mathbf{y}_{0_{ij-1}} - 2\mathbf{y}_{0_{ij}} + \mathbf{y}_{0_{ij+1}}}{\Delta x_1^2} + \bar{\mathcal{A}}^{[2]^2} \frac{\mathbf{y}_{0_{i-1j}} - 2\mathbf{y}_{0_{ij}} + \mathbf{y}_{0_{i+1j}}}{\Delta x_2^2} \right. \\ &\left. + (\bar{\mathcal{A}}^{[1]} \bar{\mathcal{A}}^{[2]} + \bar{\mathcal{A}}^{[2]} \bar{\mathcal{A}}^{[1]}) \frac{\mathbf{y}_{0_{j-1,i-1}} - \mathbf{y}_{0_{j-1,i+1}} - \mathbf{y}_{0_{j+1,i-1}} + \mathbf{y}_{0_{j+1,i+1}}}{\Delta x_1 \Delta x_2} \right), \end{aligned}$$

where the indices i and j define the grid points in the x_1 and x_2 directions, respectively. The Lax–Wendroff method belongs to the class of *spatial operator splitting* methods; it is second-order accurate and stable if the

ratios $|\lambda_i \mathcal{A}^{[1]}|\Delta t/\Delta x$ and $|\mu_i \mathcal{A}^{[2]}|\Delta t/\Delta x$, where λ_i and μ_i, the eigenvalues derived from the Von Neumann stability analysis, are reduced by $\sqrt{8}$ [Str68].

In 1968, Strang proposed a second-order variant of the Lie–Trotter–Godunov method also based on a symmetrization principle. By interchanging the order of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ and carrying out two half steps with reversed sequence gives

$$\mathbf{y}_{\Delta t} = \left(e^{\frac{\Delta t}{2}\mathcal{A}^{[1]}} e^{\frac{\Delta t}{2}\mathcal{A}^{[2]}}\right)\left(e^{\frac{\Delta t}{2}\mathcal{A}^{[2]}} e^{\frac{\Delta t}{2}\mathcal{A}^{[1]}}\right)\mathbf{y}_0. \tag{2.22}$$

Following the notation in (2.6) and (2.7), the method is represented by

$$(\check{\boldsymbol{\alpha}}_2 \Delta t)(\boldsymbol{\alpha}_1 \Delta t), \tag{2.23}$$

for $\alpha_k^{[l]} = 1/2$ for $l, k = 1, 2$. A schematic of the SM2 method is given in Figure 2.3.

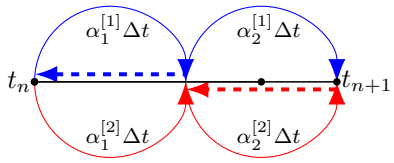

Figure 2.3: Schematic of the SM2 OS method. The solid lines denote the sub-flows and the dashed lines denote the transfer of solution information between sub-flows.

We refer to this method, independently proposed by Strang [Str68] and Marchuk [Mar71], as SM2. After a series expansion, we find the local splitting error to be [HV03]

$$\mathbf{y}_{\Delta t} - \mathbf{y}_{\Delta t}^{[2]} = \frac{\Delta t^3}{24}\left(\left[\mathcal{A}^{[1]}, \left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\right] + \left[\mathcal{A}^{[2]}, \left[\mathcal{A}^{[2]}, \mathcal{A}^{[1]}\right]\right]\right)\mathbf{y}_{\frac{\Delta t}{2}}^{[2]} + \mathcal{O}\left((\Delta t)^5\right),$$

where $\mathbf{y}_{\frac{\Delta t}{2}}^{[2]}$ is the solution computed at $t = \Delta t/2$. Because the error for the SM2 method is proportional to $(\Delta t)^3$, this OS method is second-order accurate.

We note that it is common to refer to the modification of (2.22) that combines the half steps into a single step as the Strang (or Strang–Marchuk) method, i.e.,

$$\mathcal{S}_{\Delta t}^{\text{C-SM2}} := (\Delta t/2)(\check{\Delta} t)(\Delta t/2). \tag{2.24}$$

From now on, we will refer to this scheme as C-SM2. Indeed, (2.22) and (2.24) are equivalent only when the exact flows of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ are available; otherwise they ultimately lead to different methods with correspondingly different accuracy and stability properties (see chapters 3 and 4).

2.2.3 Higher-Order Operator Splitting Methods

Not much theory about OS methods was developed until the 1980s. Some higher-order ($p > 2$) symplectic methods were developed in the previous years [MQ02]. However, all of these methods introduced spurious damping leading to inaccurate results. Therefore, they are not considered in this thesis.

The first important higher-order operator splitting method was given by Ruth in 1983 [Rut83], where the author provided an explicit third-order map to preserve the canonical form of the equation of motion for Hamiltonian systems

$$H(\boldsymbol{x},t)=\frac{\mathbf{p}^T\mathbf{p}}{2}+V(\boldsymbol{x},t).$$

The strategy to develop such methods relies essentially on performing canonical transformations $(\boldsymbol{x},\mathbf{p})\rightarrow(\boldsymbol{x}_1,\mathbf{p}_1)$ on the original Hamiltonian, H, to derive initial conditions. Classically, this is done by selecting new sets of coordinates to derive a new expression for H and by expanding it in Taylor series around $t=0$. Ruth derived consistency conditions by choosing to approximate the new Hamiltonian to a certain order (for more details, see [Rut83]). Following this approach, Ruth derived first- and second-order methods consistent with Lie–Trotter–Godunov and SM2, respectively, and proposed a third-order method. This method, denoted with R3 herein, is defined by the coefficients $\alpha_k^{[l]}$ for $l=1,2,\ k=1,2,3$, displayed in table 2.1.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	$7/24$	$2/3$
2	$3/4$	$-2/3$
3	$-1/24$	1

Table 2.1: Coefficients $\alpha_k^{[l]},\ l=1,2,\ k=1,2,3$, for the R3 method.

A noteworthy observation about the coefficients in table 2.1 is that some of them are negative. Although this may seem to be undesirable, especially for deterministic parabolic differential equations, the necessity of negative coefficients for OS methods with order higher than two has been proven by various authors between 1989 and 1996. In 1989, Sheng [She89] was solving PDEs by using exponential splitting with a similar approach as the Lie–Trotter–Godunov method. By expanding in Taylor series and equating coefficients of

$$\mathcal{S}_{\Delta t}(\mathbf{y})=e^{t(\mathcal{A}^{[1]}+\mathcal{A}^{[2]})}\mathbf{y},$$

where $\mathcal{S}_{\Delta t}(\mathbf{y})$ approximates the operator splitting method as a linear combination of products of exponentials as in (2.5), Sheng showed that, for methods with order p greater than two, at least one of the coefficients $\alpha_k^{[l]}$ must be negative. About a year later, Suzuki independently proved the same result [Suz91] by showing that a third-order decomposition of $e^{t(\mathcal{A}^{[1]}+\mathcal{A}^{[2]})}$ is only possible with the use of negative coefficients. In the literature, the results proven by Sheng and Suzuki are known as *Sheng–Suzuki Theorem*.

Theorem 2.2.2 (Sheng, Suzuki)**.** *If the splitting method* (2.5) *is of order $p\geq 3$ for general $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$, then at least one of the $\alpha_k^{[l]}$ for $l=1,2,\ k=1,2,\ldots,s$, is strictly negative.*

The proof can be found in [She89, Suz91].

The systematic interest and development of splitting methods by numerical analysts was triggered by the work of Neri [Ner88], who applied the CBH formula to derive higher-order methods. In 1990, Yoshida suggested an elegant way to construct higher-order symplectic integrators using the CBH formula. The idea

behind the CBH formula is to approximate the exact N-additive flow $\phi_t = \exp\left(t\sum_{l=1}^{N}\mathcal{A}^{[l]}\right)$ generalizing (2.1) by the product of single sub-flows $\phi_t^{[l]} = \exp\left(t\mathcal{A}^{[l]}\right)$, $l = 1, 2, \ldots, N$. By using the CBH formula (2.4), Yoshida was able to find coefficients $\alpha_k^{[l]}$, $k = 1, 2, \ldots, s$, such that

$$\mathcal{S}_{\Delta t}(\mathbf{y}) := \prod_{k=1}^{s} \exp\left(\alpha_k^{[1]}\Delta t\mathcal{A}^{[1]}\right)\exp\left(\alpha_k^{[2]}\Delta t\mathcal{A}^{[2]}\right) = \exp\left(\Delta t\left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)\right) + \mathcal{O}\left((\Delta t)^{p+1}\right). \tag{2.25}$$

To find higher-order OS methods, Yoshida made use of the following lemma.

Lemma 2.2.3 (Yoshida)**.** *Let $\mathcal{S}_{\Delta t}(\mathbf{y})$ be an operator of the form* (2.25) *that has the time reversibility property*

$$\mathcal{S}_{\Delta t}(\mathbf{y})\mathcal{S}_{-\Delta t}(\mathbf{y}) = \mathcal{S}_{-\Delta t}(\mathbf{y})\mathcal{S}_{\Delta t}(\mathbf{y}) = Id(\mathbf{y}).$$

If we expand $\mathcal{S}_{\Delta t}(\mathbf{y})$ using the CBH formula, then all the coefficients for even powers of Δt are zero.

Therefore, a method that has a symmetric form, i.e., such that the condition given by Lemma 2.2.3 holds, is automatically of even order. Having restricted the set of solutions, Yoshida presented various methods up to order eight [Yos90]. Here, we are interested in a four-stage, fourth-order method, denoted from now on by Y4. Using (2.24), the Y4 method takes the form of a 3-fold composition

$$\mathcal{S}_{\theta\Delta t}^{\text{C-SM2}} \circ \mathcal{S}_{(1-2\theta)\Delta t}^{\text{C-SM2}} \circ \mathcal{S}_{\theta\Delta t}^{\text{C-SM2}},$$

where $\theta = 1/(2 - \sqrt[3]{2})$ is arrived at from repeated application of the CBH formula; see [Yos90] for details. Following the notation used for the previous methods, the Y4 method can be written as

$$(\boldsymbol{\alpha}_4\Delta t)(\boldsymbol{\alpha}_3\Delta t)(\boldsymbol{\alpha}_2\Delta t)(\boldsymbol{\alpha}_1\Delta t), \tag{2.26}$$

where the coefficients $\alpha_k^{[l]}$ for $l = 1, 2,\ k = 1, 2, 3, 4$, for the method (2.26) are displayed in table 2.2. The method proposed by Yoshida is remarkable in the sense that is a four-stage, fourth-order OS method with real coefficients $\alpha_k^{[l]}$.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	$\theta/2$	θ
2	$(1-\theta)/2$	$1-2\theta$
3	$(1-\theta)/2$	θ
4	$\theta/2$	0

Table 2.2: Coefficients $\alpha_k^{[l]}$ $l = 1, 2,\ k = 1, 2, 3, 4$, for the Y4 method.

In 1995, McLachlan showed how to use Free Lie algebra theory to determine the order conditions that an OS method needed to attain a particular order. In particular, McLachlan used (2.4) to expand the composition (2.5) asymptotically as

$$\exp\left(t\mathcal{X}_1 + t^2\mathcal{X}_2 + \ldots\right),$$

where $\mathcal{X}_\mathcal{P} \in L^\mathcal{P}(\mathcal{A}^{[1]}, \mathcal{A}^{[2]})$ are the elements of degree $\mathcal{P}$ of the Free Lie algebra L generated by $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$, i.e., the vector space spanned by all the commutators of degree $\mathcal{P}$ of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ [McL95]. Upon choosing a basis for $L^\mathcal{P}(\mathcal{A}^{[1]}, \mathcal{A}^{[2]})$, the $\mathcal{X}_\mathcal{P}$ are polynomials of degree $\mathcal{P}$ with coefficients $\alpha_k^{[l]}$. Naturally, if $\mathcal{X}_\mathcal{P} = 0$ for $\mathcal{P} = 1, 2, \ldots, p-1$, then a method is of order p. As we have seen, there are numerous ways of deriving order conditions for OS methods (see e.g., [Yos90, Suz91]). The number of order conditions increases as we look for higher-order methods. It follows that the number of free parameters also increases, making the search for the $\alpha_k^{[l]}$ that define a method challenging. To reduce the number of order conditions, a strategy that has been adopted by various authors (e.g., [Yos90]) over the years consists of choosing methods that have symmetry properties. In particular, in the framework of Free Lie algebra theory, McLachlan noticed that if s is the number of stages in the method, then, for a non-symmetric (NS) method, there are $2s+1$ free parameters $\alpha_k^{[l]}$, whereas the number reduces to $s+1$ if the method is symmetric (S). The number of order conditions and parameters for methods up to order $p = 6$ is given in table 2.3.

Order	Type NS	Type S
	Order Conditions	
2	3	2
4	8	4
6	23	10
	Free parameters	
	$2s+1$	$s+1$

Table 2.3: Number of order conditions and free parameters for type NS and S OS methods.

McLachlan, who was working on Hamiltonian systems, derived the order conditions for various OS methods, symmetric and non-symmetric, up to order eight and determined the methods that minimized the *Hamiltonian truncation error* defined in [MA92]. Among all the methods presented in [McL95], we present a six-stage, fourth-order symmetric method (M4), which, according to our notation, can be written as

$$(\boldsymbol{\alpha}_6 \Delta t)(\boldsymbol{\alpha}_5 \Delta t)(\boldsymbol{\alpha}_4 \Delta t)(\boldsymbol{\alpha}_3 \Delta t)(\boldsymbol{\alpha}_2 \Delta t)(\boldsymbol{\alpha}_1 \Delta t).$$

The coefficients $\alpha_k^{[l]}$, $l = 1, 2$, $k = 1, 2, \ldots, 6$, for the M4 method are presented in table 2.4.

In 1996, Goldman and Kaper presented a more precise version of the Sheng–Suzuki Theorem [GK96].

Theorem 2.2.4 (Goldman, Kaper)**.** *If p is a positive integer such that $p \geq 3$, then for every order-p accurate in time approximate solution operator of the form* (2.5)*, where s is any finite positive integer,*

$$\min_{k=0,1,\ldots,s} \alpha_k^{[1]} < 0 \quad \textit{and} \quad \min_{k=0,1,\ldots,s} \alpha_k^{[2]} < 0.$$

Additionally, in [GK96], the authors provide the corresponding result for the more general case of an N-additive OS method. Here, we present a sketch of the proof as presented in [BC05]; a complete version can be found in [GK96, HLW06].

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	0.0935003487263305760	0.439051727817158558
2	−0.0690943698810950380	−0.136536314071511211
3	0.4755940211547644620	0.394969172508705306
4	$\alpha_3^{[1]}$	$\alpha_2^{[2]}$
5	$\alpha_2^{[1]}$	$\alpha_1^{[2]}$
6	$\alpha_1^{[1]}$	0

Table 2.4: Coefficients $\alpha_k^{[l]}$, $l = 1, 2$, $k = 1, 2, \ldots, 6$, for the M4 method.

Proof. Let (2.5) be an OS method. The first important observation that leads to an elementary proof of theorem 2.2.4 is that OS methods in the form of (2.5) and first-order ($p = 1$) composition methods are closely connected. In fact, an OS method can be written as a composition of composition methods $\chi_{\Delta t} = \phi_{\Delta t}^{[2]} \circ \phi_{\Delta t}^{[1]}$ and their adjoint $\chi_{\Delta t}^* = \phi_{\Delta t}^{[1]} \circ \phi_{\Delta t}^{[2]}$, for some coefficients β_k, $k = 0, 1, \ldots, 2s$, in the following way

$$\begin{aligned}
\mathcal{S}_{\Delta t} &= \chi_{\beta_{2s}\Delta t}^* \circ \chi_{\beta_{2s-1}\Delta t} \circ \cdots \circ \chi_{\beta_2\Delta t}^* \circ \chi_{\beta_1\Delta t} \circ \chi_{\beta_0\Delta t}^* \\
&= \left(\phi_{\beta_{2s}\Delta t}^{[1]} \circ \phi_{\beta_{2s}\Delta t}^{[2]}\right) \circ \left(\phi_{\beta_{2s-1}\Delta t}^{[2]} \circ \phi_{\beta_{2s-1}\Delta t}^{[1]}\right) \circ \cdots \circ \left(\phi_{\beta_1\Delta t}^{[2]} \circ \phi_{\beta_1\Delta t}^{[1]}\right) \circ \left(\phi_{\beta_0\Delta t}^{[2]} \circ \phi_{\beta_0\Delta t}^{[1]}\right) \\
&= \phi_{\beta_{2s}\Delta t}^{[1]} \circ \phi_{(\beta_{2s}\Delta t + \beta_{2s-1}\Delta t)}^{[2]} \circ \cdots \circ \phi_{(\beta_1\Delta t + \beta_0\Delta t)}^{[1]} \circ \phi_{\beta_0\Delta t}^{[2]}.
\end{aligned} \tag{2.27}$$

If $\beta_{2s} = \beta_0 = 0$, then (2.5) and (2.27) are equivalent if

$$\alpha_k^{[1]} = \beta_{2k-1} + \beta_{2k-2}, \qquad \alpha_k^{[2]} = \beta_{2k} + \beta_{2k-1}, \quad k = 1, 2, \ldots, s,$$

and we can write

$$\sum_{j=1}^{s} \alpha_j^{[1]} = \sum_{j=1}^{s} \alpha_j^{[2]} = \sum_{j=0}^{2s} \beta_j = 1. \tag{2.28}$$

If (2.28) holds, then we say that (2.5) and (2.27) are also consistent.

The next important observation is that for any order p, the order conditions for the coefficients $\alpha_k^{[1]}$, $\alpha_k^{[2]}, k = 1, 2, \ldots s$, are equivalent to the order conditions for the coefficients $\beta_i, i = 1, 2, \ldots, 2s$ (see [McL95]). Therefore, after repeated applications of the CBH formula, (2.5) can be written as

$$\begin{aligned}
\mathcal{S}_{\Delta t} = \exp\Bigg(&\Delta t f_{1,1}\left(\mathcal{A}^{[1]} + \mathcal{A}^{[2]}\right) + \frac{\Delta t^2}{2} f_{2,1}\left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right] + \\
&\Delta t^3 \left(\frac{f_{3,1}}{12}\left[\mathcal{A}^{[1]}, \left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\right] + f_{3,2}\left[\mathcal{A}^{[1]} + \mathcal{A}^{[2]}, \frac{1}{2}\left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\right]\right)\Bigg) + \mathcal{O}\left((\Delta t)^4\right),
\end{aligned}$$

where $f_{p,k}$ are homogeneous polynomials of degree p in the variable β_i. In particular

$$f_{1,1} = \sum_{i=0}^{2s} \beta_i, \qquad f_{2,1} = \sum_{i=0}^{2s} (-1)^{i+1} \beta_i^2, \qquad f_{3,1} = \sum_{i=0}^{2s} \beta_i^3. \tag{2.29}$$

For a method to be of order $p \geq 3$, we require

$$f_{3,1} = \sum_{k=0}^{2s} \beta_k^3 = 0. \tag{2.30}$$

We suppose that more than two β_k are non-zero because $\beta_1^3 + \beta_2^3 = 0$ together with the consistency condition $\beta_1 + \beta_2 = 1$ have no real solutions. Then (2.30) can be written as

$$\sum_{k=1}^{s}(\beta_{2k-1}^3 + \beta_{2k-2}^3) + \beta_{2s}^3 = \sum_{k=1}^{s}(\beta_{2k-1}^3 + \beta_{2k-2}^3) = 0.$$

But then, $\beta_{2k-1}^3 + \beta_{2k-2}^3$ must be negative for some $k = 0, 1, \ldots, s$. Because we have $\operatorname{sgn}(x^3 + y^3) = \operatorname{sgn}(x + y)$, $\forall x, y \in \mathbb{R}$, we must have

$$\alpha_k^{[1]} = \beta_{2k-1}^3 + \beta_{2k-2}^3 < 0.$$

Similarly, for some $1 \leq k \leq s$, by writing (2.30) as

$$\sum_{k=1}^{s}(\beta_{2k}^3 + \beta_{2k-1}^3) + \beta_0^3 = \sum_{k=1}^{s}(\beta_{2k}^3 + \beta_{2k-1}^3) = 0,$$

we get

$$\alpha_k^{[2]} = \beta_{2k}^3 + \beta_{2k-1}^3 < 0,$$

for some $k = 1, 2, \ldots, s$.. □

In 1998, Descombes and Schatzman proposed a fourth-order OS method [DS98] based on the SM2 method. By applying Richardson extrapolation [Ric11] to (2.23), the authors were able to derive a *weighted-sequential* (WS) splitting method [DS98, Des01]. Given (2.2), a WS splitting solves a step of the Lie–Trotter–Godunov splitting first for $t \in [0, \Delta t]$ to obtain $\mathbf{y}_{\Delta t}^{[2]}$. Next, another Lie–Trotter–Godunov splitting is performed by applying $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ reversed order: The solution at the end of each time step is then obtained by a weighted sum of solutions

$$\mathbf{y}_{\Delta t} = \theta \mathbf{y}_{\Delta t}^{[2]} + (1 - \theta)\mathbf{y}_{\Delta t}^{[1]}. \tag{2.31}$$

Following this idea, the method proposed by Descombes and Schatzman, denoted by DS4, reads

$$\frac{4}{3}(\check{\boldsymbol{\alpha}}_5 \Delta t)(\boldsymbol{\alpha}_4 \Delta t)(\boldsymbol{\alpha}_3 \Delta t) - \frac{1}{3}(\check{\boldsymbol{\alpha}}_2 \Delta t)(\boldsymbol{\alpha}_1 \Delta t),$$

where the coefficients $\boldsymbol{\alpha}_k^{[l]}$, $l = 1, 2$, $k = 1, 2, \ldots, 5$ are presented in table 2.5.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	1/2	1/2
2	1/2	1/2
3	1/4	1/2
4	1/2	0
5	1/4	1/2

Table 2.5: Coefficients $\alpha_k^{[l]}$, $l = 1, 2$, $k = 1, 2, \ldots, 5$, for the DS4 method.

In 1999, Sornborger and Stewart in [SS99] used different methods to derive order conditions and found both integer and irrational solutions that minimized the 2-norm of the leading local error coefficients. From

the order conditions, the authors derived a number of third- and fourth-order methods [SS99]. In [Sor07], a specific nine-stage, third-order method is presented and shown to be stable for a simple diffusion-reaction PDE. This method, denoted from now onwards by SS3, is represented by

$$(\boldsymbol{\alpha}_9\Delta t)(\check{\boldsymbol{\alpha}}_8\Delta t)(\check{\boldsymbol{\alpha}}_7\Delta t)(\check{\boldsymbol{\alpha}}_6\Delta t)(\check{\boldsymbol{\alpha}}_5\Delta t)(\boldsymbol{\alpha}_4\Delta t)(\boldsymbol{\alpha}_3\Delta t)(\check{\boldsymbol{\alpha}}_2\Delta t)(\check{\boldsymbol{\alpha}}_1\Delta t).$$

The coefficients $\alpha_k^{[l]}$, $l = 1, 2,\ \ k = 1, 2, \ldots, 9$ for the SS3 method are displayed in table 2.6.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	$1/6$	$1/6$
2	$1/6$	$1/6$
3	$1/6$	$1/6$
4	$-1/3$	$-1/3$
5	$1/6$	$1/6$
6	$1/6$	$1/6$
7	$1/6$	$1/6$
8	$1/6$	$1/6$
9	$1/6$	$1/6$

Table 2.6: Coefficients $\alpha_k^{[l]},\ \ l = 1, 2,\ \ k = 1, 2, \ldots, 9$, for the SS3 method.

Although our interest is mainly focused on fractional-step operator splitting methods, it is important to mention the development, in recent years, of a great number of operator splitting methods for various classes of ordinary differential equations (ODEs). Interest in these methods was driven by the fact that OS methods, compared to standard integrators, have a number of advantages that can be summarized as follows [BCM08]:

- They are usually easy to implement.
- In general, they are explicit.
- The algorithms are sequential, and the intermediate solution can be stored as a vector.
- Specific OS methods can be chosen based on the structure of the problem.
- Some OS methods preserve structural properties of the exact solution such as symplecticity, phase-space volume-preservation, and time-symmetry. Therefore, symplectic splitting methods applied to ODEs can be viewed as a class of *geometric integrators*.

A few authors provided comprehensive surveys of such methods. In particular, in 2002, McLachlan [MQ02] underlined how OS methods constitute a general and flexible tool to construct geometric integrators for conservative systems and classified the best known methods. Description of the classification of these methods goes beyond the scope of this thesis.

In the same year, Blanes and Moan [BM02] showed how the search of more efficient symplectic methods, especially for methods with $p \geq 4$, was not complete. In particular, the authors focused their search on *partitioned Runge–Kutta* (PRK) methods. Given a separable system of the form (2.1), for a time step Δt,

the composition

$$\mathcal{S} = \prod_{k=1}^{s} e^{\alpha_k^{[2]} \Delta t \mathcal{A}^{[2]}} e^{\alpha_k^{[1]} \Delta t \mathcal{A}^{[1]}} + \mathcal{O}((\Delta t)^{p+1}),$$

is a numerical method of order p usually referred in [BM02] as a PRK method. To reduce the set of possible solutions when determining the coefficients $\alpha_k^{[l]}$, $l = 1, 2$, $k = 1, 2, \ldots, s$, the authors only considered methods of even order where the coefficients have the *First Same As Last* (FSAL) property. Using this approach, the number of order conditions for a fourth-order method is only four [BM02]. Here, we consider a six-stage, fourth-order PRK method proposed in [BM02] that was found using a randomized search and an optimization algorithm (see [BM02] for more details). The selected method, denoted with "BM4" from now on, is the one with the lowest global minimizer of the error. The coefficients for the BM4 method are given in table 2.7.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	0.0792036964311957	0.209515106613362
2	0.3531729060497740	−0.143851773179818
3	−0.0420650803577195	$1/2 - (\alpha_1^{[2]} + \alpha_2^{[2]})$
4	$1 - 2(\alpha_1^{[1]} + \alpha_2^{[1]} + \alpha_3^{[1]})$	0

Table 2.7: Coefficients $\alpha_k^{[l]}$, $l = 1, 2, k = 1, 2, 3, 4$, for the BM4 method.

In 2005, Csomós, Faragó, and Havasi proposed to use a WS splitting method rather than the more conventional sequential splitting methods proposed earlier in the literature [CFH05]. The following theorem summarizes the results about the accuracy of the WS method based on the splitting error analysis [CFH05].

Theorem 2.2.5 (Csomós, Faragó, Havasi)**.** *Assume that the operators $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ do not commute. Then the WS method* (2.31) *is*

- *first order without any condition,*
- *second order if and only if $\theta = 1/2$. In this case, we call the method Symmetric Weighted Splitting (SWS).*

Furthermore, if Id is the identity, the SWS method is

- *third order if $\left[\mathcal{A}^{[1]} - \mathcal{A}^{[2]}, \left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\right](Id) = 0$,*
- *fourth order if $\left[\mathcal{A}^{[1]} - \mathcal{A}^{[2]}, \left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]\right]$ is zero on $\left(\mathcal{A}^{[1]} + \mathcal{A}^{[2]}\right)$ and on Id, and if $\left[\mathcal{A}^{[1]}, \mathcal{A}^{[2]}\right]^2(Id) = 0$.*

By considering mainly symplectic methods, Blanes, Casas, and Murua in 2008 published a comprehensive report [BCM08] on operator splitting methods for ODEs focusing mainly on Runge–Kutta–Nyström methods for the integration of Hamiltonian systems. Description about this class of methods goes beyond the scope of this thesis; the interested reader can find more information in [BC05, BCF+13, BM02, BCM08].

We continue our survey of fractional-step operator splitting methods by presenting a family of higher-order OS methods proposed by Koch, Neuhauser, and Thalhammer in [KNT13] and further analyzed by

Auzinger and co-workers in [AH14, AHHK16]. The derivation of the order conditions was based on the literal formulation that an OS method is of order p if and only if the local error $\mathcal{L}(\Delta t) = \mathcal{O}\left((\Delta t)^{p+1}\right)$, and hence

$$\frac{d}{dt}\mathcal{L}(0) = \frac{d^2}{dt^2}\mathcal{L}(0) = \cdots = \frac{d^p}{dt^p}\mathcal{L}(0) = 0.$$

After a calculation based on the Leibniz formula for higher derivatives [AHHK16], according to our notation, the derivatives of order p of $\mathcal{L}(0)$ are defined as

$$\left.\frac{d^p}{dt^p}\mathcal{L}(0)\right|_{\Delta t=0} = \sum_{|\mathbf{q}|=p}\binom{p}{\mathbf{q}}\prod_{k=s}^{1}\sum_{l=0}^{q_k}\binom{q_k}{l}\mathcal{A}^{[2]^l}\mathcal{A}^{[1]^{q_k-l}} - \left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)^p$$

where $\mathbf{q} = (q_1, \ldots, q_s) \in \mathbb{N}_0^s$, the set of integers from 0 to s. As explained in [AH14], the above definition of $\frac{d^p}{dt^p}\mathcal{L}(0)$ defines a minimal set of order conditions that include a linear combination of higher-order commutators of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$, defining a basis for an appropriate Lie algebra. For example, the order condition for the Godunov scheme can be derived from (2.2.3) in the following way. Set the order $p = 1$, the number of stages $s = 1$, and $\mathbf{q} = (0, 1)$, then we have:

$$\begin{aligned}
\left.\frac{d}{dt}\mathcal{L}(0)\right|_{\Delta t=0} &= \sum_{|\mathbf{q}|=1}^{1}\binom{1}{\mathbf{q}}\prod_{k=1}^{1}\sum_{l=0}^{q_k}\binom{q_k}{l}\mathcal{A}^{[2]^l}\mathcal{A}^{[1]^{q_k-l}} - \left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)\\
&= \sum_{|\mathbf{q}|=1}^{1}\binom{1}{\mathbf{q}}\prod_{k=1}^{1}\left[\binom{\alpha_k^{[1]}}{1}\mathcal{A}^{[2]^0}\mathcal{A}^{[1]^1} + \binom{\alpha_k^{[2]}}{1}\mathcal{A}^{[2]^1}\mathcal{A}^{[1]^0}\right] - \left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)\\
&= \sum_{|\mathbf{q}|=1}^{1}\binom{1}{\mathbf{q}}\left[\alpha_1^{[1]}\mathcal{A}^{[1]} + \alpha_1^{[2]}\mathcal{A}^{[2]}\right] - \left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)\\
&= \alpha_1^{[1]}\mathcal{A}^{[1]} + \alpha_1^{[2]}\mathcal{A}^{[2]} - \left(\mathcal{A}^{[1]}+\mathcal{A}^{[2]}\right)
\end{aligned}$$

The consistency condition $\frac{d}{dt}\mathcal{L}(0) = 0$ for order $p = 1$ is then equivalent to $\alpha_1^{[1]} = 1$ and $\alpha_1^{[2]} = 1$.

For this reason, the authors chose to use as a basis the Lyndon (or Lyndon–Shirshov) words over the alphabet $\mathcal{A}^{[1]}, \mathcal{A}^{[2]}$. To select optimal methods among those that satisfy the order conditions, the authors chose to minimize the local error measure (LEM)

$$\text{LEM} := \left(\sum_{q=1}^{\mathbf{l}_{p+1}} |\lambda_{p+1,q}|^2\right)^{1/2}, \tag{2.32}$$

where the $\lambda_{p+1,q}$ are the coefficients of the Lyndon monomials [AHKK17]. This measure is particularly well designed and advantageous because it uses precisely the same framework used to set up the order conditions (see [HLW06, AH14] for more details).

The first higher-order OS method belonging to this class considered here is one proposed by Auzinger and co-workers in [AHKK17]. This method, denoted in [AHKK17] as "Emb 3/2 AKS" and here simply as "AKS3", is third-order accurate, and it is a three-stage method. It is optimal in the sense that it has minimal LEM among similar methods with the same number of stages. Following our notation, this method can be written as

$$(\boldsymbol{\alpha}_3\Delta t)(\boldsymbol{\alpha}_2\Delta t)(\boldsymbol{\alpha}_1\Delta t),$$

where the coefficients $\alpha_k^{[l]}$ for $l = 1, 2,\ \ k = 1, 2, 3$ are displayed in table 2.8. The AKS3 method belongs to the class of *embedded exponential operator operator splitting methods*; it is, in fact, the main integrator of a 3(2) embedded pair. Such methods were presented for the first time in [KNT13]. It is also a *palindromic* method [AH14] because

$$\alpha_k^{[2]} = \alpha_{s+1-k}^{[1]},\ \ k = 1, 2, \ldots, s.$$

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	0.919661523017399857	$\alpha_3^{[1]}$
2	−0.187991618799159782	$\alpha_2^{[1]}$
3	0.268330095781759925	$\alpha_1^{[1]}$

Table 2.8: Coefficients $\alpha_k^{[l]},\ \ l = 1, 2,\ \ k = 1, 2, 3$, for the AKS3 method.

We conclude our review of OS methods by presenting a result by Auzinger, Hoastätter, and Koch [AHK19]. In their work, the authors present a result about generalized splitting methods based on the fact that, in many applications, the commutator $[B, [B, A]]$ and its exponential are easy to compute see [OMF02, AHK19]. This suggests to consider generalized splitting methods of the form

$$\mathbf{y}_{\Delta t} = \mathcal{S}\mathbf{y}_0 = e^{\alpha_s^{[3]}\Delta t^3[B,[B,A]]}e^{\alpha_s^{[2]}\Delta t\mathcal{A}^{[2]}}e^{\alpha_s^{[1]}\Delta t\mathcal{A}^{[1]}}\ldots e^{\alpha_1^{[3]}\Delta t^3[B,[B,A]]}e^{\alpha_1^{[2]}\Delta t\mathcal{A}^{[2]}}e^{\alpha_1^{[1]}\Delta t\mathcal{A}^{[1]}}\mathbf{y}_0. \tag{2.33}$$

Based on (2.33), the authors present the following fourth-order scheme [AHK19]

$$\mathcal{S}(\Delta t) = e^{\frac{1}{6}\Delta t\mathcal{A}^{[2]}}e^{\frac{1}{2}\Delta t\mathcal{A}^{[1]}}e^{\frac{2}{3}\Delta t\mathcal{A}^{[2]}\frac{1}{72}\Delta t^3[B,[B,A]]}e^{\frac{1}{2}\Delta t\mathcal{A}^{[1]}}e^{\frac{1}{6}\Delta t\mathcal{A}^{[2]}}. \tag{2.34}$$

The generalization provided by (2.33) allows the coefficients $\alpha_k^{[2]}$ and $\alpha_k^{[1]},\ \ k = 1, 2, \ldots, s$ to be positive positive even when the OS method is of order $p \geq 3$. However, the negative coefficients are necessary for order $p \geq 5$.

Theorem 2.2.6 (Auzinger, Hoastätter, Koch)**.** *If $\mathcal{S}$ is a generalized splitting method of the form* (2.33) *of order $p \geq 5$ with real coefficients when applied to an equation* (2.2) *where the operators $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ satisfy $[B, [B, A]] = 0$, then at least one of the coefficients $\alpha_k^{[1]},\ \ k = 1, 2, \ldots, s$, is strictly negative.*

A proof of theorem 2.2.6 can be found in [Chi05]

2.2.4 Operator Splitting Methods with Complex Coefficients

As presented in section 2.2.3, many OS methods with high order were developed in the last decade of the twentieth century. In accordance with the Sheng–Suzuki and the Goldman–Kaper theory, methods with order higher than two must contain negative coefficients. Historically, for many applications, this may be a drawback because backward time integration can lead to instabilities. Many authors have also claimed that higher-order OS methods may also be less efficient because the backward time integration forces movement in

the opposite direction to the main integration [Cha03, BCM10, MQ02]. Furthermore, to compensate for the backward time integration, some of these methods have large forward sub-integrations, which may destabilize the integration and increase the truncation error [Cha03].

To overcome these problems, in 2003, Chambers showed how to develop higher-order integrators with smaller coefficients and (potentially) smaller truncation errors [Cha03] than OS methods with real coefficients. The author achieved this result by deriving OS methods with complex coefficients. Throughout the next subsection, we present how some of these methods were derived and describe their advantages and drawbacks.

The approach adopted by Chambers is very similar to what numerous authors did to derive OS methods with real coefficients (see Section 2.2.3). Consider Hamilton's equations for a system of m bodies

$$\frac{dx_i}{dt} = \frac{\partial H}{\partial p_i}, \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i}, \quad i = 1, 2, \ldots, m,$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, and $\mathbf{p} = (p_1, p_2, \ldots, p_m)$ are, respectively, the coordinates and momenta of the bodies and H is the Hamiltonian of the system. Symplectic integrators are often conveniently used for Hamiltonian systems where H can be split into sub-problems that can be solved separately. For example, if we split $H = H_1 + H_2$, the evolution $\mathbf{q}$ after a time step can be expressed as

$$\mathbf{q}(t + \Delta t) = e^{\Delta t(H_1 + H_2)}\mathbf{q}(t),$$

where, unless H_1 and H_2 commute, the RHS is approximated using CBH formula (2.4). Following the same idea used by Yoshida [Yos90] and Forest and Ruth [FR90], Chambers solved order conditions for OS methods but allowing for complex solutions. This way, Chambers was able to derive third- and fourth-order methods with complex coefficients. Here, we present a three-stage, third-order method. From now on, we denote this method with C3. The coefficients $\alpha_k^{[l]}, \quad l = 1, 2, \;\; j = 1, 2, 3$ for the C3 method are presented in table 2.9. There is also a third-order OS method with complex coefficients that is exactly the conjugate of the C3 method. Details can be found in [Cha03].

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	$\frac{1}{4}\left(1 + \frac{i}{\sqrt{3}}\right)$	$\frac{1}{2}\left(1 + \frac{i}{\sqrt{3}}\right)$
2	$\frac{1}{2}$	$\frac{1}{2}\left(1 - \frac{i}{\sqrt{3}}\right)$
3	$\frac{1}{4}\left(1 - \frac{i}{\sqrt{3}}\right)$	0

Table 2.9: Coefficients $\alpha_k^{[l]}, \;\; l = 1, 2, \;\; k = 1, 2, 3$, for the C3 method.

In 2009, Castella and collaborators [CCDV09] presented a result about the order of an OS $\mathcal{S}_{\Delta t}$.

Theorem 2.2.7 (Castella et al.). *Let* $\mathcal{S}_{\Delta t} = \phi_{\Delta t} + \mathcal{O}\left((\Delta t)^{p+1}\right)$. *If*

$$\sum_{k=1}^{s} \alpha_k^{[l]} = 1, \quad l = 1, 2, \ldots, N,$$

$$\sum_{k=1}^{s} \alpha_k^{[l]^{p+1}} = 0, \quad l = 1, 2, \ldots, N,$$

then $\mathcal{S}_{\Delta t}$ *approximates* $\phi_{\Delta t}$ *to order* $p+1$*, i.e.,* $\phi_{\Delta t} \approx \mathcal{O}\left((\Delta t)^{p+2}\right)$

Moreover, [CCDV09]

Corollary 1. *Whenever* p *is even and the composition is symmetric (i.e.,* $\alpha^{[l]}_{s-k+1} = \alpha^{[l]}_k, \forall k$*), then* $\mathcal{S}_{\Delta t}$ *is of order* $p+2$*.*

Castella and coworkers chose to construct higher-order OS methods as a composition of the SM2 method by restricting the set of solutions to ones with complex $\alpha^{[l]}_k$. Additionally, to reduce the computational cost, they wanted to keep the number of stages low, and, to reduce the size of the time sub-steps, they chose coefficients small in magnitude. With these constraints in mind, the authors proposed a four-stage, fourth-order method that we denote here with CCDV4 [CCDV09] . The coefficients for the CCDV4 method are presented in table 2.10, with $\omega_1 = 1/(2 - 2^{1/3}e^{2i\pi/3})$ and $\omega_0 = 1 - 2\omega_1$.

k	$\alpha^{[1]}_k$	$\alpha^{[2]}_k$
1	$\omega_1/2$	ω_1
2	$(\omega_0+\omega_1)/2$	ω_0
3	$(\omega_0+\omega_1)/2$	ω_1
4	$\omega_1/2$	0

Table 2.10: Coefficients $\alpha^{[l]}_k$, $l = 1, 2$, $k = 1, 2, 3, 4$, for the CCDV4 method.

An interesting characteristic of this method is that it is derived after a three-fold composition of the SM2 method, precisely as the Y4 method described in section 2.2.3 (see [AH14, CCDV09] for more details).

Lastly, we introduce the reader to another family of methods that was discovered by Auzinger and collaborators using the same strategy as the used to derive the AKS3 method. The authors found a three-stage, third-order method. According to our notation, this method, denoted with AKS3C from now on, can be written as

$$(\boldsymbol{\alpha}_3\Delta t)(\boldsymbol{\alpha}_2\Delta t)(\boldsymbol{\alpha}_1\Delta t),$$

for coefficients $\alpha^{[l]}_k$, $k = 1, 2, 3$, displayed in table 2.11. As for the C3 method, the conjugate of this method also exists. See [AH14] for more details.

k	$\alpha^{[1]}_k$	$\alpha^{[2]}_k$
1	0	$\frac{1}{4} + i\frac{\sqrt{3}}{12}$
2	$\frac{1}{2} + i\frac{\sqrt{3}}{6}$	$\frac{1}{2}$
3	$\frac{1}{2} - i\frac{\sqrt{3}}{6}$	$\frac{1}{4} - i\frac{\sqrt{3}}{12}$

Table 2.11: Coefficients $\alpha^{[l]}_k$ $l = 1, 2$, $k = 1, 2, 3$, for the AKS3C method.

The same authors also found another three-stage, third-order method that has the characteristic of being palindromic. According to our notation, this method, denoted with AKS3CP from now on, can be written as

$$(\boldsymbol{\alpha}_3\Delta t)(\boldsymbol{\alpha}_2\Delta t)(\boldsymbol{\alpha}_1\Delta t),$$

where the coefficients $\alpha_k^{[l]}$, $k = 1, 2, 3$, are displayed is table 2.12.

k	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	$0.201639688260407656 + i0.105972321241365172$	$\alpha_3^{[1]}$
2	$0.410612900985895537 - i0.206043441934939727$	$\alpha_2^{[1]}$
3	$0.387747410753696807 + i0.100071120693574555$	$\alpha_1^{[1]}$

Table 2.12: Coefficients $\alpha_k^{[l]}$ $l = 1, 2,$ $k = 1, 2, 3$, for the AKS3CP method.

Another OS method we consider here was proposed by Auzinger, Hofstätter, Ketcheson, and Koch. This method was derived by minimizing (2.32), and it was presented in [AHKK17]. The method we consider, denoted in [AHKK17] as "Emb 4/3 A c" and here simply by AK4, is a five-stage, fourth-order method and belongs to the family of *embedded exponential operator splitting methods*. It is, in fact, the principal integrator of a 4(3) pair of methods. Using our notation, the AK4 method can be written as

$$(\boldsymbol{\alpha}_5\Delta t)(\boldsymbol{\alpha}_4\Delta t)(\boldsymbol{\alpha}_3\Delta t)(\boldsymbol{\alpha}_2\Delta t)(\boldsymbol{\alpha}_1\Delta t),$$

with coefficients $\alpha_j^{[i]}$ reported in Table 2.13.

j	$\alpha_j^{[1]}$	$\alpha_j^{[2]}$
1	0.109525706004194176 - i0.046046876563351871 5	$\alpha_5^{[1]}$
2	0.229070097527301312 + i0.011052076098794735 0	$\alpha_4^{[1]}$
3	0.207808170031590079 + i0.001935040036914476 5	$\alpha_3^{[1]}$
4	0.225474403617092379 + i0.143352673211691591 0	$\alpha_2^{[1]}$
5	0.228121622819822054 - i0.110292912784048931 0	$\alpha_1^{[1]}$

Table 2.13: Coefficients $\alpha_j^{[i]}$, $l = 1, 2,$ $k = 1, 2, \ldots, 5$, for the AK4 method.

As we have seen, in recent years, various authors have come up with splitting methods with complex coefficients with positive real part. This procedure allows to overcome the order barrier where splitting methods of order greater than two involve necessarily negative coefficients in the real space. One drawback of this class of methods is that, in general, splitting methods with complex coefficients are considered about

four times more expensive [BCM10] than the corresponding methods with real coefficients, and this generally makes them uncompetitive in practice. However, one can think that the main application of OS methods with complex coefficients could be on parabolic PDEs. In fact, according to the theory, higher-order methods with real coefficients cannot be used because of the necessary backward time integration. However, it has been shown in [CS18a, CS18b] that higher-order OS methods with real coefficients maintain their order of accuracy for a certain class of problems such as the bidomain and the monodomain model. Results about numerical experiments that confirm the proper accuracy and the gains in efficiency of higher-order OS with real coefficients and how they compare to methods with complex coefficients are presented in chapter 4.

As for the practical implementation of splitting methods with complex coefficients, the general idea is that the numerical integration has to be carried using complex variables. However, as pointed out in [Cha03, BCM10], it is claimed that for problems with real solutions, for the output, one should take either the real part of the variables or their modulus only. In fact, as explained in [BCM10], the authors observed that removing the imaginary part at each step, i.e., projecting on the real space at each step, the error growth can be considerably diminished in some cases.

2.2.5 Operator Splitting Methods used in the Numerical Experiments

In this section, we present a concise summary of the OS methods that are used in the numerical experiments presented in chapters 3 and 4.

OS abbreviation	Reference	Order	Number of Stages	Coefficients
SM2	(2.23)	2	2	Real
SS3	table 2.6	3	9	Real
AKS3	table 2.8	3	3	Real
R3	table 2.1	3	3	Real
Y4	table 2.2	4	4	Real
C3	table 2.9	3	3	Complex
AKS3CP	table 2.12	3	3	Complex
CCDV4	table 2.10	4	4	Complex

Table 2.14: Summary of OS methods used in the numerical experiments

3 On the Stability of Operator Splitting Methods

This chapter treats the stability of OS methods. In section 3.1, we present some results from the literature regarding the stability of OS methods and we motivate our research. In section 3.2, we describe the connection between OS and Generalized Additive Runge–Kutta methods, and how to write an extended Butcher tableau that incorporates such information. Additionally, we present a general result about the stability analysis for OS methods. In section 3.3, we present some simple examples that describe how to apply the theory described in section 3.2. Finally, in section 3.4, we show how to apply the order conditions for OS and Runge–Kutta methods to the extended Butcher tableau.

3.1 Motivation

We begin our journey towards understanding the stability of OS methods by introducing to the concepts of linear stability analysis, A-stability, and L-stability.

Consider the following *test equation*

$$\frac{dy}{dt} = \lambda y, \quad y(0) = y_0, \quad t > 0, \tag{3.1}$$

where λ is a constant complex scalar. The solution to (3.1), $y = e^{\lambda t} y_0$, satisfies

$$|y(t)| \leq |y_0| \iff \mathcal{R}e(z) \leq 0, \tag{3.2}$$

where $\mathcal{R}e(z)$ represents the real part of some complex number $z = \lambda \Delta t$. When we apply a numerical method to the test equation, if we define the numerical solution after n time steps to be $\mathbf{y}_n$, it is possible to define an amplification factor $\mathbf{y}_{n+1} = \mathcal{R}(z)\mathbf{y}_n$ and a region of absolute stability as that region of the complex z-plane where

$$|\mathbf{y}_{n+1}| \leq |\mathbf{y}_n| \;\; n = 0, 1, 2, \ldots. \tag{3.3}$$

Based on these definitions, the concept of *A-stability* can be defined in the following way [AP98].

Definition 1 (A-stability)**.** *A method is considered to be A-stable if its absolute stability region contains the entire left plane, i.e., if a complex value $z \in \mathbb{C}$ has $|\mathcal{R}(z)| \leq 1$ whenever $\mathcal{R}e(z) \leq 0$.*

In other words, if a method is A-stable, it is ensured that modes in the numerical solution decay when the corresponding modes in the original problem decay [RS05].

Another useful concept is regarding the stability of numerical methods is *L-stability* [AP98].

Definition 2 (L-stability)**.** *A method is considered to be L-stable if it is A-stable and the condition* $\lim_{z\to-\infty}\mathcal{R}(z) = 0$ *holds.*

This ensures that, for an L-stable method, the amplification factor $\mathcal{R}(z)$ has the correct asymptotic behaviour in the limit of large negative z.

Having established the main concepts about linear stability analysis of numerical methods, we summarize some results about the stability of OS methods presented by various authors.

We start our survey by describing the work of Ropp and Shadid. In [RS05], the authors explored the stability of first- and second-order OS methods applied to the Brusselator equation [PL68]

$$\frac{\partial T}{\partial t} = D_1\frac{\partial^2 T}{\partial x^2} + a - (b+1)T + T^2C, \tag{3.4a}$$

$$\frac{\partial C}{\partial t} = D_2\frac{\partial^2 C}{\partial x^2} + bT - T^2C, \tag{3.4b}$$

subject to time-independent Dirichlet boundary conditions $T(0,t) = T(1,t) = a$, $C(0,t) = C(1,t) = b/a$, $a = 0.6$, $b = 2$, $D_1 = D_2 = 1/40$, ICs $T(x,0) = a + x(1-x)$ and $C(x,0) = b/a + x^2(1-x)$ and for $t = [0,2]$. Numerical solutions can be found via OS methods by splitting the diffusion and the reaction terms in system (3.4) as

$$\frac{\partial \mathbf{y}}{\partial t} = \mathcal{A}_D^{[1]}(\mathbf{y}) + \mathcal{A}_R^{[2]}(\mathbf{y}), \tag{3.5}$$

where $\mathbf{y} = [T, C]^T$ and $\mathcal{A}_D^{[1]}(\mathbf{y})$ and $\mathcal{A}_R^{[2]}(\mathbf{y})$ are the diffusion and the reaction terms, respectively defined as

$$\mathcal{A}_D^{[1]}(\mathbf{y}) = \begin{bmatrix} D_1\frac{\partial^2 T}{\partial x^2} \\ D_2\frac{\partial^2 C}{\partial x^2} \end{bmatrix} \text{ and } \mathcal{A}_R^{[2]}(\mathbf{y}) = \begin{bmatrix} a - (b+1)T + T^2C \\ bT - T^2C \end{bmatrix}. \tag{3.6}$$

To solve the split system, the authors considered the G1 and the combined-SM2 splitting method described in section 2.2. In both cases, the authors chose to integrate the reaction term in (3.5) as a system of ODEs using Heun's method [RS05]. When using the G1 splitting, the diffusion part is solved using the first-order Backward Euler (BE) method. Similarly, when using the combined-SM2 splitting, the diffusion part is solved using the second-order Implicit Midpoint method. By applying linear stability analysis to the split system, the authors established the following result [RS05]:

Theorem 3.1.1 (Ropp–Shadid, Godunov A-stability)**.** *Consider an operator-split time-discretization of* (3.5)*. Assume that*

- $\mathcal{A}_D^{[1]} + \mathcal{A}_R^{[2]}$ *is negative definite;*
- $\mathcal{A}_D^{[1]}$ *is normal with real negative eigenvalues,* $\lambda_i < 0,\ \ i = 1, 2, \ldots n.$

Let $\upsilon_R(\Delta t) = ||\mathcal{R}(\Delta t A_R)||_{L_2}$*, i.e., the 2-norm of the amplification factor of the reaction term. If the following holds:*

$$\max_i |\mathcal{R}_{\mathcal{D}}(\Delta t\lambda_i)| \le 1/\upsilon_R(\Delta t) \quad \textit{for} \quad 0 \le \Delta t \le \Delta t^* \le \infty, \tag{3.7}$$

where $\mathcal{R}_{\mathcal{D}}$ *is the amplification factor for the diffusion term and* Δt^* *is the time-step restriction required for A-stability, then the Godunov OS scheme is A-stable.*

The proof can be found in [RS05]. The authors also hint at a similar result in the case where the combined-SM2 splitting is used (see [RS05]).

An even stronger result was derived in the case where the diffusion part is solved using an L-stable Singly Diagonally Implicit Runge–Kutta (SDIRK) method. In fact, further results in [RS05] confirmed that when using an L-stable scheme, such as the two-stage, second-order SDIRK (SDIRK2O2), the solution is smooth and well behaved for all values of Δt. Additionally, the authors found that when solving the diffusion term with a method that is not L-stable, a time step restriction Δt^* is required to avoid instabilities [RS05] .

In a later publication [RS09], the same authors studied the stability of an advection-diffusion-reaction (ADR) equation solved with the G1 and the combined-SM2 splittings. In a similar way as in [RS05], the authors study the stability of the splitting method and derive similar results for A- and L- stability of the G1 and the combined-SM2 methods for the ADR case.

The results presented in [RS05] and [RS09] provide some insight regarding the stability of the G1 and the combined-SM2 methods in certain cases. However, they do not provide results regarding the stability of others OS and time-stepping methods.

More results about the stability of OS methods were further described by Christlieb, Liu, and Xu in [CLX15]. The authors considered the problem

$$\frac{d\mathbf{y}}{dt} = \mathcal{A}(\mathbf{y}) = \sum_{l=1}^{N} \mathcal{A}^{[l]}(\mathbf{y}), \qquad \mathbf{y}(0) = \mathbf{y}_0. \tag{3.8}$$

The idea of the work presented in [CLX15] was to write OS methods as additive Runge–Kutta (ARK) methods. The family of ARK methods belongs to the class of Runge–Kutta methods that can be defined as [HLW06].

Definition 3 (RK methods)**.** *Let b_i and a_{ij}, $i,j = 1,2,\ldots,\tilde{s}$, be real numbers and let $c_i = \sum_{i=1}^{\tilde{s}} a_{ij}$. One step of an $\tilde{s}$-stage Runge–Kutta (RK) method is given by*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{i=1}^{\tilde{s}} b_i \mathcal{A}(t_n + c_i \Delta t, \tilde{\mathbf{y}}_i), \tag{3.9a}$$

$$\tilde{\mathbf{y}}_i = \mathbf{y}_n + \Delta t \sum_{j=1}^{\tilde{s}} a_{ij} \mathcal{A}(t_n + c_j \Delta t, \tilde{\mathbf{y}}_j), \;\; i = 1,2,\ldots,\tilde{s}. \tag{3.9b}$$

The coefficients b_i, c_i, and a_{ij}, $i,j = 1,2,\ldots,\tilde{s}$, of a RK method can be written in a table called a Butcher tableau in the following way

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1\tilde{s}} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2\tilde{s}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{\tilde{s}} & a_{\tilde{s}1} & a_{\tilde{s}2} & \ldots & a_{\tilde{s}\tilde{s}} \\ \hline & b_1 & b_2 & \ldots & b_{\tilde{s}} \end{array}.$$

We can cast the Butcher tableau in vector/matrix form in the following way

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

where, for later convenience of notation, $\mathbf{b}$ is intended to be a row vector.

For an IVP in the form of (3.8), when different $\tilde{s}$-stage Runge–Kutta (RK) integrators are applied to each operator $\mathcal{A}^{[l]}$, the entire numerical method is called an additive RK (ARK) method [KC03].

Definition 4 (ARK methods)**.** *Let $b_i^{[l]}$ and $a_{ij}^{[l]}$, $i,j = 1,2,\ldots,\tilde{s}$, $l = 1,2,\ldots,N$, be real numbers, and let $c_i^{[l]} = \sum_{i=1}^{\tilde{s}} a_{ij}^{[l]}$. One step of an $\tilde{s}$-stage ARK method is given by*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{l=1}^{N} \sum_{i=1}^{\tilde{s}} b_i^{[l]} \mathcal{A}^{[l]}(t_n + c_i^{[l]} \Delta t, \tilde{\mathbf{y}}_i), \tag{3.10a}$$

$$\tilde{\mathbf{y}}_i = \mathbf{y}_n + \Delta t \sum_{l=1}^{N} \sum_{j=1}^{\tilde{s}} a_{ij}^{[l]} \mathcal{A}^{[l]}(t_n + c_j^{[l]} \Delta t, \tilde{\mathbf{y}}_j), \quad i = 1,2,\ldots,\tilde{s}, \tag{3.10b}$$

where $b_i^{[l]}$, $c_j^{[l]}$, and $a_{ij}^{[l]}$ are the coefficients of the method.

The Butcher tableau for ARK methods can be written as [SG15]

$$\begin{array}{c|c|c|c|c|c} \mathbf{c}^{[1]} & \ldots & \mathbf{c}^{[N]} & \mathbf{A}^{[1]} & \ldots & \mathbf{A}^{[N]} \\ \hline & & & \mathbf{b}^{[1]} & \ldots & \mathbf{b}^{[N]} \end{array}, \tag{3.11}$$

where $\mathbf{A}^{[l]}$, $l = 1,2,\ldots,N$, is the integrator of the operator l. Details about the study presented in [CLX15] go beyond the scope of this thesis. However, it is important to point out that all the results regarding the splitting error and the stability of the OS methods considered in [CLX15], again, were formulated based on on the assumptions that each sub-flow is solved exactly (i.e., no error contributions from the integrators). However, it is clear that, during the time integration, both the OS and the methods used in the time integration of the sub-flows introduce errors and potentially contribute to instabilities. It follows, that neither of these components can be ignored when studying the stability of an OS method or when trying to design a new method. We present our contribution to these results in the next section.

3.2 A General Result About Stability of Splitting Methods

The first important assumption that leads to establishing a general result for the stability of OS methods is that we consider Runge–Kutta method (3.9) as integrators used in the time integration of the sub-flows.

An important observation is that the family Generalized Additive Runge–Kutta (GARK) methods defined in [SG15] can be regarded as a natural generalization of ARK methods (3.10). In fact, GARK methods can be defined in the following way:

Definition 5 (GARK methods)**.** *Let $b_i^{[l]}$ and $a_{ij}^{[l]}$, $i,j = 1,2,\dots,\tilde{s}$, $l = 1,2,\dots,N$, be real numbers, and let $c_i^{[l]} = \sum_{i=1}^{\tilde{s}} a_{ij}^{[l]}$. One step of a GARK scheme with an N-way partitioning of the right-hand side of* (3.8) *stages reads*

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{l=1}^{N} \sum_{i=1}^{\tilde{s}^{[l]}} b_i^{[l]} \mathcal{A}^{[l]}(t_n + c_i^{[l]} \Delta t, \tilde{\mathbf{y}}_i), \tag{3.12a}$$

$$\tilde{\mathbf{y}}_i = \mathbf{y}_n + \Delta t \sum_{l=1}^{N} \sum_{j=1}^{\tilde{s}^{[l]}} a_{ij}^{[l]} \mathcal{A}^{[l]}(t_n + c_j^{[l]} \Delta t, \tilde{\mathbf{y}}_j). \tag{3.12b}$$

In other words, GARK methods generallize the structure of ARK methods by allowing for different stage values with different components of the right-hand side [SG15]. As described in [SG15], any GARK method can be written as an ARK method (and vice versa) so that the Butcher tableaux for GARK methods have the same structure as Butcher tableaux for ARK methods (3.11) (see theorem 2.5 in [SG15]).

With definitions 3 and 5, we are now ready to present a result that describes the connection between the OS method and the RK integrators chosen to solve a split problem.

Theorem 3.2.1. *Consider* (3.8)*. Let s be the number of stages in the OS method considered, and let $\alpha_k^{[l]}$, $l = 1,2,\dots,N$, $k = 1,2,\dots,s$, be the coefficients of the OS method associated with operator l. Let $\bar{\boldsymbol{\alpha}}$ be the vector that contains the coefficients $\alpha_k^{[l]}$ of the OS in the order they are applied during each time step of the time integration. Denote with $\bar{\alpha}_m$ entry m of said vector. Additionally, let*

$$\begin{array}{c|c} \mathbf{c}_k^{[l]} & \mathbf{A}_k^{[l]} \\ \hline & \mathbf{b}_k^{[l]} \end{array},$$

$l = 1,2,\dots,N$, $k = 1,2,\dots,s$, be the Butcher tableau of each $\tilde{s}_k^{[l]}$-stage Runge–Kutta integrator used in the sub-integration time-stepping. Then, we can construct an extended Butcher tableau with the same structure as (3.11) *that incorporates the coefficients of the Runge–Kutta integrators scaled by the coefficients of the OS method. The size of the Butcher tableau is $\mathcal{S} = \sum_{l=1}^{N} \sum_{k=1}^{s} \tilde{s}_k^{[l]}$.*

Proof. Consider (3.8). We integrate the N split operators with sN different integrators with Butcher tableaux

$$\begin{array}{c|c} \mathbf{c}_k^{[l]} & \mathbf{A}_k^{[l]} \\ \hline & \mathbf{b}_k^{[l]} \end{array}, \qquad l = 1,2,\dots,N, \quad k = 1,2,\dots,s.$$

Then the size of the Butcher tableau is $\mathcal{S} = \sum_{l=1}^{N} \sum_{k=1}^{s} \tilde{s}_k^{[l]}$. From $\bar{\boldsymbol{\alpha}}$, fill the first $\tilde{s}_1^{[1]}$ rows of the block with $\alpha_1^{[1]} \mathbf{A}_1^{[1]}$. Next, copy $\alpha_1^{[1]} \mathbf{b}_1^{[1]}$ in the remaining rows to get

$$\begin{array}{ccc|cccc|ccc|ccc} & & & \alpha_1^{[1]}\mathbf{A}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \dots & \mathbf{0} & \ddots & \mathbf{0} \\ & & & \vdots & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \hline & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \end{array}$$

Next, copy $\alpha_1^{[2]}\mathbf{A}_1^{[2]}$ in the second block on the $\tilde{s}_1^{[1]}+1$ column starting from the $\tilde{s}_1^{[1]}+1$ row. Copy $\alpha_1^{[2]}\mathbf{b}_1^{[2]}$ in the remaining rows to get

$$\left|\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c} & & \alpha_1^{[1]}\mathbf{A}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{A}_1^{[2]} & \ddots & \mathbf{0} & \dots & \mathbf{0} & \ddots & \mathbf{0} \\ & & \vdots & \mathbf{0} & \dots & \mathbf{0} & \vdots & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \vdots & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \dots & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \\ \hline & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} \end{array}\right|$$

Repeating the process for the remaining stages of the OS method, we obtain the complete extended Butcher tableau

$$\left|\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c} & & & \alpha_1^{[1]}\mathbf{A}_1^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{A}_1^{[2]} & \vdots & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ & & & \vdots & \mathbf{0} & \alpha_2^{[1]}\mathbf{A}_2^{[1]} & \ddots & \vdots & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \ddots & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots & \vdots & \alpha_s^{[2]}\mathbf{A}_s^{[2]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \vdots & \alpha_s^{[2]}\mathbf{b}_s^{[2]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \alpha_s^{[N]}\mathrm{A}_s^{[N]} \\ \hline & & & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \mathbf{0} & \alpha_s^{[1]}\mathbf{b}_s^{[1]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \alpha_s^{[N]}\mathbf{b}_s^{[N]} \end{array}\right| .$$

The columns on the left of the tableau are filled with the consistency condition

$$c_i^{[l]} = \sum_{j=1}^{s} a_{ij}^{[l]}, \quad i = 1, 2, \dots, S, \;\; l = 1, 2, \dots, N, \tag{3.13}$$

to obtain

$$\left|\begin{array}{c|c|c|c} \alpha_1^{[1]}c_1^{[1]} & 0 & \dots & 0 \\ \sum_{j=1}^{s}\alpha_1^{[1]}b_j^{[1]} & \alpha_1^{[2]}c_1^{[2]} & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \alpha_s^{[N]}a_s^{[N]} \end{array}\right| \dots \tag{3.14}$$

$$\dots \left|\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c} \alpha_1^{[1]}\mathbf{A}_1^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \dots & v & \mathbf{0} & \alpha_1^{[2]}\mathbf{A}_1^{[2]} & \vdots & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ \vdots & \mathbf{0} & \alpha_2^{[1]}\mathbf{A}_2^{[1]} & \ddots & \vdots & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \ddots & \dots & \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \\ \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \vdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \vdots & \vdots & \alpha_s^{[2]}\mathbf{A}_s^{[2]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \vdots & \alpha_s^{[2]}\mathbf{b}_s^{[2]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \alpha_s^{[N]}\mathrm{A}_s^{[N]} \\ \hline \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_1^{[2]} & \mathbf{0} & \alpha_s^{[1]}\mathbf{b}_s^{[1]} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \alpha_s^{[N]}\mathbf{b}_s^{[N]} \end{array}\right| . \tag{3.15}$$

To conclude, we observe that, when solving a system (3.8) by combining an s-stage OS method with various RK time-stepping methods, the time integration described by (3.15) can be written as GARK method where the entries have been scaled by the coefficients $\alpha_k^{[l]}$, $l = 1, 2, \dots, N$, $k = 1, 2, \dots, s$ of the OS method. □

For convenience, we show the extended Butcher tableau for a consistent 3-stage, 3-additive OS method. Note that this tableau has structure

$$\begin{array}{c|c|c|c|c|c|c} \mathbf{c}^{[1]} & \mathbf{A}^{[1]} & \mathbf{c}^{[2]} & \mathbf{A}^{[2]} & \dots & \mathbf{c}^{[N]} & \mathbf{A}^{[N]} \\ \hline & \mathbf{b}^{[1]} & & \mathbf{b}^{[2]} & \dots & & \mathbf{b}^{[N]} \end{array}. \tag{3.16}$$

$$\begin{array}{c|cccccccc} \alpha_1^{[1]}\mathbf{c}_1^{[1]} & \alpha_1^{[1]}\mathbf{A}_1^{[1]} & \mathbf{0} & \dots & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[1]}\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[1]}\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \dots & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[1]}\mathbb{1} + \alpha_2^{[1]}\mathbf{c}_2^{[1]} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbf{A}_2^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} + \alpha_3^{[1]}\mathbf{c}_3^{[1]} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} + \alpha_3^{[1]}\mathbf{c}_3^{[1]} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0}v & \dots & \alpha_3^{[1]}\mathbf{A}_3^{[1]} & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]} + \alpha_3^{[1]})\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_3^{[1]}\mathbb{1}\mathbf{b}_3^{[1]} & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]} + \alpha_3^{[1]})\mathbb{1} & \alpha_1^{[1]}\mathbb{1}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbb{1}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_3^{[1]}\mathbb{1}\mathbf{b}_3^{[1]} & \mathbf{0} \; \mathbf{0} \\ \hline & \alpha_1^{[1]}\mathbf{b}_1^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[1]}\mathbf{b}_2^{[1]} & \mathbf{0} & \mathbf{0} & \alpha_3^{[1]}\mathbf{b}_3^{[1]} & \mathbf{0} \; \mathbf{0} \end{array} \dots \tag{3.17}$$

$$\dots \begin{array}{c|cccccccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[2]}\mathbf{c}_1^{[2]} & \mathbf{0} & \alpha_1^{[2]}\mathbf{A}_1^{[2]} & \dots & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[2]}\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ \alpha_1^{[2]}\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \dots & \dots & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[2]}\mathbf{A}_2^{[2]} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[2]}\mathbf{b}_2^{[2]} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[2]}\mathbf{b}_2^{[2]} & \dots & \dots & \mathbf{0} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]})\mathbb{1} + \alpha_3^{[2]}\mathbf{c}_3^{[2]} & \mathbf{0} & \alpha_2^{[2]}\mathbb{1}\mathbf{b}_2^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_2^{[2]}\mathbf{b}_2^{[2]} & \dots & \dots & \alpha_3^{[2]}\mathbf{A}_3^{[2]} \; \mathbf{0} \\ (\alpha_1^{[1]} + \alpha_2^{[1]} + \alpha_3^{[2]})\mathbb{1} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_2^{[2]} & \dots & \dots & \alpha_3^{[2]}\mathbf{b}_3^{[2]} \; \mathbf{0} \\ \hline & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[2]} & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbf{b}_2^{[2]} & \dots & \dots & \alpha_3^{[2]}\mathbf{b}_3^{[2]} \; \mathbf{0} \end{array} \dots \tag{3.18}$$

$$
\begin{array}{c|cccccccccc}
 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
 & \alpha_1^{[3]}\mathbf{c}_1^{[3]} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbf{A}_1^{[3]} & \mathbf{0} & \dots & \dots & \dots & \mathbf{0} & \mathbf{0} \\
 & \alpha_1^{[3]}\mathbb{1} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \dots & \dots & \mathbf{0} & \mathbf{0} \\
 & \alpha_1^{[3]}\mathbb{1} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \dots & \dots & \dots & \mathbf{0} \\
\dots & \alpha_1^{[3]}\mathbb{1} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \dots & \dots & \dots & \mathbf{0} \\
 & \alpha_1^{[3]}\mathbb{1} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \alpha_2^{[3]}\mathbf{A}_2^{[3]} & \dots & \dots & 0 \\
 & \alpha_1^{[3]}\mathbb{1} + \alpha_2^{[3]}\mathbf{c}_2^{[3]} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_2^{[3]} & \mathbf{0} & \dots & \alpha_2^{[3]}\mathbf{A}_2^{[3]} & \dots & \dots & \mathbf{0} \\
 & (\alpha_1^{[3]} + \alpha_2^{[3]})\mathbb{1} & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \alpha_2^{[3]}\mathbb{1}\mathbf{b}_2^{[3]} & \dots & \dots & \mathbf{0} \\
 & (\alpha_1^{[3]} + \alpha_2^{[3]})\mathbb{1} + \alpha_3^{[3]}\mathbf{c}_3^{[3]} & \mathbf{0} & \mathbf{0} & \alpha_1^{[2]}\mathbb{1}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \alpha_1^{[3]}\mathbb{1}\mathbf{b}_2^{[3]} & \dots & \dots & \alpha_3^{[3]}\mathbf{A}_3^{[3]} \\
\hline
 & & \mathbf{0} & \mathbf{0} & \alpha_1^{[3]}\mathbf{b}_1^{[3]} & \mathbf{0} & \dots & \alpha_1^{[3]}\mathbf{b}_2^{[3]} & \dots & \dots & \alpha_3^{[3]}\mathbf{b}_3^{[3]}
\end{array}
\tag{3.19}
$$

We define a GARK method scaled by the coefficients of the OS method in the following way:

Definition 6 (OS-GARK methods)**.** *Consider* (3.8) *and assume that we advance the time integration by choosing a combination of OS methods and Runge–Kutta time-stepping methods. Then, one step of an OS-GARK methods reads*

$$
\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{k=1}^{s} \sum_{l=1}^{N} \sum_{i=1}^{\tilde{s}_k^{[l]}} \alpha_k^{[l]} b_i^{[l]} \mathcal{A}^{[l]}(t_n + c_i^{[l]} \Delta t, \tilde{\mathbf{y}}_i), \tag{3.20a}
$$

$$
\tilde{\mathbf{y}}_i = \mathbf{y}_n + \Delta t \sum_{k=1}^{s} \sum_{l=1}^{N} \sum_{j=1}^{\tilde{s}_k^{[l]}} \alpha_k^{[l]} a_{ij}^{[l]} \mathcal{A}^{[l]}(t_n + c_j^{[l]} \Delta t, \tilde{\mathbf{y}}_j). \tag{3.20b}
$$

Having determined the connection between the combination of OS and RK methods used to solve (3.8), the extended Butcher tableau, and OS-GARK methods, we are now ready to present a general result about the stability of OS-GARK methods.

Theorem 3.2.2. *We apply the OS-GARK method* (3.20) *to the linear test equation*

$$
\dot{y} = \sum_{l=1}^{N} \lambda^{[l]} y. \tag{3.21}
$$

We define $z^{[l]} = \Delta t \lambda^{[l]}$, $\mathbf{B}^{[l]} = [\alpha_1^{[l]} \mathrm{b}_1^{[l]}, \alpha_1^{[l]} \mathrm{b}_2^{[l]}, \dots, \alpha_s^{[l]} \mathrm{b}_s^{[l]}]$, *and* $\mathbf{A}^{[l]}$ *be the* l^{th} *building block in the extended Butcher tableau. Additionally, we define the stability function of each RK method used to integrate each operator* $\mathcal{A}^{[l]}$, $l = 1, 2, \dots, N$, *to be* $\mathbf{R}_{RK}^{[l]}(z^{[l]})$. *Then, the stability function* $\mathbf{R}(z^{[1]}, z^{[2]}, \dots, z^{[l]})$ *of the OS method is given by*

$$
\mathbf{R}(z^{[1]}, z^{[2]}, \dots, z^{[N]}) = \prod_{l=1}^{N} \prod_{k=1}^{s} \mathbf{R}_{RK}^{[l]}(\alpha_k^{[l]} z^{[l]}) = 1 + \left(\sum_{l=1}^{N} z^{[l]} \mathbf{B}^{[l]} \right) \cdot \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - \left(\sum_{l=1}^{N} z^{[l]} \mathbf{A}^{[l]} \right) \right)^{-1} \cdot \mathbb{1}_{\mathcal{S}\times 1}, \tag{3.22}
$$

where $\mathbf{I}_{S\times S}$ *denotes the* $\mathcal{S} \times \mathcal{S}$ *identity matrix and* $\mathbb{1}_{\mathcal{S}\times 1}$ *represents the* $\mathcal{S} \times 1$ *vector of ones.*

Proof. Suppose that we consider an s-stage OS method with coefficients $\bar{\boldsymbol{\alpha}}$.

Assume that we apply an RK method (3.9) to the split operator $\mathcal{A}^{[1]}$ for $t = \bar{\alpha}_1 \Delta t$. Then, after performing linear stability analysis, by defining the intermediate solution at the end of the sub-step to be $y_{[1]}$, we can write

$$y_{[1]} = \mathbf{R}^{[1]}_{\mathrm{RK}}(\bar{\alpha}_1 z^{[1]}) y_n, \tag{3.23}$$

with

$$\mathbf{R}^{[1]}_{\mathrm{RK}}(\bar{\alpha}_1 z^{[1]}) = 1 + z^{[l]} \mathbf{B}^{[l]} \cdot \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - z^{[1]} \mathbf{A} \right)^{-1} \cdot \mathbb{1}_{\mathcal{S}\times 1}. \tag{3.24}$$

Next, assume that we apply a (potentially different) RK method to the operator $\mathcal{A}^{[2]}$ for $t = \bar{\alpha}_2 \Delta t$. Then, after performing linear stability analysis, by defining the intermediate solution at the end of the second sub-step to be $\mathbf{y}_{[2]}$, we can write

$$y_{[2]} = \mathbf{R}^{[2]}_{\mathrm{RK}}(\bar{\alpha}_2 z^{[l]}) y_{[1]} = \mathbf{R}^{[1]}_{\mathrm{RK}}(\alpha^{[1]}_1 1 z^{[1]}) \mathbf{R}^{[2]}_{\mathrm{RK}}(\alpha^{[2]}_2 z^{[2]}) y_n, \tag{3.25}$$

with

$$\mathbf{R}^{[2]}_{\mathrm{RK}}(\alpha^{[2]}_2 z^{[2]}) = 1 + z^{[l]} \mathbf{B}^{[2]} \cdot \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - z^{[2]} \mathbf{A} \right)^{-1} \cdot \mathbb{1}_{\mathcal{S}\times 1}. \tag{3.26}$$

By repeating this process over all the stages, we can write $\mathbf{y_{n+1}}$ as

$$y_{n+1} = \mathbf{R}_{\mathrm{RK}}(\alpha^{[1]}_1 z^{[1]}) \mathbf{R}^{[2]}_{\mathrm{RK}}(\alpha^{[2]}_2 z^{[2]}) \cdot \cdots \cdot \mathbf{R}^{[l]}_{\mathrm{RK}}(\alpha^{[l]}_s 1 z^{[l]}) y_n \tag{3.27}$$

$$= \prod_{l=1}^{N} \prod_{k=1}^{s} \mathbf{R}^{[l]}_{\mathrm{RK}}(\alpha^{[l]}_k z^{[l]}) y_n. \tag{3.28}$$

Suppose now that we apply the GARK method (3.20) to the test equation (3.21). Then we obtain

$$y_{n+1} = y_n + \Delta t \sum_{k=1}^{s} \sum_{l=1}^{N} \sum_{i=1}^{\tilde{s}^{[l]}_k} \alpha^{[l]}_k b^{[l]}_i \lambda^{[l]} \tilde{y}_i, \tag{3.29a}$$

$$\tilde{y}_i = y_n + \Delta t \sum_{k=1}^{s} \sum_{l=1}^{N} \sum_{j=1}^{\tilde{s}^{[l]}_k} \alpha^{[l]}_k a^{[l]}_{ij} \lambda^{[l]} \tilde{y}_j. \tag{3.29b}$$

By casting (3.29b) in matrix form we can write

$$\tilde{\mathbf{Y}} = y_n \mathbb{1}_{\mathcal{S}\times 1} - \sum_{l=1}^{N} \alpha^{[l]}_k z^{[l]} \mathbf{A}^{[l]} \tilde{\mathbf{Y}}, \tag{3.30}$$

where $\tilde{\mathbf{Y}} = [\tilde{y_1}, \tilde{y_2}, \ldots, \tilde{y}_{\tilde{s}^{[l]}_k}]^{\mathrm{T}}$ and $\mathbf{A}^{[l]}$ is the matrix with entries $a^{[l]}_{ij}$ in the Butcher tableau of the Runge–Kutta method applied to operator l. Equation (3.30) implies

$$\tilde{\mathbf{Y}} = y_n \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - \sum_{l=1}^{N} \alpha^{[l]}_k z^{[l]} \mathbf{A}^{[l]} \right)^{-1} \mathbb{1}_{\mathcal{S}\times 1} \tag{3.31}$$

Similarly, we can write (3.29a) as

$$y_{n+1} = y_n + \sum_{l=1}^{N} \alpha^{[l]}_k z^{[l]} \mathbf{B}^{[l]} \tilde{\mathbf{Y}}, \tag{3.32}$$

where $\bar{\mathbf{B}}^{[l]}$ is the vector with entries $b_i^{[l]}$. Combining (3.31) and (3.33) we can write

$$y_{n+1} = y_n \mathbb{1}_{\mathcal{S}\times 1} + \sum_{l=1}^{N} \alpha_k^{[l]} z^{[l]} \mathbf{B}^{[l]} \cdot \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - \sum_{l=1}^{N} \alpha_k^{[l]} z^{[l]} \mathbf{A} \right)^{-1} \cdot y_n \mathbb{1}_{\mathcal{S}\times 1}. \tag{3.33}$$

Then, the stability function for OS-GARK methods is defined as

$$\mathbf{R}(z^{[1]}, z^{[2]}, \ldots, z^{[N]}) = 1 + \sum_{l=1}^{N} \alpha_k^{[l]} z^{[l]} \mathbf{B}^{[l]} \cdot \left(\mathbf{I}_{\mathcal{S}\times\mathcal{S}} - \sum_{l=1}^{N} \alpha_k^{[l]} z^{[l]} \mathbf{A} \right)^{-1} \cdot \mathbb{1}_{\mathcal{S}\times 1}. \tag{3.34}$$

□

3.3 Examples

Having presented the main results about the stability of OS methods, we now describe some simple examples to illustrate how to apply and interpret theorems 3.2.1 and 3.2.2.

Example 3.3.1 (Construction of the extended Butcher tableau)**.** *We now present a simple illustrative problem and the result regarding the construction of the extended Butcher tableau. Consider the following problem*

$$\dot{\mathbf{y}} = \mathcal{A}\mathbf{y} = \mathcal{A}^{[1]}\mathbf{y} + \mathcal{A}^{[2]}\mathbf{y} + \mathcal{A}^{[3]}\mathbf{y}, \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{3.35}$$

where the the RHS has been split into three operators. We choose to integrate the system using the three-stage, second-order AK3-2 OS method (see website reference in [AHKK17]) whose coefficients are given in table 3.1.

k \ l	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$	$\alpha_k^{[3]}$
1	1/2	$1-\sqrt{2}/2$	$\sqrt{2}/2$
2	0	$\sqrt{2}/2$	$1-\sqrt{2}/2$
3	1/2	0	0

Table 3.1: Coefficients $\alpha_j^{[i]}$, $i, j = 1, 2, 3$ for the AK3-2 method.

In this case, $\bar{\alpha}$ reads

$$\bar{\alpha} = [1/2, 1-\sqrt{2}/2, \sqrt{2}/2, 0, \sqrt{2}/2, 1-\sqrt{2}/2, 1/2, 0, 0].$$

Suppose that, at each stage, we decide to integrate $\mathcal{A}^{[1]}$ using the BE, Heun, and FE methods. The tableaux for these methods read, respectively

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}, \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array}, \qquad \textit{and} \qquad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}.$$

Additionally, we choose to integrate $\mathcal{A}^{[2]}$ using the FE, Explicit Midpoint Rule, and BE methods. The tableaux in this case read, respectively

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}, \qquad \textit{and} \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}.$$

Finally, we integrate $\mathcal{A}^{[3]}$ using the FE, BE, and FE methods. In this case, the tableaux are given by

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}, \qquad \textit{and} \qquad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}.$$

According to theorem 3.2.1, the tableau produced has size 11 and a structure similar to (3.16)*. Next, we fill the tableau in the following way. We multiply $\mathbf{A}_1^{[1]} = 1$ by $\bar{\alpha}_1 = 1/2$ and we copy the result in the first row and column of the first building block. Next, we copy $\mathbf{b}_1^{[1]}$ multiplied by $\bar{\alpha}_1 = 1/2$ in the remaining rows to obtain*

$$\begin{array}{cc|c|cccccccccc|cccccccccc} 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & & & & & \ldots & & & 0 & 0 & 0 & 0 & 0 & 0 & & & & \ldots & & 0 \\ 0 & 0 & \frac{1}{2} & 0 & & & & & \ldots & & & 0 & 0 & 0 & 0 & 0 & 0 & & & & \ldots & & 0 \\ 0 & 0 & \frac{1}{2} & 0 & & & & & \ddots & & & 0 & 0 & 0 & 0 & 0 & 0 & & & & \ddots & & 0 \\ \vdots & \vdots & \frac{1}{2} & 0 & & & & & \ldots & & & 0 & 0 & 0 & 0 & 0 & 0 & & & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & & & & \vdots & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & & \ldots & & \vdots \\ \hline & & \frac{1}{2} & & & & & & & & & 0 & 0 & 0 & & & & & & & & & 0 \end{array} \cdots$$

$$\begin{array}{c|cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & & & & & & \ldots & & & 0 & 0 \\ 0 & 0 & 0 & & & & & & \ldots & & & 0 & 0 \\ \cdots\ 0 & 0 & 0 & & & & & & \ddots & & & 0 & 0 \\ \vdots & 0 & 0 & & & & & & \ldots & & & 0 & 0 \\ \vdots & \vdots & \vdots & & & & & & \ldots & & & \vdots & \vdots \\ \hline & 0 & & & & & & & & & & 0 & 0 \end{array} .$$

Next, in the second building block, we copy $\mathbf{A}_1^{[2]}$ multiplied by $\bar{\alpha}_2 = 1 - \sqrt{2}/2$ on the $\tilde{s}_1^{[1]} + 1$ row starting from the $\tilde{s}_1^{[1]} + 1 = 2$ column. We fill the remaining rows with $\mathbf{b}_1^{[2]}$ multiplied by $\bar{\alpha}_2 = 1 - \sqrt{2}/2$ in the

remaining rows to get

$$\left[\begin{array}{cc|c|cccccccccc|cccccccccc}
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \dots & & 0 & 0 & 0 & 0 & 0 & & 0 & & & & \dots & & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \ddots & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & \ddots & & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \ddots & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & \ddots & & 0 \\
\vdots & \vdots & \frac{1}{2} & 0 & & & & & \dots & & & 0 & 0 & \vdots & 1-\frac{\sqrt{2}}{2} & 0 & & & & \vdots & & 0 \\
\vdots & \vdots & \vdots & \vdots & & & & & \dots & & & \vdots & \vdots & \vdots & \vdots & \vdots & & & & \dots & & \vdots \\
\hline
 & & \frac{1}{2} & & & & & & & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & & & 0
\end{array}\right] \cdots$$

$$\left[\begin{array}{c|cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & & & & & \dots & & 0 & 0 \\
0 & 0 & 0 & & & & & \ddots & & 0 & 0 \\
0 & 0 & 0 & & & & & \ddots & & 0 & 0 \\
\vdots & 0 & 0 & & & & & \vdots & & 0 & 0 \\
\hline
 & 0 & & & & & & 0 & & 0 & 0
\end{array}\right] \cdots .$$

Next, in the third block, we copy $\mathbf{A}_1^{[3]}$ multiplied by $\bar{\alpha}_3 = \sqrt{2}/2$ on the $\tilde{s}_1^{[1]}+\tilde{s}_1^{[2]}+1$ row and on the $\tilde{s}_1^{[1]}+\tilde{s}_1^{[2]}+1$ column. We fill the remaining rows with $\mathbf{b}_1^{[3]}$ multiplied by $\bar{\alpha}_3 = \sqrt{2}/2$ in the remaining rows.

$$\left[\begin{array}{cc|c|cccccccccc|cccccccccc}
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \dots & & 0 & 0 & 0 & 0 & 0 & & 0 & & & & \dots & & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \ddots & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & \ddots & & 0 \\
0 & 0 & \frac{1}{2} & 0 & & & & & \ddots & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & \ddots & & 0 \\
\vdots & \vdots & \frac{1}{2} & 0 & & & & & \dots & & & 0 & 0 & \vdots & 1-\frac{\sqrt{2}}{2} & 0 & & & & \vdots & & 0 \\
\vdots & \vdots & \vdots & \vdots & & & & & \dots & & & \vdots & \vdots & \vdots & \vdots & \vdots & & & & \dots & & \vdots \\
\hline
 & & \frac{1}{2} & & & & & & & & & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & & & & & & 0
\end{array}\right] \cdots$$

$$
\begin{array}{c|cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \vdots & & & \dots & & & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & & & \dots & & & 0 & 0 \\
\cdots\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & & & \ddots & & & 0 & 0 \\
\vdots & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & & & \dots & & & 0 & 0 \\
\vdots & 0 & 0 & 0 & \vdots & & & \dots & & & \vdots & \vdots \\
\hline
 & 0 & & & \frac{\sqrt{2}}{2} & & & \dots & & & 0 & 0
\end{array}
$$

.

Repeating the step until we exhaust all the columns in each building block, upon simplification, we get the complete extended Butcher tableau for this case

$$
\begin{array}{c|c|c|cccccccccc|cccccccccc}
\frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1-\frac{\sqrt{2}}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1-\frac{\sqrt{2}}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 1-\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
\hline
 & & & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0
\end{array}
\quad \dots
$$

$$
\cdots
\begin{array}{|ccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\
\hline
0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & 0 & 0 & 1-\frac{\sqrt{2}}{2} & 0 & 0 & 0
\end{array}
$$

.

Example 3.3.2 (Derivation of the stability function)**.** *To understand theorem 3.2.2, consider the following problem*

$$\dot{\mathbf{y}} = \mathcal{A}\mathbf{y} = \mathcal{A}^{[1]}\mathbf{y} + \mathcal{A}^{[2]}\mathbf{y} \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{3.36}$$

where the the RHS has been split into two operators. We choose to integrate the system using the G1 OS method whose coefficients are given by

k \ l	$\alpha_k^{[1]}$	$\alpha_k^{[2]}$
1	1	1

Table 3.2: Coefficients $\alpha_j^{[i]}$ for the G1 method.

In this case, $\bar{\alpha}$ reads

$$\bar{\alpha} = [1, 1].$$

Suppose that, at each stage, we decide to integrate $\mathcal{A}^{[1]}$ using the FE method with tableau

$$
\begin{array}{c|c}
0 & 0 \\
\hline
 & 1
\end{array},
$$

and $\mathcal{A}^{[2]}$ *with the BE method with coefficients*

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}.$$

Then, the extended Butcher tableau reads

$$\begin{array}{cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ \hline & & 1 & 0 & 0 & 1 \end{array}.$$

By applying the extended Butcher tableau to the test equation (3.21)*,* (3.22) *becomes*

$$\mathbf{R}(z^{[1]}, z^{[2]}) = 1 + \begin{pmatrix} z^{[1]} \\ z^{[2]} \end{pmatrix}^T \cdot \begin{pmatrix} 1 & 0 \\ -z^{[1]} & 1 - z^{[2]} \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{3.37}$$

To simplify (3.37)*, we compute*

$$\begin{pmatrix} 1 & 0 \\ -z^{[1]} & 1 - z^{[2]} \end{pmatrix}^{-1} = \frac{1}{1 - z^{[2]}} \begin{pmatrix} 1 - z^{[2]} & 0 \\ z^{[1]} & 1 \end{pmatrix}.$$

Then, we can (3.37) *to obtain the stability function associated with the extended Butcher tableau*

$$\mathbf{R}(z^{[1]}, z^{[2]}) = 1 + \begin{pmatrix} z^{[1]} \\ z^{[2]} \end{pmatrix}^T \cdot \begin{pmatrix} 1 & 0 \\ \frac{z^{[1]}}{1-z^{[2]}} & \frac{1}{1-z^{[2]}} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 1 + \left(z^{[1]} + \frac{z^{[1]} z^{[2]}}{1-z^{[2]}}, \frac{z^{[2]}}{1-z^{[2]}} \right) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} =$$

$$= 1 + z^{[1]} + \frac{z^{[1]} z^{[2]}}{1 - z^{[2]}} + \frac{z^{[2]}}{1 - z^{[2]}} = \frac{1 - z^{[2]} + z^{[1]} - z^{[2]} z^{[1]} + z^{[1]} z^{[2]} + z^{[2]}}{1 - z^{[2]}} = \frac{1 + z^{[1]}}{1 - z^{[2]}} \tag{3.38}$$

To conclude, we apply the FE method to the test equation

$$\dot{\mathbf{y}} = \mathcal{A}^{[1]} \mathbf{y} \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{3.39}$$

to get the stability function for FE

$$\mathbf{R}^{[1]}_{FE}(z^{[1]}) = 1 + z^{[1]}. \tag{3.40}$$

In a similar way, we can apply the BE method to the test equation

$$\dot{\mathbf{y}} = \mathcal{A}^{[2]} \mathbf{y} \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{3.41}$$

to get the stability function of BE

$$\mathbf{R}^{[2]}_{BE}(z^{[2]}) = \frac{1}{1 - z^{[2]}}. \tag{3.42}$$

We notice that the last term in (3.38) *is exactly the product of the* (3.40) *and* (3.42)*, with* $z^{[1]}$ *and* $z^{[2]}$ *scaled by the coefficients of the G1 OS method as expected from theorem 3.2.2.*

3.4 Order Conditions

In section 3.2, we have seen how to construct a Butcher tableau that describes the choice of OS method and RK integrators to advance the time integration for a given problem. Of course, although in most situations this is not desirable, the order of the OS p_{OS} is not always the same as the order p_{RK} of the RK integrators chosen. To check the global order of accuracy of the combination of OS and RK methods, we can use the order conditions for an alternating OS method (2.5), (2.6) and RK methods.

The order condition up to $p_{\mathrm{OS}} = 3$ for OS methods with coefficients $\alpha = \left\{\left(\alpha_k^{[1]}, \alpha_k^{[2]}\right)\right\}$ are given by [BC05]

$$p_{\mathrm{OS}} = 1: \qquad \sum_{k=1}^{s} \alpha_k^{[1]} = 1, \qquad \sum_{k=1}^{s} \alpha_k^{[2]} = 1, \tag{3.43a}$$

$$p_{\mathrm{OS}} = 2: \qquad \sum_{i=1}^{s} \alpha_i^{[2]} \left(\sum_{k=1}^{i} \alpha_k^{[1]} \right) = \frac{1}{2}, \tag{3.43b}$$

$$p_{\mathrm{OS}} = 3: \qquad \sum_{i=1}^{s-1} \alpha_i^{[2]} \left(\sum_{k=i+1}^{s} \alpha_k^{[1]} \right)^2 = \frac{1}{3}, \qquad \sum_{i=1}^{s} \alpha_i^{[1]} \left(\sum_{k=i}^{s} \alpha_k^{[2]} \right)^2 = \frac{1}{3}. \tag{3.43c}$$

For an $\tilde{s}$-stage RK method with Butcher tableau

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

the order conditions p_{RK} up to order 2 are given by

$$p_{\mathrm{RK}} = 1: \qquad \sum_{j=1}^{\bar{s}} b_j = 1, \tag{3.44a}$$

$$p_{\mathrm{RK}} = 2: \qquad \sum_{j=1}^{\bar{s}} b_j c_j = \frac{1}{2}. \tag{3.44b}$$

Having defined the order conditions for OS and RK methods, we are now ready to verify the global order of convergence of the extended Butcher tableau based on the order of convergence of the OS and RK methods.

Example 3.4.1 (SM2 OS with BE on both operators)**.** *When solving*

$$\dot{\mathbf{y}} = \mathcal{A}^{[1]}\mathbf{y} + \mathcal{A}^{[2]}\mathbf{y}, \quad \mathbf{y}(0) = \mathbf{y}_0,$$

using the SM2 OS method and BE on both operators, the extended tableau reads

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

Because $p_{OS} = 2$ and $p_{RK} = 1$, we expect the overall order of the extended Butcher tableau to be equal to 1. *We can check that by verifying the order condition for the RK method* (3.44a) *on each building block of the tableau. For order* 1, *we have*

$$\sum_{j=1}^{4} \alpha_j^{[1]} \mathbf{b}_j^{[1]} = \frac{1}{2} + \frac{1}{2} = 1$$
$$\sum_{j=1}^{4} \alpha_j^{[2]} \mathbf{b}_j^{[2]} = \frac{1}{2} + \frac{1}{2} = 1$$

For order 2, *we need to check condition* (3.44b)

$$\sum_{j=1}^{4} \alpha_j^{[1]} \mathbf{b}_j^{[1]} \cdot \alpha_j^{[1]} \mathbf{c}_j^{[1]} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4} \neq \frac{1}{2}$$
$$\sum_{j=1}^{4} \alpha_j^{[2]} \mathbf{b}_j^{[2]} \cdot \alpha_j^{[2]} \mathbf{c}_j^{[2]} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4} \neq \frac{1}{2}$$

Because the condition for order 2 *is not satisfied, the order of convergence of the extended tableau is* 1 *as expected.*

Example 3.4.2 (SM2 OS with Heun on both operators)**.** *When solving*

$$\dot{\mathbf{y}} = \mathcal{A}^{[1]}\mathbf{y} + \mathcal{A}^{[2]}\mathbf{y}$$

using the SM2 OS method and Heun on both operators, the extended tableau reads

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	1/2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	0	1/4	1/4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2	1/2	1/4	1/4	0	0	0	0	0	0	0	0	1/2	0	0	0	0	0
1/2	1/2	1/4	1/4	0	0	0	0	0	0	0	0	1/4	1/4	0	0	0	0
1/2	1	1/4	1/4	0	0	0	0	0	0	0	0	1/4	1/4	1/2	0	0	0
1/2	1	1/4	1/4	0	0	0	0	0	0	0	0	1/4	1/4	1/4	1/4	0	0
1	1	1/4	1/4	0	0	0	0	1/2	0	0	0	1/4	1/4	1/4	1/4	0	0
		1/4	1/4	0	0	0	0	1/4	1/4	0	0	1/4	1/4	1/4	1/4	0	0

Because $p_{OS} = 2$ and $p_{RK} = 2$, in this case, we expect the overall order of the extended Butcher tableau to be equal to 2. *We can check that by verifying the order conditions for the RK method* (3.44a) *and* (3.44b) *on*

each building block of the tableau. For order 1*, we have*

$$\sum_{j=1}^{4} \alpha_j^{[1]} \mathbf{b}_j^{[1]} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1,$$
$$\sum_{j=1}^{4} \alpha_j^{[2]} \mathbf{b}_j^{[2]} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1.$$

For order 2*, we have*

$$\sum_{j=1}^{4} \alpha_j^{[1]} \mathbf{b}_j^{[1]} \cdot \alpha_j^{[1]} \mathbf{c}_j^{[1]} = \frac{1}{8} + \frac{1}{8} + \frac{1}{4} = \frac{1}{2},$$
$$\sum_{j=1}^{4} \alpha_j^{[2]} \mathbf{b}_j^{[2]} \cdot \alpha_j^{[2]} \mathbf{c}_j^{[2]} = \frac{1}{8} + \frac{1}{8} + \frac{1}{4} = \frac{1}{2}.$$

Because the condition for overall order equal 2 *is satisfied, the order of convergence of the extended tableau is* 2 *as expected.*

4 Numerical Experiments

In this chapter, we report some numerical experiments to assess the accuracy, efficiency, and stability of some of the OS methods presented in the previous chapters. In section 4.1, we describe results about the accuracy and the efficiency of higher-order OS methods compared to the second-order SM2 method on various cardiac electrophysiology problems. In section 4.2, we present results about higher-order OS methods on an advection-diffusion-reaction problem, and we present some preliminary results that support the stability theory presented in chapter 3. Finally, in section 4.3, we extend the use of our stability theory to understand instabilities in the solution of the Brusselator equations, and we compute stable solutions based on our analysis.

4.1 Cardiac Electrophysiology: Bidomain and Monodomain Models

The heart is a muscular organ that pumps blood to the rest of the body. This pumping function is a process that is controlled by a complex pattern of electrical activation of approximately billion muscle cells. Because the myocardial electrical activity is fundamental for the function of the heart, extensive research has been done to understand the different mechanisms occurring at a cellular, tissue, and organ level [SLC06]. It follows that the organ-level electrical activity in the heart is the result of billions of small-scale processes occurring in the cells, yet the knowledge about how these processes interact is limited. For this reason, the use of mathematical models and computer simulation to reproduce the electrical activity of the heart has been a promising technique to study the heart and heart diseases.

Mathematically, the electrophysiological behaviour of the heart can be modelled using the bidomain [Tun78] and the monodomain models [SLC06]. Both are multi-scale reaction-diffusion models that were developed to overcome the fact that the distance scales used to measure the electrical potentials are large compared to the length of a single cell [Tun78]. For this reason, the bidomain and monodomain models use a continuum approach in order to model the large number of cells. The monodomain model is a simplification of the bidomain model [SLC06]. It is widely used by the research community, but the bidomain model is generally considered the most accurate and general model of cardiac tissue electrophysiology [GG16].

Because of the high resolutions in time and space required to produce relevant data, the bidomain and monodomain models are challenging to solve in realistic situations. Accordingly, a common approach to simulate these models is via OS. Historically, first- and second-order OS methods have been used successfully in finding solutions for the bidomain and monodomain models; see, e.g., [SLT05], [STZ16], [QG99] and

references therein. In this section, we show how to obtain stable solutions to the bidomain and monodomain models using higher-order OS methods. Additionally, we present the significant accompanying efficiency gains offered by higher-order methods when compared to lower-order methods.

Before presenting the results of the numerical experiments, we give the mathematical description of the bidomain and monodomain models. In the bidomain model, it is assumed the heart is divided into two separate domains: the intracellular domain and the extracellular domain. The two domains are co-located, but it is assumed that they are separated by the cell membrane. The cell membrane is embedded with small channels called gap junctions that allow ions to pass directly from one cell to another without entering the space between the two cells [SLT05].

The bidomain model has become widely accepted as an accurate and general mathematical description for signal propagation and external stimulation of heart tissue. It couples a system of reaction PDEs on the micro-scale describing the chemical reactions and flow of ions across the cell membrane of individual myocardial cells with a system of diffusion PDEs on the macroscale describing the propagation of the electrical activation through the heart [STZ16].

On a spatial domain $\Omega \subset \mathbb{R}^d$ of dimension d and time interval $[t_0, t_f]$, the bidomain model may be written as

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(t, \mathbf{s}, v), \tag{4.1a}$$

$$\chi C_m \frac{\partial v}{\partial t} + \chi I_{\text{ion}}(t, \mathbf{s}, v) = \nabla \cdot (\sigma_i \nabla v) + \nabla \cdot (\sigma_i \nabla u_e), \tag{4.1b}$$

$$0 = \nabla \cdot (\sigma_i \nabla v) + \nabla \cdot ((\sigma_i + \sigma_e) \nabla u_e), \tag{4.1c}$$

subject to boundary conditions on $\partial\Omega \times [t_0, t_f]$ given by

$$\hat{\mathbf{n}} \cdot (\sigma_i \nabla v + \sigma_i \nabla u_e) = 0, \tag{4.1d}$$

$$\hat{\mathbf{n}} \cdot (\sigma_e \nabla u_e) = 0, \tag{4.1e}$$

where $\mathbf{s} = \mathbf{s}(t, \mathbf{x})$ is a vector describing the cellular state at location $\mathbf{x} \in \Omega$ and time $t \in [t_0, t_f]$, $v = v(t, \mathbf{x})$ is the transmembrane potential, and $u_e = u_e(t, \mathbf{x})$ is the extracellular potential. The ionic current term $I_{\text{ion}}(t, \mathbf{s}, v)$ and the term $\mathbf{f}(t, \mathbf{s}, v)$ are non-linear terms related to the cell model, and σ_i and σ_e are the intracellular and extracellular conductivities of the heart tissue. Finally, χ is the area of the cell membrane per unit volume, C_m is the capacitance of the cell membrane per unit area, and $\hat{\mathbf{n}}$ is the outward unit normal to $\partial\Omega$.

The monodomain model is a simplified version of the bidomain model that describes only the dynamics of the transmembrane potential v [SLC06]. Under the assumption $\sigma_e = \lambda \sigma_i$, where λ is a constant scalar, we can rewrite (4.1c) as

$$\nabla \cdot (\sigma_i \nabla u_e) = -\frac{\lambda}{1 + \lambda} \nabla \cdot (\sigma_i \nabla v).$$

Substituting this into (4.1b) together with (4.1a) give the final form of the monodomain model [SLC06]

$$\frac{\partial \mathbf{s}}{\partial t} = \mathbf{f}(t, \mathbf{s}, v), \tag{4.2a}$$

$$\chi C_m \frac{\partial v}{\partial t} + \chi I_{\text{ion}}(t, \mathbf{s}, v) = \frac{\lambda}{1+\lambda} \nabla \cdot (\sigma_i \nabla v), \tag{4.2b}$$

subject to

$$\hat{\mathbf{n}} \cdot (\sigma_i \nabla v) = 0, \tag{4.2c}$$

where (4.2c) is derived from (4.1d) and (4.1e) in a similar fashion to (4.2b).

To apply OS to the bidomain model, we write the bidomain model (4.1) as

$$\mathbf{M}\dot{\mathbf{Y}} = \mathcal{A}(\mathbf{Y}) = \mathcal{A}^{[1]}(\mathbf{Y}) + \mathcal{A}^{[2]}(\mathbf{Y}), \tag{4.3}$$

with $\mathbf{Y} \doteq (\mathbf{s}^T, v, u_e)^T$,

$$\mathbf{M} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{4.4}$$

$$\mathcal{A}^{[1]}(\mathbf{Y}) = \begin{pmatrix} \mathbf{f}(t, \mathbf{Y}) \\ -\frac{1}{C_m} I_{\text{ion}}(\mathbf{Y}) \\ 0 \end{pmatrix} \text{ and } \mathcal{A}^{[2]}(\mathbf{Y}) = \begin{pmatrix} \mathbf{0} \\ \frac{1}{\chi C_m} \left(\nabla \cdot (\sigma_i \nabla v) + \nabla \cdot (\sigma_i \nabla u_e) \right) \\ \nabla \cdot (\sigma_i \nabla v) + \nabla \cdot ((\sigma_i + \sigma_e) \nabla u_e) \end{pmatrix}. \tag{4.5}$$

Doing so, we split (4.1b) into a non-linear PDE, involving only the I_{ion} term and no spatial derivatives, and a linear PDE [STZ16]. Upon spatial discretization, the non-linear PDE forms an ODE sub-system at each mesh point, and hence we loosely refer to this sub-system as such. Similarly, the discretized linear PDE system forms a differential-algebraic equation (DAE) sub-system with index one where u_e is an algebraic variable and s and v are differential variables. In a similar fashion to the bidomain model, we write the monodomain model (4.2) as

$$\frac{\partial \mathbf{Y}}{\partial t} = \mathcal{A}^{[1]}(\mathbf{Y}) + \mathcal{A}^{[2]}(\mathbf{Y}),$$

where $\mathbf{Y} \doteq (\mathbf{s}^T, v)^T$,

$$\mathcal{A}^{[1]}(\mathbf{Y}) = \begin{pmatrix} \mathbf{f}(t, \mathbf{Y}) \\ -\frac{1}{C_m} I_{\text{ion}}(\mathbf{Y}) \end{pmatrix}, \text{and } \mathcal{A}^{[2]}(\mathbf{Y}) = \begin{pmatrix} \mathbf{0} \\ \frac{\lambda}{(1+\lambda)} \frac{1}{\chi C_m} \nabla \cdot (\sigma_i \nabla v) \end{pmatrix}. \tag{4.6}$$

After spatial discretization, this splitting reduces the non-linear PDE to a set of non-linear ODEs at each spatial mesh point and a linear PDE.

In this section, we present the numerical experiments that were performed in order to assess the accuracy and the efficiency of some higher-order OS methods described in chapter 2, when applied to the bidomain and monodomain models. When applying third-order OS methods to solve the bidomain and the monodomain models, to ensure the overall integration is third-order accurate in time, we need to approximate the sub-flows

of the sub-systems to at least third order in time. For both the bidomain and the monodomain models, we choose to approximate the flow of $\mathcal{A}^{[1]}$ with the ERK3 method defined by the Butcher tableau

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

To approximate the flow of $\mathcal{A}^{[2]}$, we use the L-stable, two-stage, third-order singly diagonal Runge–Kutta method (SDIRK(3,2)) defined by the Butcher tableau

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}$$

with $\gamma = (3+\sqrt{3})/6$ [ARS97].

Similar considerations must be made when solving the bidomain and monodomain models using a fourth-order splitting. To ensure the overall integration is fourth-order accurate, we approximate the flows of the ODE and PDE sub-systems using fourth-order integrators. In this case, for both the bidomain and monodomain models, $\mathcal{A}^{[1]}$ is solved using the classic ERK4 method

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

The numerical method used to approximate the flow of $\mathcal{A}^{[2]}$ for both system is the A-stable, three-stage, fourth-order (SDIRK(3,4)) method defined by the Butcher tableau

$$\begin{array}{c|ccc} (1+\gamma)/2 & (1+\gamma)/2 & 0 & 0 \\ 1/2 & -\gamma/2 & (1+\gamma)/2 & 0 \\ (1-\gamma)/2 & 1+\gamma & -(1-2\gamma) & (1+\gamma)/2 \\ \hline & 1/(6\gamma^2) & 1-1/(3\gamma^2) & 1/(6\gamma^2) \end{array}$$

with $\gamma = \frac{2}{\sqrt{3}}\cos\left(\frac{\pi}{18}\right)$ [Nor74, Cro78].

A detailed summary of the discretization of $\mathcal{A}^{[1]}$ and $\mathcal{A}^{[2]}$ with explicit and implicit RK methods can be found in [TZ15]. All the experiments consist of problems set in one and three spatial dimensions. The

experiments also include a variety of different initial conditions and cardiac ionic cell models with a range of stiffness characteristics.

The Chaste software environment [MAB+13] is used for all the numerical tests. Chaste uses linear finite elements for the spatial discretization and integrates in time as per the method of lines. The (Chaste default) method for handling the I_{ion} term in the finite element method uses ionic current interpolation [PMSW11]. To solve the linear systems (4.5) and (4.6) in Chaste, we use a Krylov subspace solver and the conjugate gradient solver with block Jacobi preconditioner.

Timings were recorded in Chaste 3.4 running in serial on a dual Quad-Core Intel Xeon 2.26GHz with 16GB of RAM running OS X El Capitan 10.11.6 and represent the minimum of ten runs.

4.1.1 Accuracy

We assess the accuracy of the third-order SS3 (table 2.6), ASK3 (table 2.8), and R3 (table 2.1), and fourth-order Y4 (table 2.2) OS methods described in chapter 2 using the Mixed Root Mean Square (MRMS) error [MZS12] defined by

$$[\text{MRMS}]_w = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\tilde{w}_i - w_i}{1+|\bar{w_i}|}\right)^2}, \tag{4.7}$$

where $\tilde{w_i}$ and w_i denote the reference and numerical solution respectively for the quantity w at the space-time point i. The reference solution is assumed to be accurate enough that it can serve as a proxy for the true solution. The order of convergence p of the numerical solution was then computed in the standard way as

$$p = \frac{\log([\text{MRMS}]_{v1}/[\text{MRMS}]_{v2})}{\log(\Delta t_1/\Delta t_2)},$$

where the subscripts 1 and 2 refer to computations performed using time steps Δt_1 and Δt_2. In the remainder of the chapter, the MRMS is measured only for the transmembrane potential v; i.e., only $[\text{MRMS}]_v$, is used. This is because usually v is considered the variable of interest and because the error of u_e is, in general, smaller [SLT05, SLC06].

Reference solutions for the tests were generated as follows. The spatial domain was uniformly discretized into cells of size $\Delta x = 0.001$ mm. We compared solutions computed by the default semi-implicit method [SPVW09] in Chaste by halving the time step until there were 4 to 6 matching digits between successive approximations. The final time step that led to a solution with 4 to 6 matching digits was $\Delta t = 5\text{e}{-6}$ ms for the 1D problems and $\Delta t = 2.5\text{e}{-6}$ ms for the 3D problems. Solutions were then stored from the finest mesh at $N_s = 101$ equally spaced points in the spatial interval and $N_t = 21$ equally spaced points in the time interval of the simulations for a total of $N = N_s N_t$ space-time points.

4.1.1.1 1D experiments

The monodomain model is simulated in one spatial dimension with two cell models, those of FitzHugh and Nagumo (FHN) [Fit66] and Ten Tusscher and Panfilov (TTP) for epicardial tissue [tTP06].

The FHN model has two cellular state variables and is considered to be non-stiff [SD10], whereas the TTP model has 19 cellular variables and is considered to be highly stiff [SD10]. We use the following initial conditions for the variables in each model:

$$\mathbf{s}(0,x) = \mathbf{s}_0,$$
$$v(0,x) = v_0 + 100(1-\sin(x)),$$

where $\mathbf{s}_0$ and v_0 are the default values for $\mathbf{s}$ and v according to the definition in CellML [ALN+03] for the cell model in use. We use the Chaste default parameters $\chi = 1400/\text{cm}$, $C_m = 1\ \mu\text{F/cm}^2$, and $\sigma_i = 1.75$ mS/cm. For all the reported cases, we set the spatial domain to be $(0,1)$ cm and the time interval to $[0,40]$ ms. We use a regular grid with $N_s = 51$ points giving a spatial resolution $\Delta x = 0.02$ mm. Convergence of the third- and fourth-order methods is assessed using different combinations of time steps for both cell models considered. Results for all the higher-order methods described are reported in Tables 4.1, 4.2, 4.3, and 4.4, confirming the expected order of convergence.

	FHN		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03896	–	0.04058	–
1.0e−5	0.01245	2.84	0.01358	2.90
8.0e−6	0.00653	2.89	0.00635	3.04

Table 4.1: Convergence of the SS3 method applied to the monodomain model in one dimension using the FHN and the TTP epicardial cell models.

	FHN		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03683	–	0.03946	–
1.0e−5	0.01304	3.20	0.01058	3.24
8.0e−6	0.00787	2.92	0.00528	3.11

Table 4.2: Convergence of the AKS3 method applied to the monodomain model in one dimension using the FHN and the TTP epicardial cell models.

	FHN		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03748	–	0.03699	–
1.0e−5	0.01102	3.01	0.0101	3.20
8.0e−6	0.00571	2.95	0.00512	3.04

Table 4.3: Convergence for the R3 method applied to the monodomain model in one dimension using the LR1 and the TTP epicardial cell models.

	FHN		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03051	–	0.02985	–
1.0e−5	0.00621	3.92	0.00571	4.07
8.0e−6	0.00321	3.96	0.00237	3.94

Table 4.4: Convergence for the Y4 method applied to the monodomain model in one dimension using the LR1 and the TTP epicardial cell models.

4.1.1.2 3D experiments

The convergence of the higher-order OS methods when applied to solve the bidomain model is verified by considering the cell models of Luo–Rudy Phase I (LR1) [LR91] and TTP for epicardial tissue [tTP06]. The

model of LR1 has nine cellular state variables and is considered mildly stiff [SD10]. We use the following initial conditions for the variables in our model:

$$\begin{aligned} \mathbf{s}(0,x,y,z) &= \mathbf{s}_0, \\ v(0,x,y,z) &= v_0 + 100(1-\sin(xyz)), \\ u_e(0,x,y,z) &= 0, \end{aligned}$$

where, as before $\mathbf{s}_0$ and v_0 are the CellML default values for $\mathbf{s}$ and v according to the cell model in use [ALN+03]. We use the (Chaste default) parameter values $\chi = 1400/\text{cm}$, $C_m = 1\ \mu\text{F/cm}^2$, $\sigma_i = \text{diag}(\sigma_i^f, \sigma_i^n, \sigma_i^t)$, $\sigma_e = \text{diag}(\sigma_e^f, \sigma_e^n, \sigma_e^t)$, with $\sigma_i^f = \sigma_i^n = \sigma_i^t = 1.75$ mS/cm and $\sigma_e^f = \sigma_e^n = \sigma_e^t = 7$ mS/cm.[1] The spatial domain is set to $(0,1)$ cm $\times (0,1)$ cm $\times (0,1)$ cm, and the time interval to $[0,40]$ ms. We use a grid with $N_s = 51$ equally spaced points per direction resulting in a $\Delta x = 0.02$ cm. The accuracy of the third- and fourth-order methods when applied to the bidomain model is assessed for all the cell models considered in this study, and the results are reported in tables 4.5, 4.6, 4.7, and 4.8, confirming the expected order of convergence for all the schemes.

	LR1		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03975	–	0.03948	–
1.0e−5	0.01199	2.90	0.01219	2.89
8.0e−6	0.00597	3.12	0.00611	3.09

Table 4.5: Convergence for the SS3 method applied to the bidomain model in three dimensions using the LR1 and the TTP epicardial cell models.

	LR1		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03751	–	0.03846	–
1.0e−5	0.00995	3.27	0.01145	2.98
8.0e−6	0.00491	2.98	0.00583	3.02

Table 4.6: Convergence for the AKS3 method applied to the bidomain model in three dimensions using the LR1 and the TTP epicardial cell models.

	LR1		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03799	–	0.03699	–
1.0e−5	0.01012	3.21	0.01184	2.90
8.0e−6	0.00495	3.19	0.00578	3.20

Table 4.7: Convergence for the R3 method applied to the bidomain model in three dimensions using the LR1 and the TTP epicardial cell models.

	LR1		TTP	
Δt (ms)	$[\text{MRMS}]_v$	p	$[\text{MRMS}]_v$	p
1.5e−5	0.03147	–	0.03004	–
1.0e−5	0.00571	4.20	0.00583	4.04
8.0e−6	0.00234	3.99	0.00242	3.97

Table 4.8: Convergence for the Y4 method applied to the bidomain model using the LR1 and the TTP epicardial cell models.

[1] It can be observed that because the conductivity tensors are proportional in this example, the bidomain model considered can in theory be written as a monodomain model. It is nevertheless solved numerically as a bidomain model. Accordingly, this observation has no bearing on the convergence rates reported.

4.1.2 Efficiency comparison on the Niederer benchmark problem

A well-known benchmark problem in the field of cardiovascular modelling appears in the study by Niederer et al. [NKB+11], in which various cardiac tissue electrophysiology simulation codes were used to solve a standardized benchmark problem by evaluating convergence properties in space and time. In this section, we briefly present the benchmark problem and the performance results that were obtained for the SS3, AKS3, and R3 third-order and Y4, CCDV4 (table 2.10), and AK4 (table 2.13) fourth-order OS methods.

The Niederer benchmark problem is defined over a cuboidal domain of dimension 20 mm × 7 mm × 3 mm as shown in figure 4.1, with all boundaries assumed to have no-flux boundary conditions and a stimulus delivered from one corner of the domain [NKB+11]. The ventricular fibre microstructure and orientation are modelled using anisotropic tensors. The cardiac electrical activation was modelled using the monodomain model (4.2) coupled with the TTP cell model for epicardial tissue [tTP06]. The final time for the simulation was set to 40 ms. A complete list of the initial values for the state variables can be found in [NKB+11]. The Niederer benchmark problem was solved by various research groups using different codes and the results compared using an "N-version" code evaluation [HR94, NKB+11]. In [NKB+11], the analysis was presented in a quantitative way by looking at the differences in activation times T_1^8 between the corners labelled P1 and P8 in the domain; see figure 4.1. The analysis was performed in this way because activation times at various points are commonly evaluated in experiments, and hence they provide a practical way to extract information about the spatial and temporal errors in codes.

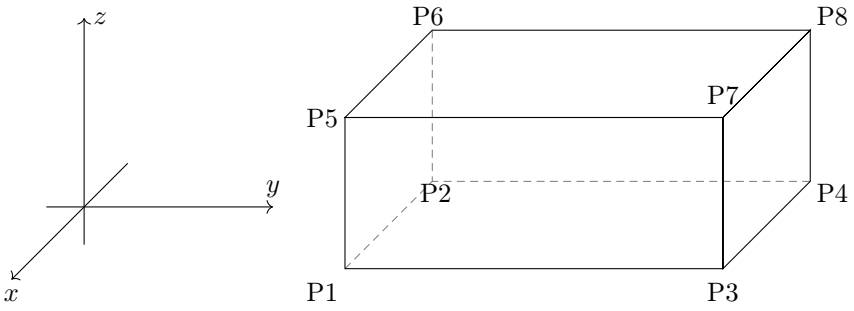

Figure 4.1: Summary of points over the cuboidal domain at which activation times were evaluated.

The reference solution for the Niederer benchmark problem was generated in Nektar++ [CMAC+15], a spectral/hp element code, using the following protocol. The spatial domain was uniformly discretized into cells of size $\Delta x = 0.01$ mm for a total of $N_x = 420,000$ spatial points. The solution of the PDE system (4.2b) was then expressed in terms of polynomials of degree 11 via the spectral/hp element method on each cell. The Strang–Marchuk operator-splitting method together with the forward Euler method for (4.2a) and the second-order IMEX Gear method [EB08] for (4.2b) were used for the temporal discretization to obtain first-order accurate solutions. Seven solutions were then generated using time step sizes according to the sequence $\Delta t_j = 2^j \cdot 10^{-2}$ ms, $j = 1, 0, \ldots, -5$, at which point a precision of two matching decimal places was reached between successive approximations at all shared space-time points for the two finest discretizations.

Richardson extrapolation [Ric11] was then used to generate an eighth-order solution that had converged to six matching decimal places. This solution is used as the reference solution in our experiments.

As per [NKB+11], we first look at the difference in activation times T_1^8 from P1–P8 between the reference solution and the numerical solutions. $T_1^8 \approx 38$ ms for the reference solution. Numerical solutions are generated using the the Chaste default first-order semi-implicit (SI) method [SPVW09] and the SS3 method with ERK3 and SDIRK(3,2) as integrators. The differences ΔT_1^8 in computed T_1^8 between the reference solution and the numerical solution computed using the SI and the SS3 schemes are summarized in tables 4.9 and 4.10, respectively. The spatial and temporal resolutions considered were chosen to match those considered in [NKB+11] to evaluate the various software packages used in the study. The difference in activation times between the reference and the numerical solution is smaller when using the third-order OS method, meaning the higher-order method appears to be more accurate than the lower-order method by this measure [CS18a].

	$\Delta t = 0.05$ ms		$\Delta t = 0.01$ ms		$\Delta t = 0.005$ ms	
	ΔT_1^8	$[\text{MRMS}]_v$	ΔT_1^8	$[\text{MRMS}]_v$	ΔT_1^8	$[\text{MRMS}]_v$
$\Delta x = 0.5$ mm	0.1912	0.1512	0.1657	0.1343	0.1575	0.1175
$\Delta x = 0.2$ mm	0.1535	0.1195	0.1178	0.1078	0.1291	0.0969
$\Delta x = 0.1$ mm	0.1294	0.1057	0.0843	0.0833	0.1057	0.0772

Table 4.9: Difference in activation times from P1–P8 and $[\text{MRMS}]_v$ error between the reference solution and the solution computed using the SI method for various spatial and temporal resolutions.

	$\Delta t = 0.05$ ms		$\Delta t = 0.01$ ms		$\Delta t = 0.005$ ms	
	ΔT_1^8	$[\text{MRMS}]_v$	ΔT_1^8	$[\text{MRMS}]_v$	ΔT_1^8	$[\text{MRMS}]_v$
$\Delta x = 0.5$ mm	0.1763	0.1216	0.1486	0.1047	0.1256	0.0862
$\Delta x = 0.2$ mm	0.1445	0.1045	0.1085	0.0818	0.0916	0.0719
$\Delta x = 0.1$ mm	0.1122	0.0864	0.0867	0.0638	0.0781	0.0579

Table 4.10: Difference in activation times from P1–P8 and $[\text{MRMS}]_v$ error between the reference solution and the solution computed using the SS3 method for various spatial and temporal resolutions.

To provide context to the way errors have been measured in this thesis, tables 4.9 and 4.10 also display the $[\text{MRMS}]_v$ errors between the reference and the numerical solutions computed for the SI and the SS3 methods, respectively. We observe that all the solutions computed with the temporal and spatial resolutions used in [NKB+11] have an $[\text{MRMS}]_v$ in excess of 5%.

Finally, we look at the efficiency of the SS3, AKS3, and R3 third-order and the Y4, CCDV4, and AK4 fourth-order OS methods described in chapter 2 when applied to the Niederer benchmark problem using Chaste with $\Delta x = 0.1$ mm. In a similar way as before, the efficiency of a scheme is assessed by determining (to one significant figure) the maximum constant time step size that yielded a numerical solution that approximately satisfied, but did not exceed, a 5% MRMS error tolerance for the transmembrane potential v.

In table 4.11, we report the maximum time step size for the second-order SM2 method and the third-order SS3, AKS3, and R3 methods. Table 4.11 also shows the actual MRMS error and the computational times, from which we notice that the third-order methods can use larger time steps than the second-order method while meeting the 5% MRMS error tolerance. This leads to SS3 and AKS3 being over 20% and 30% more efficient than SM2, respectively. Again, the AKS3 method is more efficient than the SS3 method by over 10%.

SM2			SS3			AKS3			R3		
Δt (ms)	$[\text{MRMS}]_v$	Time (s)	Δt (ms)	$[\text{MRMS}]_v$	Time (s)	Δt (ms)	$[\text{MRMS}]_v$	Time (s)	Δt (ms)	$[\text{MRMS}]_v$	Time (s)
3e−3	0.049	1031.32	1e−2	0.046	1132.27	7e−3	0.045	992.27	9e−3	0.047	612.89

Table 4.11: Efficiency comparison between the second- and third-order methods applied to the Niederer benchmark problem.

From the results in table 4.11 and 4.12, we notice that the R3 method is the most efficient OS method among all the OS methods considered in this study. In particular, when using the TTP cell model as in the original Niederer benchmark, the R3 method appears to between 15% and 25% more efficient than AKS3 and Y4, and between 50% and 60% more efficient than the methods with complex coefficients. For this problem, the methods with complex coefficients are not able to take a significantly larger step size for the accuracy requested. The additional expense per step associated with complex arithmetic alone then leads to their underperformance.

Y4			CCDV4			AK4		
Δt (ms)	$[\text{MRMS}]_v$	Time (s)	Δt (ms)	$[\text{MRMS}]_v$	Time (s)	Δt (ms)	$[\text{MRMS}]_v$	Time (s)
8e−2	0.048	830.32	9e−2	0.049	1957.85	8e−2	0.048	2036.82

Table 4.12: Efficiency comparison of the Y4, CCDV4, and AK4 OS methods on the Niederer benchmark problem.

4.2 Advection-Diffusion-Reaction Combustion model

Another set of experiments was conducted on an advection-diffusion-reaction (ADR) model describing the combustion of a chemical quantity c. In this section, we confirm the convergence rates of methods with complex coefficients and compare their performance with the R3 and Y4 methods. We show that, indeed, the methods with complex coefficients can take a larger stable step size while meeting a given accuracy criterion; however, their performance is generally not competitive on hardware that does not optimize for complex arithmetic.

Combustion simulations can be difficult to perform because they involve many coupled PDEs that describe the interactions between burning agents and the conservation of energy, mass, and momentum [EM11]. A simplified, one-dimensional version of a combustion model was presented in [BCP11], where the combustion

process is represented by the single PDE

$$\frac{\partial c}{\partial t} = \left[1 + U_0 \sin\left(\frac{\pi x}{L}\right)\right]\left(\frac{\gamma}{\beta}\frac{\partial^2 c}{\partial x^2} - \delta\frac{\partial c}{\partial x}\right) + f(c), \tag{4.9}$$

where $x \in [10, 50]$, $c = c(x, t)$ is the mass fraction of the reacted products, $U_0 \in [0, 1]$ is a parameter that controls density and velocity changes of the fluid domain, $L = 1$ is the length scale of the velocity field, the parameters $\gamma = 0.1$ and $\beta = 1$ represent the average diffusivities, and $\delta = 0, 1$ is a parameter that allows the advection to be switched off and on.

For our experiments, following [BCP11], we evaluate the model setting $U_0 = 0$ and by using the Fisher–Kolmogorov–Petrovskii–Piskunov (FKPP) ignition reaction type

$$f_{\text{FKPP}}(c) = \omega c(1 - c),$$

where $\omega = 0.1$ represents the rate of reaction. The boundary conditions are periodic, and the initial condition is given as

$$c(x, 0) = \exp\left[-\frac{(x - x_0)^2}{\sigma_0}\right],$$

with $x_0 = 20.5$ and $\sigma_0 = 10$. In all the experiments described below, we use $N = 2000$ uniformly distributed spatial points. The spatial discretization is performed using and first- and second-order central finite differences to discretize the first- and second-order derivatives in (4.9), respectively.

To assess the accuracy of some of the third- and fourth-order OS methods described in chapter 2, we use the MRMS defined in (4.7) using 2000 points in space and 30 points in time. The reference solution for this set of experiments is assumed to be accurate enough that it can serve as a proxy for the true solution. The order of convergence of the numerical solution p was then computed again in the standard way as per (4.7).

Reference solutions for the combustion problem were generated for decreasing tolerances until the numerical solution had converged to six matching decimal digits using a constant time step size of $\Delta t = 5e - 5$ and the ERK4 on the spatially discretized version of (4.9).

For the first set of experiments, we set $\delta = 1$, $\mathcal{A}^{[1]}(c) = \left(\frac{\gamma}{\beta}\frac{\partial^2 c}{\partial x^2} - \delta\frac{\partial c}{\partial x}\right)$, and $\mathcal{A}^{[2]}(c) = f_{\text{FKPP}}(c)$. The final time is set at $t_f = 15$ s.

Convergence of the third- and fourth-order methods is assessed using different combinations of time steps. To ensure the overall accuracy of OS methods with order $p \geq 3$, we need to approximate the flows of both sub-systems to at least third order in time. We choose to do so by integrating $\mathcal{A}^{[1]}(c)$ using the ERK3 method and by approximating the flow of $\mathcal{A}^{[2]}(c)$ using the SDIRK(2,3) method. At order four, we choose to integrate $\mathcal{A}^{[1]}(c)$ using the ERK4 method and by approximating the flow of $\mathcal{A}^{[2]}(c)$ using the SDIRK(3,4) method.

Results for the R3, C3, and AKS3CP (table 2.12) methods are reported in table 4.13, and those for the Y4 and CCDV4 methods are reported in table 4.14. The expected orders of convergence are confirmed for all the methods considered.

Next, we present the relative efficiency results by comparing the methods with complex coefficients against the R3 and Y4 methods, the most OS efficient methods with real coefficients. In particular, the Y4 OS method

	R3		C3		AKS3CP	
Δt (ms)	$[\text{MRMS}]_c$	p	$[\text{MRMS}]_c$	p	$[\text{MRMS}]_c$	p
1.5e−5	0.02578	–	0.03457	–	0.02876	–
1.0e−5	0.00736	3.24	0.01059	2.93	0.00762	3.24
8.0e−6	0.00992	2.84	0.00543	2.95	0.00397	2.92

Table 4.13: Convergence of the R3, C3, and AKS3CP methods applied to the ADR problem with the advection term ($\delta = 1$).

	Y4		CCDV4	
Δt (ms)	$[\text{MRMS}]_c$	p	$[\text{MRMS}]_c$	p
1.5e−5	0.03956	–	0.03952	–
1.0e−5	0.00766	4.05	0.00812	3.90
8.0e−6	0.00295	4.27	0.00317	4.19

Table 4.14: Convergence of the Y4 and CCDV4 methods applied to the ADR problem with the advection term ($\delta = 1$).

appears to be more efficient than the R3 method on this particular set of problems. We solve the same set of problems described above. In this case, we consider (4.9) in both cases where $\delta = 0$ and $\delta = 1$. We compared the CPU times obtained from using the maximum constant time step (correct to within 1e−3 ms) that yielded a numerical solution that did not exceed a 5% MRMS error tolerance for c. In the case where Δt was not commensurate with the time intervals on which the reference solution was stored, we calculated the MRMS_c error using interpolated values from a Hermite polynomial spline of the appropriate order. Efficiency results for the third- and fourth-order methods with complex coefficients considered in this study are reported in tables 4.15 and 4.16, respectively. In all except one of the cases, we observe that OS methods with complex coefficients can use a larger time step than the methods they are compared to while attaining a $[\text{MRMS}]_c$ that does not exceed 5%. However, the increase in step size does not lead to reduced CPU times, which appear to be much larger than the ones for the R3 and Y4 methods. This can be explained by the fact that on a standard computer, complex arithmetic normally takes approximately four times the number of flops as real arithmetic. The computational cost of complex arithmetic can be made comparable to that of real arithmetic by making use of an FPGA device [CS15]. In such an environment, methods with complex coefficients would outperform those with real coefficients due to the increased acceptable step sizes. Such an investigation, however, is beyond the scope of this thesis.

	R3			C3			AKS3CP		
	Δt (ms)	$[\text{MRMS}]_c$	Time (s)	Δt (ms)	$[\text{MRMS}]_c$	Time (s)	Δt (ms)	$[\text{MRMS}]_c$	Time (s)
$\delta = 0$	3.5e−3	0.0481	125.12	5.5e−3	0.0484	298.38	5.0e−3	0.0491	312.28
$\delta = 1$	2.5e−3	0.0493	138.43	4.0e−3	0.0489	407.52	4.0e−3	0.0486	361.82

Table 4.15: Efficiency comparison between the R3, C3, and AKS3CP methods applied to the ARD combustion problem without, ($\delta = 0$), and with, ($\delta = 1$), advection term. The table displays the maximum constant time step that maintained $[\text{MRMS}]_c \leq 5\%$.

	Y4			CCDV4		
	Δt (ms)	$[\text{MRMS}]_c$	Time (s)	Δt (ms)	$[\text{MRMS}]_c$	Time (s)
$\delta = 0$	4.0e−3	0.0492	96.71	5.0e−3	0.0494	409.54
$\delta = 1$	4.0e−3	0.0496	123.72	4.0e−3	0.0489	488.36

Table 4.16: Efficiency comparison between the Y4 and CCDV4 methods applied to the ARD combustion problem without, ($\delta = 0$), and with, ($\delta = 1$), advection term. The table displays the maximum constant time step that maintained $[\text{MRMS}]_c \leq 5\%$.

4.3 The Brusselator Problem

As explained in chapter 3, in [RS05], the authors explored the stability of the G1 and the combined-SM2 OS methods applied the Brusselator equations introduced in [PL68]. The Brusselator equations are used to model chemical dynamics and are given by (3.4).

Solutions can be found numerically by splitting (3.4) as (3.5). In [RS05], the authors solved (3.4) via OS. When using the G1 OS method, the authors obtained stable solutions over the entire Δt range considered ($\Delta t = 0.01, 0.05, 0.1, 0.25$). However, when using the combined-SM2 method, the authors found that the solution did not converge unless the time step was chosen to be sufficiently small (see [RS05] for more details).

Here, we reproduce and improve some results reported in [RS05]. We compute the reference solution using the parabolic and elliptic Matlab solver `pdepe` until there were 4 to 6 matching digits between successive approximations. Following the experiments in [RS05], we compute numerical solutions to (3.4) by using the G1 OS method and by integrating both the diffusion and the reaction terms using the backward Euler method. The numerical solution showed good convergence over the entire Δt range considered. In the next set of experiments, we solve (3.5) by using the combined-SM2 splitting paired with Heun's method to approximate the flow of both operators. As in [RS05], the solution obtained in this case showed instabilities for Δt. Figures 4.2 and 4.3 display the unstable solutions computed with the combined-SM2 splitting and $\Delta t = 0.25$ for the variables T and C, respectively, compared to the reference solution.

To understand and improve such oscillatory behaviours, we analyze the stability region for the case where the problem is solved using the combined-SM2 OS paired with Heun's method. Figure 4.4 shows

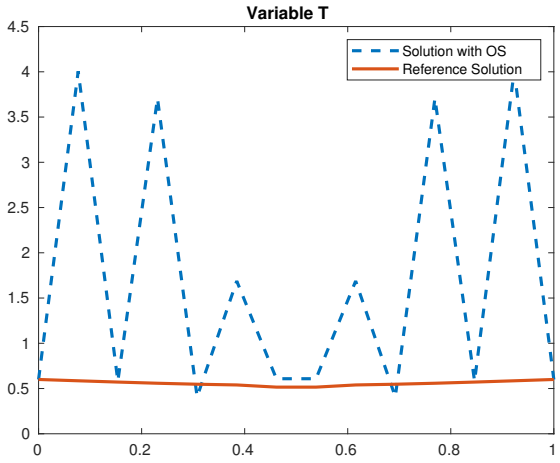

Figure 4.2: Solution of the variable T in the Brusselator problem solved with the combined-SM2 OS for $\Delta t = 0.25$. The reference solution is also plotted.

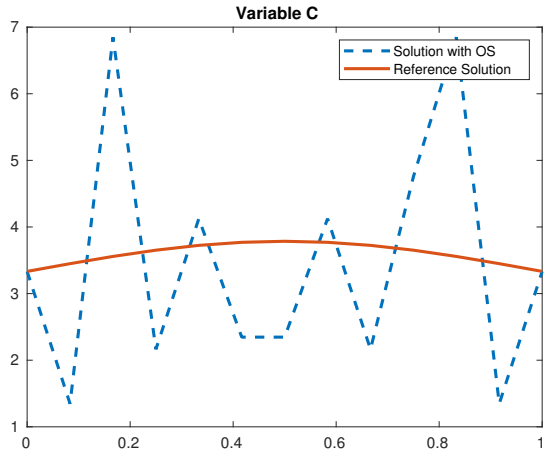

Figure 4.3: Solution of the variable C in the Brusselator problem solved with the combined-SM2 OS for $\Delta t = 0.25$. The reference solution is also plotted.

the eigenvalues of the reference solution and stability regions of the combined-SM2 OS when using Heun's method on both operators. We observe that not all the eigenvalues are captured by the stability region of the combined-SM2 OS and Heun's methods and, therefore, the solution is unstable for this time step.

We are able to eliminate the instabilities in the solution by using the SM2 OS method paired with Heun's method to integrate both operators. Figure 4.5 shows the stability region and the eigenvalues of the reference solution for this case. As shown, for the same Δt as in the previous experiment, now all the eigenvalues are captured by the stability region resulting in a stable numerical solution.

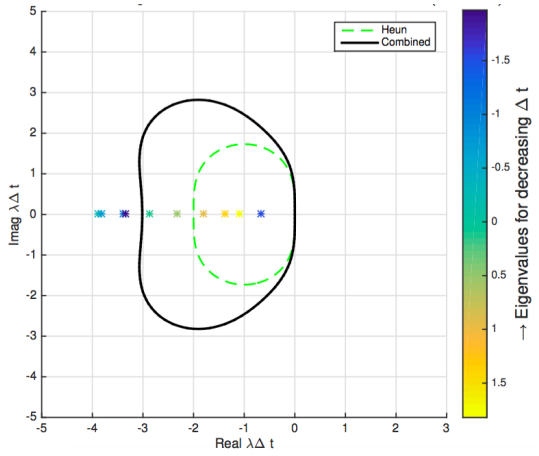

Figure 4.4: Eigenvalues of the reference solution and stability regions of the combined-SM2 OS when using Heun's method on both operators with $\Delta t = 0.25$.

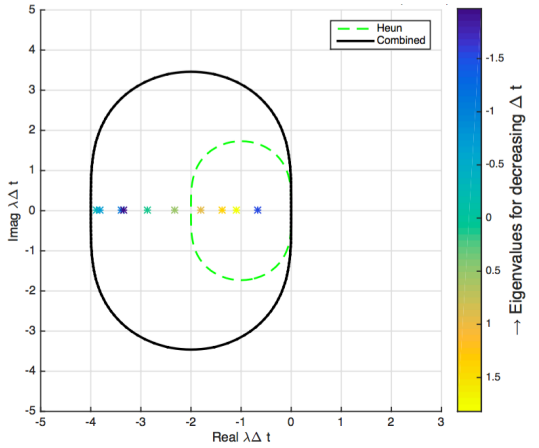

Figure 4.5: Eigenvalues of the reference solution and stability regions of the SM2 OS when using Heun's method on both operators with $\Delta t = 0.25$.

5 Conclusions and Future Work

5.1 Conclusions

Operator splitting methods represent a powerful strategy to solve a broad range of problems involving the solution of differential equations. Accordingly, the understanding of their properties is of paramount importance. In this thesis, we analyzed the accuracy, stability, and efficiency of various OS methods, focusing, in particular, on methods with order higher than two.

We studied the accuracy of some third- and fourth-order methods present in the literature on various problems describing cardiac electrophysiology and chemical combustion. In all our experiments, we showed the correct order of convergence for all the methods considered.

Having determined the proper order of accuracy for each method, we compared the efficiency performance of the higher-order methods compared to the "state-of-the-art" second-order SM2 methods. We showed that, despite the higher number of stages defining the higher-order OS methods, methods with order higher than two were able to take a larger time steps than second-order methods while attaining the same accuracy. This resulted in more efficient performance. In particular, we found that third-order methods can be 20% to 30% more efficient than the SM2 method when solving both cardiac electrophysiology problems and problems describing chemical combustion. We also considered the performance of OS methods with complex coefficients. In particular, we found that, although methods with complex coefficients are in some cases able to take a larger time step than methods with real coefficients of the same order, they are outperformed by OS methods with real coefficients due to the expense of complex arithmetic.

We observed that, when using OS methods together with Runge–Kutta type methods to advance the time integration, it is possible to construct an extended Butcher tableau that has the same structure as the ones defining Generalized Additive Runge–Kutta (GARK) methods. We defined this new class of methods to be OS-GARK methods. We applied linear stability analysis to OS-GARK methods and derived a stability function that can be simplified as the product of the stability functions of the Runge–Kutta methods used in the integration scaled by the coefficients of the OS methods.

Being able to analyze and plot the stability region is useful when trying to understand unstable numerical solutions. In fact, using an eigenvalue analysis, we were able to determine whether the stability region was capturing all the eigenvalues or not, resulting in stable or unstable numerical solutions. We assessed the validity of our stability analysis on various problems including the combustion problem and the Brusselator equations.

5.2 Future Work

In this thesis, we studied the accuracy, efficiency, and stability of higher-order OS methods and demonstrated their advantages over second-order methods on some classes of problems.

In principle, by solving the optimization problem by increasing the number of degrees of freedom and by adding order conditions to obtain an OS methods of desired order p, it is possible to solve for an OS method of any order. We leave the search for new OS methods for future work.

Because the stability function and, therefore, the shape of the stability region of OS-GARK methods depend on the coefficients of the OS method, we propose that it would be possible to derive coefficients of an OS method by maximizing the area of the stability region. This translates into solving a global optimization problem that has the area of the stability region as objective function, the coefficients of the new OS methods as degrees of freedom, and the order conditions as constraints.

Additionally, we must specify that, when solving the optimization problem to derive new OS methods, the objective function, the degrees of freedom, and the constraints could be set to be only dependent on the coefficients of the new OS methods. In other words, it is possible to optimize the area of the stability region for pre-determined Runge–Kutta integrators chosen to advance the time integration. This choice can be made to simplify the complexity of the optimization algorithm. In fact, the number of degrees of freedom and order conditions rapidly increases when allowing for various integrators of the desired order. Of course, the algorithm can be improved by not choosing a priori the Runge–Kutta integrators, but this comes at the expense of increasing the computational cost of the algorithm. We leave the implementation of the simplified and the improved version of the optimization algorithm that includes the coefficients of the Runge–Kutta methods as degrees of freedom and their order conditions as constraints for future work.

Bibliography

[AGC96] S. Abarbanel, D. Gottlieb, and M. H. Carpenter. On the removal of boundary errors caused by Runge–Kutta integration of nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 17(3):777–782, 1996.

[AH14] W. Auzinger and W. Herfort. Local error structures and order conditions in terms of Lie elements for exponential splitting schemes. *Opuscula Mathematica*, 34(2):243–255, 2014.

[AHHK16] W. Auzinger, W. Herfort, H. Hofstätter, and O. Koch. Setup of Order Conditions for Splitting Methods. In *Computer Algebra in Scientific Computing: 18th International Workshop, CASC 2016, Bucharest, Romania, September 19–23, 2016, Proceedings*, pages 30–42. Springer International Publishing, 2016.

[AHK19] W. Auzinger, H. Hofstätter, and O. Koch. Non-existence of generalized splitting methods with positive coefficients of order higher than four. *Applied Mathematics Letters*, 97:48–52, 2019.

[AHKK17] W. Auzinger, H. Hofstätter, D. Ketcheson, and O. Koch. Practical splitting methods for the adaptive integration of nonlinear evolution equations. Part I: Construction of optimized schemes and pairs of schemes. *BIT Numerical Mathematics*, 57(1):55–74, 2017.

[ALN+03] C. A. Autumn, C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, and P. J. Hunter. An overview of CellML 1.1, a biological model description language. *Simulation*, 79(12):740–747, 2003.

[AP98] U. M Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. SIAM, 1998.

[ARS97] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge–Kutta Methods for Time-dependent Partial Differential Equations. *Applied Numerical Mathematics*, 25(2-3):151–167, 1997.

[BC78] C. Bolley and M. Crouzeix. Conservation de la positivité lors de la discrétisation des problèmes d'évolution paraboliques. *RAIRO. Analyse numérique*, 12(3):237–245, 1978.

[BC05] S. Blanes and F. Casas. On the necessity of negative coefficients for operator splitting schemes of order higher than two. *Applied Numerical Mathematics*, 54(1):23–37, 2005.

[BCF+13] S. Blanes, F. Casas, A. Farres, J. Laskar, J. Makazaga, and A. Murua. New families of symplectic splitting methods for numerical integration in dynamical astronomy. *Applied Numerical Mathematics*, 68:58–72, 2013.

[BCM08] S. Blanes, F. Casas, and A. Murua. Splitting and composition methods in the numerical integration of differential equations. *arXiv preprint :0812.0377*, 2008.

[BCM10] S. Blanes, F. Casas, and A. Murua. Splitting methods with complex coefficients. *SeMA Journal*, 50(1):47–60, 2010.

[BCP11] F. Bianco, S. Chibbaro, and R. Prudh́omme. Etude dúne équation de convection-réaction-diffusion en écoulement compressible. *arXiv preprint 1104.1052*, 2011.

[BJM02] W. Bao, S. Jin, and P. A. Markowich. On time-splitting spectral approximations for the Schrödinger equation in the semiclassical regime. *Journal of Computational Physics*, 175(2):487–524, 2002.

[BM02] S. Blanes and P. C. Moan. Practical Symplectic Partitioned Runge–Kutta and Runge–Kutta–Nyström Methods. *Journal of Computational and Applied Mathematics*, 142(2):313–330, 2002.

[BV19] G. Bertoli and G. Vilmart. Strang splitting method for semilinear parabolic problems with inhomogeneous boundary conditions: a correction based on the flow of the nonlinearity. *arXiv preprint 1904.08826*, 2019.

[CCDV09] F. Castella, P. Chartier, S. Descombes, and G. Vilmart. Splitting methods with complex times for parabolic equations. *BIT Numerical Mathematics*, 49(3):487–508, 2009.

[CFH05] P. Csomós, I. Faragó, and Á Havasi. Weighted sequential splittings and their analysis. *Computers & Mathematics with Applications*, 50(7):1017–1031, 2005.

[CGA94] M. H. Carpenter, D. Gottlieb, and S. Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes. *Journal of Computational Physics*, 111(2):220–236, 1994.

[CGAD95] M. H. Carpenter, D. Gottlieb, S. Abarbanel, and W. Don. The theoretical accuracy of Runge–Kutta time discretizations for the initial boundary value problem: a study of the boundary error. *SIAM Journal on Scientific Computing*, 16(6):1241–1252, 1995.

[Cha03] J.E. Chambers. Symplectic integrators with complex time steps. *The Astronomical Journal*, 126(2):1119, 2003.

[Chi05] S. A. Chin. Structure of positive decompositions of exponential operators. *Physical Review E*, 71(1):016703, 2005.

[CHMM78] A. Chorin, T. Hughes, M. McCracken, and J. Marsden. Product formulas and numerical algorithms. *Communications on Pure and Applied Mathematics*, 31(2):205–256, 1978.

[CLX15] A. J Christlieb, Y. Liu, and Z. Xu. High order operator splitting methods based on an integral deferred correction framework. *Journal of Computational Physics*, 294:224–242, 2015.

[CMAC+15] C. D. Cantwell, D. Moxey, A. Bolis A. Comerford, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R.M. Kirby, and S. J. Sherwin. Nektar++: An open-source spectral/hp element framework . *Computer Physics Communications*, 192:205–219, 2015.

[Cro78] M. Crouzeix. *Sur l'approximation des équations différentielles opérationelles linéaires par des méthodes de Runge–Kutta*. PhD thesis, Université Paris, 1978.

[CS15] M. Czyżak and R. Smyk. FPGA computation of magnitude of complex numbers using modified CORDIC algorithm. *Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej*, 2015.

[CS18a] J. Cervi and R. J. Spiteri. High-order operator splitting for the bidomain and monodomain models. *SIAM Journal on Scientific Computing*, 40(2):A769–A786, 2018.

[CS18b] J. Cervi and R. J. Spiteri. High-order operator-splitting methods for the bidomain and monodomain models. In *Mathematical and Numerical Modeling of the Cardiovascular System and Applications*, pages 23–40. Springer, 2018.

[CS19] J. Cervi and R. J Spiteri. A comparison of fourth-order operator splitting methods for cardiac simulations. *Applied Numerical Mathematics*, 145:227–235, 2019.

[Des01] S. Descombes. Convergence of a Splitting Method of High Order for Reaction-Diffusion Systems. *Mathematics of Computation*, 70(236):1481–1501, 2001.

[DK01] J. Douglas and S. Kim. Improved accuracy for locally one-dimensional methods for parabolic equations. *Mathematical Models and Methods in Applied Sciences*, 11(09):1563–1579, 2001.

[DR56] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[DRFHZ14] D. C. Del Rey Fernández, J. E. Hicken, and D. W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Computers & Fluids*, 95:171 – 196, 2014.

[DS98] S. Descombes and M. Schatzman. On Richardson extrapolation of Strang formula for reaction-diffusion equations. *Equations aux Dérivées Partielles et Applications, articles dédiés à Jacques-Louis Lions*, 1998.

[Dur10] D. R. Durran. *Numerical methods for fluid dynamics*, volume 32 of *Texts in Applied Mathematics*. Springer, New York, second edition, 2010.

[EB08] M. Ethier and Y. Bourgault. Semi-implicit time-discretization schemes for the bidomain model. *SIAM Journal on Numerical Analysis*, 46(5):2443–2468, 2008.

[EK01] L. Edelstein-Keshet. Mathematical models of swarming and social aggregation. In *Proceedings of the 2001 International Symposium on Nonlinear Theory and Its Applications, Miyagi, Japan*, pages 1–7, 2001.

[EM11] T. Echekki and E. Mastorakos. Turbulent combustion: concepts, governing equations and modeling strategies. In *Turbulent Combustion Modeling*, pages 19–39. Springer, 2011.

[EO15] L. Einkemmer and A. Ostermann. Overcoming order reduction in diffusion-reaction splitting. part 1: Dirichlet boundary conditions. *SIAM Journal on Scientific Computing*, 37(3):A1577–A1592, 2015.

[EO16] L. Einkemmer and A. Ostermann. Overcoming order reduction in diffusion-reaction splitting. part 2: Oblique boundary conditions. *SIAM Journal on Scientific Computing*, 38(6):A3741–A3757, 2016.

[Fit66] R. FitzHugh. Mathematical models of excitation and propagation in nerve. *Biological Engineering, McGraw-Hill Book Co*, pages 1–85, 1966.

[FR90] E. Forest and R. D. Ruth. Fourth-order symplectic integration. *Physica D: Nonlinear Phenomena*, 43(1):105 – 117, 1990.

[GG16] L. Geraldo-Giorda. *Nonlinear Dynamics in Biological Systems*. Springer International Publishing, 2016.

[GK96] G. Goldman and T. J. Kaper. Nth-order operator splitting schemes and nonreversible systems. *SIAM Journal on Numerical Analysis*, 33(1):349–367, 1996.

[GMS06] J. L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Comput. Methods Appl. Mech. Engrg.*, 195(44-47):6011–6045, 2006.

[God59] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959.

[God99] S. K. Godunov. Reminiscences about difference schemes. *Journal of Computational Physics*, 153(1):6–25, 1999.

[GOY16] R. Glowinski, S. J. Osher, and W. Yin, editors. *Splitting methods in communication, imaging, science, and engineering*. Scientific Computation. Springer, Cham, 2016.

[GPH+01] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Périaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of Computational Physics*, 169(2):363–426, 2001.

[GV02] A. Gerisch and J. G. Verwer. Operator splitting and approximate factorization for taxis-diffusion-reaction models. *Applied Numerical Mathematics*, 42(1-3):159–176, 2002. Ninth Seminar on Numerical Solution of Differential and Differential-Algebraic Equations (Halle, 2000).

[GW96] C. Gao and Y. Wang. A general formulation of peaceman and rachford adi method for the n-dimensional heat diffusion equation. *International communications in heat and mass transfer*, 23(6):845–854, 1996.

[HKO12] E. Hansen, F. Kramer, and A. Ostermann. A second-order positivity preserving scheme for semilinear parabolic problems. *Applied Numerical Mathematics*, 62(10):1428–1435, 2012.

[HLW06] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[HR94] L. Hatton and A. Roberts. How Accurate is Scientific Software? *IEEE Transactions in Software Engineering*, 20(10):785–797, 1994.

[HV03] W. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33. Springer-Verlag, Berlin, 2003.

[IT08] S. Ikonen and J. Toivanen. Efficient numerical methods for pricing american options under stochastic volatility. *Numerical Methods for Partial Differential Equations: An International Journal*, 24(1):104–126, 2008.

[KC03] C. A. Kennedy and M. H. Carpenter. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics*, 44(1-2):139–181, 2003.

[KIFS10] Á. Kriston, G. Inzelt, I. Faragó, and T. Szabó. Simulation of the transient behavior of fuel cells by using operator splitting techniques for real-time applications. *Computers & Chemical Engineering*, 34(3):339 – 348, 2010.

[KNT13] O. Koch, C. Neuhauser, and M. Thalhammer. Embedded exponential operator splitting methods for the time integration of nonlinear evolution equations. *Applied Numerical Mathematics*, 63:14–24, 2013.

[Kos80] Y. Kosaku. *Functional analysis*, volume 6. Springer-Verlag, Berlin New York, 1980.

[Kut01] W. Kutta. Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:435–453, 1901.

[LE70] S. Lie and F. Engel. *Theorie der transformationsgruppen (Vol I)*. American Society, Providence, 1970.

[LeV92] R. J. LeVeque. *Numerical methods for conservation laws*. Birkhauser, Basel, 1992.

[LM79] P. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[LR91] C. H. Luo and Y. Rudy. A model of the ventricular cardiac action potential. depolarization, repolarization, and their interaction. *Circulation Research*, 68(6):1501–1526, 1991.

[LW64] P. D. Lax and B. Wendroff. Difference schemes for hyperbolic equations with high order of accuracy. *Communications on Pure and Applied Mathematics*, 17(3):381–398, 1964.

[MA92] R. I. McLachlan and P. Atela. The accuracy of symplectic integrators. *Nonlinearity*, 5(2):541–562, 1992.

[MAB+13] G. R. Mirams, C. J. Arthurs, M. O. Bernabeu, R. Bordas, J. Cooper, A. Corrias, Y. Davit, S. Dunn, A. G. Fletcher, D. G Harvey, et al. Chaste: An open source C++ library for computational physiology and biology. *PLoS Comput. Biol.*, 9(3), 2013.

[Mar71] G. I. Marchuk. On the theory of the splitting-up method. In *Numerical Solution of Partial Differential Equations-II*, pages 469 – 500. Academic Press, 1971.

[Mar90] G. I. Marchuk. Splitting and alternating direction methods. In *Handbook of Numerical Analysis, Vol. I*, Handb. Numer. Anal., I, pages 197–462. North-Holland, Amsterdam, 1990.

[McL95] R. I. McLachlan. On the numerical integration of ordinary differential equations by symmetric composition methods. *SIAM Journal on Scientific Computing*, 16(1):151–168, 1995.

[MQ02] R. I. McLachlan and G. R. W. Quispel. Splitting methods. *Acta Numerica*, 11:341–434, 2002.

[Mur03] J. D. Murray. *Mathematical biology. II*, volume 18 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, New York, third edition, 2003.

[MZS12] M. E. Marsh, S. Ziaratgahi, and R. J. Spiteri. The Secrets to the Success of the Rush–Larsen Method and its Generalizations. *IEEE Transactions on Biomedical Engineering*, 59(9):2506–2515, 2012.

[Ner88] F. Neri. Lie Algebras and Canonical Integration. Technical report, Department of Physics, University of Maryland, 1988.

[NKB+11] S. A. Niederer, E. Kerfoot, A. P. Benson, M. O. Bernabeu, O. Bernus, C. Bradley, E. M. Cherry, R. Clayton, F. H. Fenton, A. Gary, et al. Verification of cardiac tissue electrophysiology simulators using an N-version benchmark. *Philosophical Transactions of the Royal Society A*, 369(1954):4331–4351, 2011.

[Nor74] S. P. Norsett. *Semi explicit Runge–Kutta methods*. Matematisk Institut, Universitet I, 1974.

[OB05] E. S. Oran and J. P. Boris. *Numerical Simulation of Reactive Flow*. Cambridge University Press, 2005.

[OMF02] I. P. Omelyan, I. M. Mryglod, and R. Folk. Construction of high-order force-gradient algorithms for integration of motion in classical and quantum systems. *Physical Review E*, 66(2):026701, 2002.

[PL68] I. Prigogine and R. Lefever. Symmetry breaking instabilities in dissipative systems. ii. *The Journal of Chemical Physics*, 48(4):1695–1700, 1968.

[PMSW11] P. Pathmanathan, G. R. Mirams, J. Southern, and J. P. Whiteley. The significant effect of the choice of ionic current integration method in cardiac electro-physiological simulations. *International Journal for Numerical Methods in Biomedical Engineering*, 27(11):1751–1770, 2011.

[PR55] D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, 3(1):28–41, 1955.

[QG99] Z. Qu and A. Garfinkel. An advanced algorithm for solving partial differential equation in cardiac conduction. *IEEE Transactions on Biomedical Engineering*, 46(9):1166–1168, 1999.

[Ric11] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society A*, 210(459-470):307–357, 1911.

[RS05] D. L. Ropp and J. N. Shadid. Stability of operator splitting methods for systems with indefinite operators: reaction-diffusion systems. *Journal of Computational Physics*, 203(2):449–466, 2005.

[RS09] D. L. Ropp and J. N. Shadid. Stability of operator splitting methods for systems with indefinite operators: Advection–diffusion–reaction systems. *Journal of Computational Physics*, 228(9):3508–3516, 2009.

[RS11] J. A. Rossmanith and D. C. Seal. A positivity-preserving high-order semi-Lagrangian discontinuous Galerkin scheme for the Vlasov–Poisson equations. *Journal of Computational Physics*, 230(16):6203–6232, 2011.

[Rut83] R. D. Ruth. A canonical integration technique. *IEEE Transactions on Nuclear Science*, 30:2669–2671, 1983.

[San01] A. Sandu. Positive numerical integration methods for chemical kinetic systems. *Journal of Computational Physics*, 170(2):589–602, 2001.

[SD10] R. J. Spiteri and R. C. Dean. Stiffness analysis of cardiac electrophysiological models. *Annals of Biomedical Engineering*, 38(12):3592–3604, 2010.

[SG15] A. Sandu and M. Günther. A generalized-structure approach to additive runge–kutta methods. *SIAM Journal on Numerical Analysis*, 53(1):17–42, 2015.

[She89] Q. Sheng. Solving linear partial differential equations by exponential splitting. *IMA Journal of Numerical Analysis*, 9:199–212, 1989.

[SLC06] J. Sundnes, G. T. Lines, and X. Cai. *Computing the Electrical Activity in the Heart*. Springer, 2006.

[SLT05] J. Sundnes, G. T. Lines, and A. Tveito. An operator splitting method for solving the bidomain equations coupled to a volume conductor model for the torso. *Mathematical Biosciences*, 194(2):233–248, 2005.

[SN92] J. M. Stone and M. L. Norman. ZEUS-2D: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. I – The hydrodynamic algorithms and tests. *The Astrophysical Journal Supplement Series*, 80:753–790, 1992.

[Sor07] A. T. Sornborger. Higher-order operator splitting methods for deterministic parabolic equations. *International Journal of Computer Mathematics*, 84(6):887–893, 2007.

[SPVW09] J. A. Southern, G. Plank, E. J. Vigmond, and J. P. Whiteley. Solving the coupled system improves computational efficiency of the bidomain equations. *IEEE Transactions on Biomedical Engineering*, 56(10):2404–2412, 2009.

[SS99] A. T. Sornborger and E. D. Stewart. Higher-order methods for simulations on quantum computers. *Physical Review A*, 60(3):765–789, 1999.

[Str68] G. Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[STZ16] R. J. Spiteri and S. Torabi Ziaratgahi. Operator splitting for the bidomain model revisited. *Journal of Computational and Applied Mathematics*, 296:550–563, 2016.

[Suz91] M. Suzuki. Solving linear partial differential equations by exponential splitting. *IMA Journal of Numerical Analysis*, 9:400–407, 1991.

[SWL+15] L. Sun, Z. Wu, J. Liu, L. Xiao, and Z. Wei. Supervised spectral–spatial hyperspectral image classification with weighted markov random fields. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1490–1503, 2015.

[Tha08] M. Thalhammer. High-order exponential operator splitting methods for time-dependent Schrödinger equations. *SIAM Journal on Numerical Analysis*, 46(4):2022–2038, 2008.

[Tro59] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[TSL00] R. Tyson, L. G. Stern, and R. J. LeVeque. Fractional step methods applied to a chemotaxis model. *J. Math. Biol.*, 41(5):455–475, 2000.

[tTP06] K. H. ten Tusscher and A. V. Panfilov. Alternans and spiral breakup in a human ventricular tissue model. *American Journal of Physiology - Heart and Circulatory Physiology*, 291(3):H1088–H1100, 2006.

[Tun78] L. Tung. *A bi-domain model for describing ischemic myocardial d-c potentials.* PhD thesis, Massachusetts Institute of Technology, 1978.

[TZ15] S. Torabi Ziaratgahi. *Efficient Numerical Methods for Heart Simulation.* PhD thesis, University of Saskatchewan, Canada, 2015.

[UD06] A. Usadi and C. Dawson. 50 Years of ADI Methods: Celebrating the Contributions of Jim Douglas, Don Peaceman, and Henry Rachford. *SIAM News*, 39(2), 03 2006.

[Wac13] E. Wachspress. *The ADI model problem.* Springer, 2013.

[Wan18] J. Wang. *Operator Splitting Method and Its Applications in Quantitative Finance.* PhD thesis, State University of New York at Stony Brook, 2018.

[Yan71] N. N. Yanenko. *The method of fractional steps. The solution of problems of mathematical physics in several variables.* Springer-Verlag, New York-Heidelberg, 1971. Translated from the Russian by T. Cheron. English translation edited by M. Holt.

[Yos90] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5):262–268, 1990.

[ZMSS14] S. T. Ziaratgahi, M. E. Marsh, J. Sundnes, and R. J. Spiteri. Stable time integration suppresses unphysical oscillations in the bidomain model. *Frontiers in Physics*, 2:1–9, 2014.