2019

Higher-order Runge--Kutta type schemes based on the Method of Characteristics for hyperbolic equations with crossing characteristics

Jeffrey Steven Jewell
University of Vermont

Higher-order Runge–Kutta type schemes based on the Method of Characteristics for hyperbolic equations with crossing characteristics

A Thesis Presented

by

Jeffrey S. Jewell

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Mathematics

May, 2019

Defense Date: March 19, 2019
Thesis Examination Committee:

Taras I. Lakoba, Ph.D., Advisor
Jeffrey S. Marshall, Ph.D., Chairperson
Jianke Yang, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

Abstract

The Method of Characteristics (MoC) is a well-known procedure used to find the numerical solution of systems of hyperbolic partial differential equations (PDEs). The main idea of the MoC is to integrate a system of ordinary differential equations (ODEs) along the characteristic curves admitted by the PDEs. In principle, this can be done by any appropriate numerical method for ODEs. In this thesis, we will examine the MoC applied to systems of hyperbolic PDEs with straight-line and crossing characteristics. So far, only first- and second-order accurate explicit MoC schemes for these types of systems have been reported. As such, the purpose of this thesis is to develop MoC schemes which are of an order greater than two.

The order of the global truncation error of an MoC scheme goes hand-in-hand with the order of the ODE solver used. The MoC schemes which have already been developed use the first-order Simple Euler (SE) and second-order Modified Euler (ME) methods as the ODE solvers. The SE and ME methods belong to a larger family of numerical methods for ODEs known as the Runge–Kutta (RK) methods. First, we will attempt to develop third- and fourth-order MoC schemes by using the classical third- and fourth-order RK methods as the ODE solver. We will show that the resulting MoC schemes can be strongly unstable, meaning that the error in the numerical solution becomes unbounded rather quickly.

We then turn our attention to the so-called pseudo-RK (pRK) methods for ODEs. The pRK methods are at the intersection of RK and multistep methods, and a variety of third- and fourth-order schemes can be constructed. We show that when certain pRK schemes are used in the MoC, at most a weak instability, or no instability at all, is present, and thus the resulting methods are suitable for long-time computations. Finally, we present some numerical results confirming that the MoC using third- and fourth-order pRK schemes have the desired accuracy.

Acknowledgement

I would like to thank my advisor, Dr. Taras I. Lakoba, for his guidance throughout my tenure in the graduate program at UVM. From teaching me the basics of numerical analysis to conquering the adversity of the MoC-pRK4 algorithm (see Chapter 6), the patience, resolve, and wisdom of Dr. Lakoba has never wavered.

Table of Contents

List of Figures

List of Tables

1 Introduction

The purpose of this thesis is to develop higher-order numerical methods used to solve hyperbolic partial differential equations. The numerical schemes developed will follow a framework which applies Runge–Kutta methods to the well-known Method of Characteristics.

1.1 Hyperbolic partial differential equations and characteristics

First, we introduce the concept of a hyperbolic partial differential equation (PDE).

The general quasilinear second-order nonhomogeneous PDE in two variables is given by

$$Au_{tt} + Bu_{tx} + Cu_{xx} + Du_t + Eu_x + Fu = G, \tag{1.1}$$

where A, B, C are coefficients which may depend on t, x, u, u_t, u_x, D, E, F are coefficients which may depend on t, x, u, and G is a nonhomogeneous term which may depend on t, x. In the context of this thesis, t and x are the time and space variables.

The characteristics of a PDE are the curves in the solution space over which information propagates. For PDEs of the form (1.1), a differential equation for the characteristics is given by

$$\frac{dx}{dt} = \frac{B \pm \sqrt{B^2 - 4AC}}{2A}. \tag{1.2}$$

PDEs of the form (1.1) are classified as elliptic, parabolic, or hyperbolic. This nomenclature is in analogy with the usual conic sections of Euclidean geometry. The classification is based on the sign of the discriminant, $\sqrt{B^2 - 4AC}$. If the discriminant is positive, then the PDE is hyperbolic. Therefore, according to (1.2), hyperbolic PDEs have real and distinct characteristics.

A classic and simple example of a hyperbolic PDE is the wave equation,

$$u_{tt} - c^2 u_{xx} = 0, \tag{1.3}$$

where c is a positive constant usually having the meaning of the wave's speed. In (1.3), $A = 1$, $B = 0$, and $C = -c^2$, and therefore by (1.2), a differential equation for the characteristics is given by

$$\frac{dx}{dt} = \pm c. \tag{1.4}$$

Integrating (1.4), one obtains the characteristics of (1.3),

$$x - ct = \xi_1 \quad \text{and} \quad x + ct = \xi_2, \tag{1.5}$$

where $\xi_{1,2}$ are some constants. More complicated systems of hyperbolic PDEs arise frequently in the natural sciences. A form which occurs often is:

$$u_{1t} + cu_{1x} = f_1(u_1, u_2) \tag{1.6a}$$

$$u_{2t} - cu_{2x} = f_2(u_1, u_2), \tag{1.6b}$$

where $f_{1,2}$ are some differentiable functions. In this case, it can be shown that the

characteristics are also given by (1.5). Note that the characteristics (1.5) are **crossing**; see Fig. 2.1. This fact will play a key role in motivating the research undertaken in this thesis.

Let us briefly comment upon the physical relevance of systems with straight-line characteristics. A large body of problems which admit straight-line characteristics have been reviewed in [1]. In Sec. 1.6, we will introduce two such systems which will be extensively analyzed in this thesis.

1.2 Method of Characteristics

Now equipped with the knowledge of hyperbolic PDEs and characteristics, we introduce the Method of Characteristics (MoC), a widely used method for solving them. We describe the MoC here with a simple example, with only one characteristic (as opposed to two, as in (1.4) and (1.5)), which still involves two stages.

1.2.1 Homogeneous case of the MoC

At the first stage, consider a homogeneous hyperbolic PDE

$$w_t + cw_x = 0, \tag{1.7}$$

where c is a positive constant, with initial and boundary conditions given by

$$w(x, t = 0) = \phi(x), \quad x \geq 0,$$
$$w(x, = 0, t) = g(t), \quad t \geq 0.$$

The general solution of (1.7) is

$$w(x,t) = w(x - ct), \tag{1.8}$$

and the characteristic is

$$x - ct = \xi, \tag{1.9}$$

where ξ is constant. To arrive at (1.8), one makes the change of variables

$$(x,t) \to (\xi = x - ct, t). \tag{1.10}$$

Applying the Chain Rule to (1.7) under change of variables (1.10), one obtains:

$$\left(-c\frac{\partial}{\partial \xi} + \frac{\partial}{\partial t}\right) w + c\frac{\partial}{\partial \xi} w = 0.$$

This implies

$$w_t(\xi, t) = 0, \tag{1.11}$$

and thus the solution of (1.7) does not change along the characteristic (1.9).

Let us show how a numerical solution of (1.7) given by (1.8) can be computed on the grid

$$x_m = mh,\ t_n = n\Delta t,\ m, n = 0, 1, 2, \ldots, \tag{1.12}$$

where $h = c\Delta t$ (see Fig. 1.1). Let W_m^n denote the numerical solution of (1.7) at the

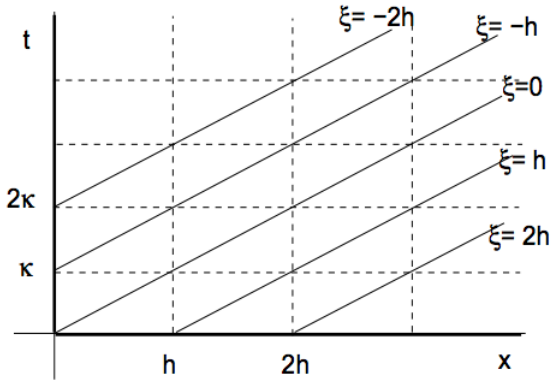

Figure 1.1: Grid given by (1.12)

point (m, n) on the grid. Thus, the initial and boundary condition are

$$W_m^0 = \phi(x_m), \quad m \geq 0, \tag{1.13a}$$

$$W_0^n = g(t_n), \quad n \geq 0, \tag{1.13b}$$

and by (1.8)

$$W_m^n = \begin{cases} W_{m-n}^0 = \phi(x_{m-n}), & m \geq n, \\ W_0^{n-m} = g(t_{n-m}), & n \geq m. \end{cases} \tag{1.14}$$

This scheme is the MoC used to solve (1.7).

1.2.2 Non-homogeneous case of the MoC

Now, consider the non-homogeneous case of (1.7),

$$w_t + cw_x = f(x, t, w), \tag{1.15}$$

where c is constant and $f(x, t, w) \neq 0$. In this case, following the derivation of (1.11), one arrives at

$$w_t = f(\xi, t, w), \tag{1.16}$$

and so the solution **will** change along the characteristic. When solving (1.16), one must treat ξ as a constant parameter, because in the variables (ξ, t), one keeps ξ constant when evaluating $\partial_t w$.

Therefore, one arrives at the numerical scheme

$$W_m^n = W_{m-1}^{n-1} + \int_{(n-1)\Delta t}^{\Delta t} f(\xi, t, w(\xi, t))\, dt,\ n \geq 1,\ m \geq 0, \tag{1.17}$$

where the initial and boundary condition are identical to (1.13).

In scheme (1.17), the expression

$$\int_{(n-1)\Delta t}^{\Delta t} f(\xi, t, w(\xi, t))\, dt$$

is merely a symbol that denotes the numerical integration of (1.16). This integration can be performed by any appropriate numerical method for ODEs. The notation $w(\xi, t)$ refers to the solution along the characteristic with a constant value of ξ.

1.2.3 The MoC for systems with curved characteristics

Suppose we wish to generalize the approach leading to schemes (1.14) and (1.17) for a system of two PDEs with intersecting families of curved characteristics (see Fig. 1.2).

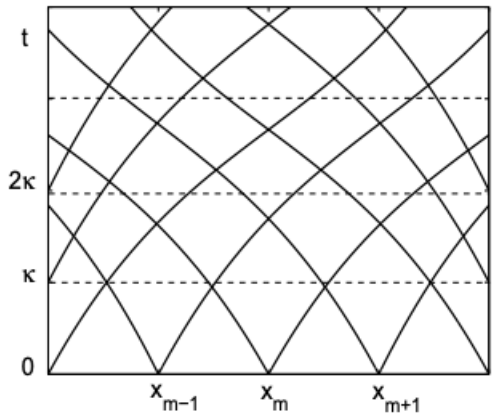

Figure 1.2: MoC stencil for system with intersecting and curved characteristics

Suppose we choose the characteristics to intersect at time level $t = 0$. Then, the intersection points are x_{m-1}, x_m, x_{m+1}, etc. A problem arises because the characteristics **do not** intersect at subsequent time levels, as was assumed in schemes (1.14) and (1.17). This greatly complicates the development of an MoC scheme. We will not examine the case of curved characteristics in this thesis, but it is an avenue to explore in future research.

1.3 Benefits and Shortcomings of the MoC

Now understanding the general idea behind the MoC, we discuss its main benefits and shortcomings.

1.3.1 Benefits: numerical dispersion and the MoC

One of the **benefits** of the MoC is that it preserves what is known as the linear dispersion relation of equations like (1.3) and (1.6). Any process f developing in time consists of Fourier harmonics that follow the equation

$$\widehat{f}(x,t) = e^{i\omega t - ikx}, \tag{1.18}$$

where ω is frequency and k is the wavenumber of the harmonic. The relationship between ω and k is called the **dispersion relation**. In equations of the form (1.6) (which the MoC is used to solve), such as the so-called hyperbolic convection equation,

$$u_t + cu_x = 0, \tag{1.19}$$

by (1.18) one has

$$\omega = ck, \tag{1.20}$$

which we call the **linear dispersion relation**. Substituting (1.20) into (1.18), one sees that

$$\widehat{u}(x,t) = e^{-ik(x-ct)}. \tag{1.21}$$

In (1.21), if c is the speed at which the wave moves, the equation shows that when the displaced velocity is linear, each harmonic is simply shifted by the same amount, ct, regardless of k. This, however, is not guaranteed to be the case if one solves (1.19)

by another method, as we will illustrate now.

As described in [2], a well-known scheme used to solve (1.19) is the Lax–Wendroff Method:

$$u_m^{n+1} = u_m^n - \frac{\gamma}{2}(u_{m+1}^n - u_{m-1}^n) + \frac{\gamma^2}{2}(u_{m+1}^n - 2u_m^n + u_{m-1}^n), \tag{1.22a}$$

where γ ($\equiv O(1)$) is the so-called *convection number*, given by

$$\gamma = \frac{c\Delta t}{h}. \tag{1.22b}$$

We provide the derivation of method (1.22) in Appendix A. Expanding (1.22a) in a Taylor series and simplifying, one obtains:

$$u_t + cu_x = \left(-\frac{1}{6}ch^2 + \frac{1}{6}c^3\Delta t^2\right) u_{xxx} + O(h^3) + O(\Delta t^3). \tag{1.23}$$

Therefore, if one uses the Lax–Wendroff Method to solve (1.19), one is actually solving the equation

$$u_t + cu_x = \beta u_{xxx}, \tag{1.24}$$

where β is a constant given in (1.23). Using (1.18), one can show that the dispersion relation is then given by

$$\omega = ck - \beta k^3, \tag{1.25}$$

and thus

$$\hat{u}(x,t) = e^{-ik(x-ct+\beta k^2 t)}. \tag{1.26}$$

In this case, the shift **does** depend on wavenumber, when, according to (1.21), it should **not**. This results in **numerical dispersion**.

We will now show how numerical dispersion can be damaging to a solution. Suppose one has a Gaussian pulse centered at $x = 0$ and propagating with speed $c = 1$. We compute the position and shape of the pulse using the Lax–Wendroff Method, taking $h = 0.1$ and $\Delta t = 0.05$, and implementing periodic boundary conditions. Numerical results are shown in Fig. 1.3. One can see that the numerical solution quickly becomes out of phase with the exact solution (as early as $t = 25$), which is the consequence of numerical dispersion. By $t = 200$, in addition to being out of phase, the numerical solution produces a distinctive "wiggle," a common feature of numerical dispersion. If one were to implement a long-time simulation, the numerical solution would be completely destroyed due to numerical dispersion.

Fortunately, numerical dispersion is avoided when one uses the MoC to solve a PDE, since the MoC preserves the linear dispersion relation. This ability of the MoC will be discussed later on.

1.3.2 Shortcomings: accuracy of MoC schemes

The main **shortcoming** of the MoC is that only first- and second order-accurate **explicit** methods have been reported thus far for the case of two crossing characteristics. That is, the global error of the MoC scheme is $O(\Delta t)$ or $O(\Delta t^2)$. Higher-order

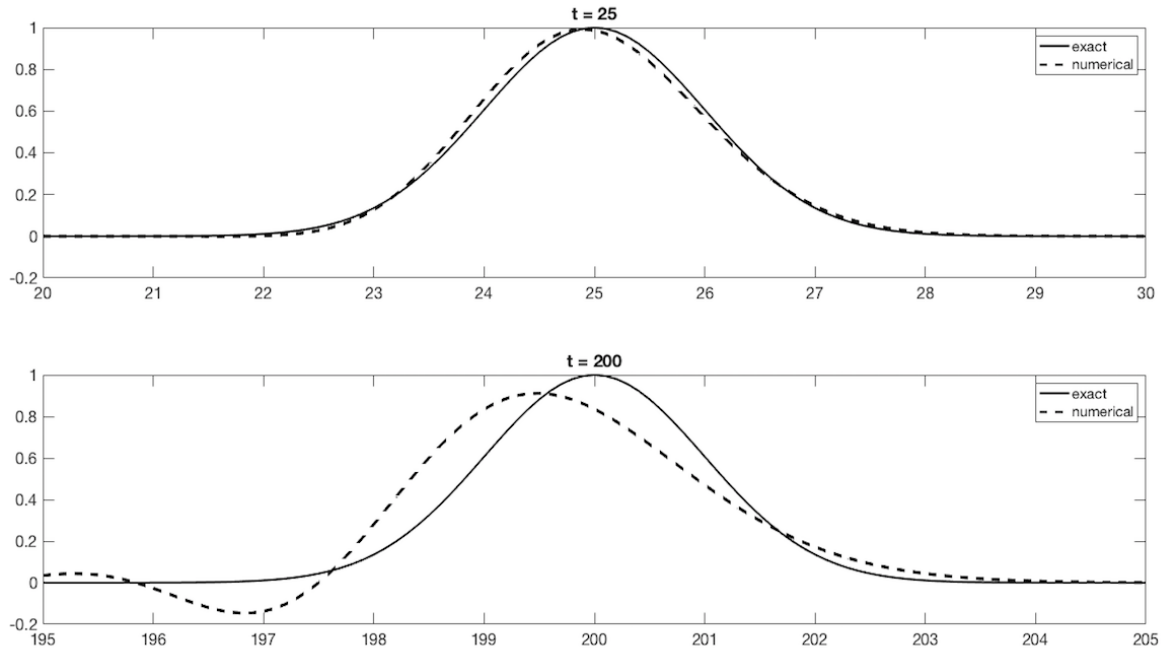

Figure 1.3: Integration of Gaussian pulse by Lax–Wendroff

implicit methods exist (see e.g. [3]), but these are more difficult to implement and less computationally efficient. Therefore, **the objective of this thesis** is to develop explicit MoC schemes for systems of the form (1.6) which are of an order greater than two.

1.4 Runge–Kutta Methods

The order of an MoC scheme goes hand-in-hand with the order of the ODE solver used to integrate along the characteristics. This follows from the derivation of Method (1.17). Some of the most popular and widely used explicit methods for integrating ODEs are Runge–Kutta (RK) methods, and therefore we will endeavor to implement them within the MoC. We now describe the general framework of the explicit RK

methods.

To illustrate how one implements an RK method for an ODE, suppose $y'(t) = f(y,t)$ is an ODE that we wish to integrate. Suppose one has Y_n, the numerical solution at level n. Then, given a stepsize h, one computes Y_{n+1} as follows:

$$k_1 = f(Y_n, t_n + b_1 h); \tag{1.27a}$$

$$k_2 = f(Y_n + a_{21} h k_1, t_n + b_2 h); \tag{1.27b}$$

$$k_3 = f(Y_n + a_{31} h k_1 + a_{32} k_2, t_n + b_3 h); \tag{1.27c}$$

$$\vdots$$

$$k_m = f\left(Y_n + h \sum_{i=1}^{m-1} a_{mi} k_i, t_n + b_m h\right); \tag{1.27d}$$

$$Y_{n+1} = Y_n + h \sum_{i=1}^{m} c_i k_i. \tag{1.27e}$$

Each k_i is referred to as the i^{th} *stage derivative*. Here the constants a_{ij}, b_i, c_i, and m define the particular RK method. To ensure at least $O(h)$ accuracy, one must have $b_1 = 0$ and $\sum c_i = 1$. It is common to present RK methods as a *Butcher tableau* [4], named for the New Zealand mathematician John C. Butcher. For example, Method (1.27) has the following Butcher tableau:

$$\begin{array}{c|cccc} b_1 & & & & \\ b_2 & a_{21} & & & \\ b_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \vdots & \ddots & \\ b_m & a_{m1} & a_{m2} & \cdots & a_{m(m-1)} \\ \hline & c_1 & c_2 & \cdots & c_m \end{array} \tag{1.28}$$

The only first-order RK method is the Simple Euler (SE) method:

$$k_1 = f(Y_n, t_n), \tag{1.29a}$$

$$Y_{n+1} = Y_n + hk_1. \tag{1.29b}$$

In tableau form, method (1.29) is:

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

Second-order RK methods include, among others, the Modified Euler (explicit trapezoid, ME) and Midpoint (MP) methods. Written in the form of (1.27), the ME method is:

$$k_1 = f(Y_n, t_n); \tag{1.30a}$$

$$k_2 = f(Y_n + hk_1, t_{n+1}); \tag{1.30b}$$

$$Y_{n+1} = Y_n + \frac{h}{2}(k_1 + k_2), \tag{1.30c}$$

where we use the notation $t_{n+\alpha}$ to mean $t_n + \alpha h$. In tableau form, the ME method is therefore:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{1.31}$$

Similarly, the MP method is:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \tag{1.32}$$

MoC schemes have already been developed based on first- and second-order RK methods (namely, SE and ME). These methods will be given in a later section. We refer to the resulting methods as MoC-SE and MoC-ME. Thus, the main focus of this thesis will be on the development of MoC algorithms based on third- and fourth-order explicit RK methods. From now on, we will **omit the word 'explicit'** since we **will not consider** implicit RK methods.

1.5 Stability Analysis of ODEs

When solving a system of PDEs (or ODEs) numerically, it is important to consider if the method being used is *stable*. Slightly deviating from the definition commonly used in the numerical analysis literature, we will refer to a method as *stable* if, when solving a system of PDEs (or ODEs) with a bounded solution, the numerical solution is also bounded. If the numerical solution is unbounded, then the method is called *unstable*. The procedure that one undertakes when determining if a method is stable or unstable is called the *stability analysis*.

We first show how one performs the stability analysis of methods used to solve ODEs, as this technique and its results will be useful to us later on. Then, we will show how one performs the stability analysis of methods used to solve PDEs, as this can predict and explain the numerical results of the MoC schemes that we develop in

this thesis.

We now present a more rigorous discussion of stability. Suppose we solve the IVP

$$y' = f(y,t), \quad y(t_0) = y_0, \tag{1.33}$$

with some numerical method, denoting the *ideal* solution, computed **without** round-off error, as Y_n, and the *actual* solution, computed **with** round-off error, as U_n. Denote the round-off error which arises in the computation of U_n as ϵ_n. We say that the method is stable if

$$||U - Y||_\infty \leq C||\epsilon||_\infty, \tag{1.34}$$

where C is a constant independent of the stepsize h. When analyzing the stability of a numerical method, one applies the method to the so-called *model equation*,

$$y' = \lambda y, \quad \lambda \in \mathbb{C}. \tag{1.35}$$

Let us explain why this is so. According to (1.34), we define stability in terms of the deviation between numerical solutions Y and U. Y and U are supposed to match their corresponding analytical solutions $y(t)$ and $u(t)$. The difference between y and u satisfies:

$$(y-u)' = f(y,t) - f(u,t) \approx f_y(y,t)(y-u). \tag{1.36}$$

Near any given point (y,t), $f_y(y,t)$ can be approximated by a constant, and the equation for $(y-u)$ takes the form of (1.35). Thus, the model problem (1.35) is the

linearized form of the original problem (1.33). We perform the stability analysis on (1.35) because it is usually simpler than (1.33). Depending on the value of λ, which is the range of values f_y takes on in (1.36), a method applied to (1.35) may be stable or unstable.

1.5.1 Stability Analysis of RK Methods: Simple Euler

To show how one can do this for an RK method, we present the stability analysis of the SE and ME methods ((1.29) and (1.30), respectively). Applying (1.35) to (1.29), we have:

$$k_1 = \lambda Y_n, \tag{1.37a}$$

$$Y_{n+1} = Y_n + hk_1. \tag{1.37b}$$

Substituting (1.37a) into (1.37b) yields

$$Y_{n+1} = Y_n + h\lambda Y_n, \tag{1.38}$$

and therefore

$$Y_{n+1} - (1 + h\lambda)Y_n = 0. \tag{1.39}$$

We now present the general procedure that we will use in this thesis to analyze the stability of a numerical method for ODEs. One needs to solve the simple equation (1.39). To do this, we follow the procedure used to solve a linear ODE. For example,

to solve the ODE

$$y'' + a_1 y' + a_0 y = 0, \quad a_{0,1} = \text{const.},$$

one makes the substitution $y = e^{\rho t}$, yielding the polynomial equation

$$\rho^2 + a_1 \rho + a_0 = 0.$$

In a similar fashion, we substitute

$$Y_n = \rho^n Y_0 \tag{1.40}$$

into (1.39) to obtain:

$$\rho - (1 + h\lambda) = 0. \tag{1.41}$$

The l.h.s. of (1.41) is called the *stability polynomial* for the SE method and has the solution $\rho = 1 + h\lambda$. Therefore, from (1.40),

$$Y_n = Y_0(1 + h\lambda)^n. \tag{1.42}$$

For $\text{Re}(\lambda) \leq 0$, the actual solution of the model problem, $y_0 e^{\lambda t}$, does not increase. According to (1.42), this will be the case only if $|1 + h\lambda| \leq 1$. Thus, the SE method is stable when

$$|1 + h\lambda| \leq 1. \tag{1.43}$$

Later in this thesis, we will encounter many cases where the stability polynomial is *not* linear. However, the stability analysis will be the same. If one obtains a k^{th}-degree stability polynomial, one will have the k roots $\rho_1, \ldots, \rho_k$, and the method will be stable for $|\rho_1| \leq 1, \ldots, |\rho_k| \leq 1$.

1.5.2 Stability Analysis of RK Methods: Modified Euler

Now let us follow the procedure given in the previous subsection to analyze the stability of the ME method, (1.30). Applying the model equation (1.35) to (1.30) gives us:

$$k_1 = \lambda Y_n, \tag{1.44a}$$

$$k_2 = \lambda(Y_n + hk_1), \tag{1.44b}$$

$$Y_{n+1} = Y_n + \frac{h}{2}(k_1 + k_2). \tag{1.44c}$$

Making the appropriate substitutions and simplifying yields:

$$Y_{n+1} - \left(1 + h\lambda + \frac{(h\lambda)^2}{2}\right) Y_n = 0. \tag{1.45}$$

We make the substitution (1.40) to obtain the stability polynomial

$$\rho - \left(1 + h\lambda + \frac{(h\lambda)^2}{2}\right) = 0. \tag{1.46}$$

Therefore, the ME method is stable for

$$\left|1 + h\lambda + \frac{(h\lambda)^2}{2}\right| \leq 1. \tag{1.47}$$

The stability criterion for the MP method (1.32) is the same as for the ME method. In fact, all second-order explicit RK methods share the stability polynomial (1.46). The same can be said of the stability polynomial obtained for third- and fourth-order explicit RK methods [4].

1.5.3 Stability Analysis of RK Methods: Stability Regions

Recall that in (1.35), λ is a complex number. Thus, the stability criterion for a numerical method corresponds to a region in the $\mathrm{Re}(h\lambda)$-$\mathrm{Im}(h\lambda)$ plane. It is easy to see that the stability criterion for the SE method, (1.43), corresponds to the closed unit disk centered at $\mathrm{Re}(h\lambda) = -1$. To see the stability region for the ME method, one can plot (1.47) to obtain an oval-shaped region with the same center. Both of these regions are shown in Fig. 1.4.

Examining Fig. 1.4, one can observe that the stability region of the ME method is "flatter" along the $\mathrm{Im}(h\lambda)$ axis than that of the SE method. This observation will be quite useful to us for the following reason: systems which we apply the MoC to admit conserved quantities, and the equations governing these quantities have eigenvalues that are purely imaginary. Therefore, we can expect to better numerically conserve these quantities when the boundary of the stability region for the method being used follows the imaginary axis as closely as possible.

We illustrate the preceding discussion with an example. Consider the equations of a simple harmonic oscillator,

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1. \tag{1.48}$$

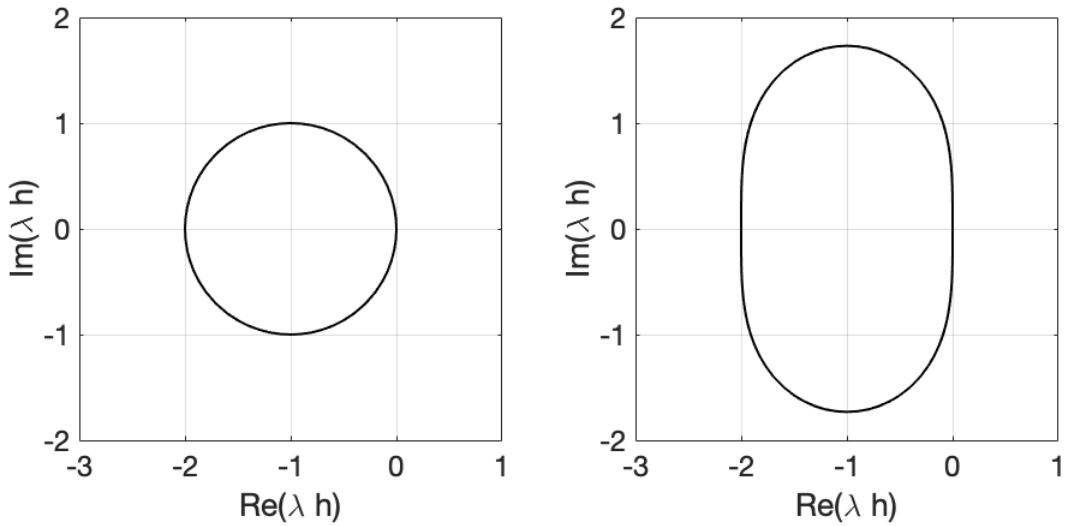

Figure 1.4: Stability regions of the SE (left) and ME (right) methods. The boundary of each region corresponds to $|\rho| = 1$, and the interior of each region corresponds to $|\rho_1| < 1$. This means that for $h\lambda$ on the boundary, the numerical solution will neither grow nor decay, and for $h\lambda$ in the interior, the numerical solution will decay. For $h\lambda$ outside of each region, the numerical solution grows.

It is a well-known fact that the harmonic oscillator admits a conserved quantity called the *Hamiltonian*, given by

$$H(v, y) = \frac{1}{2}v^2 + U(y), \quad U(y) = -\int f(y)\, dy,$$

where $v = y'$ and $f(y) = y'$. Now, let us solve (1.48) numerically using SE and ME. The stability region of SE is tangent to the imaginary axis with the minimal degree of tangency; that is, $\mathrm{Re}(\lambda h) = O\left((\mathrm{Im}(\lambda h))^2\right)$ near the origin. Therefore, we should expect this method to conserve the Hamiltonian quite poorly. The ME stability region, however, has a higher degree of tangency to the imaginary axis: $\mathrm{Re}(\lambda h) = O\left((\mathrm{Im}(\lambda h)^4\right)$ near the origin; so we should expect this method to conserve the Hamiltonian much better than the SE method. Numerical results verifying this

hypothesis are shown in Fig. 1.5.

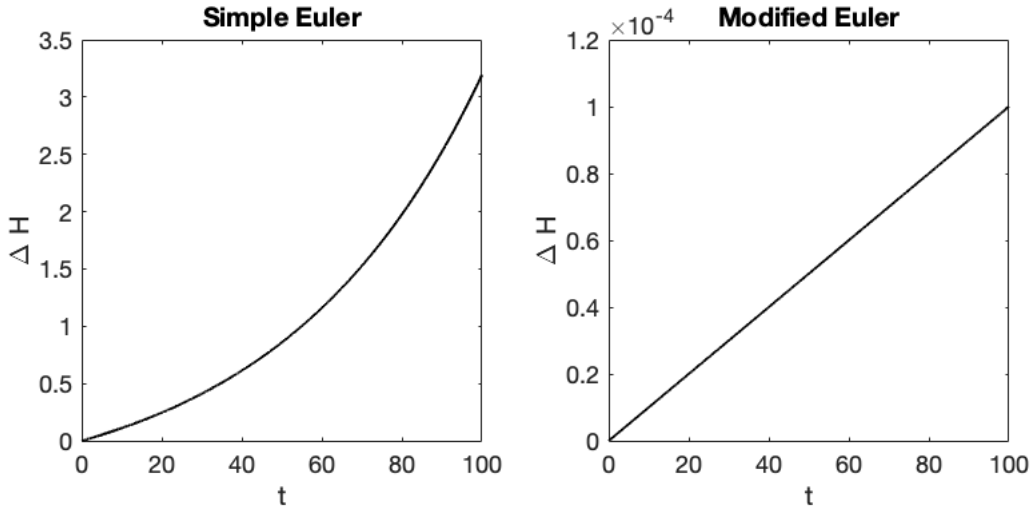

Figure 1.5: Numerical change in the Hamiltonian as a result of integrating (1.48) to $t = 100$ with SE and ME, using $h = 0.02$. Notice the disparate vertical scales in the two figures. As predicted, the ME does a much better job in conserving the Hamiltonian.

Thus, the take-home message of this subsection is the following: When using RK methods in the MoC, we will seek methods which preserve some of the conserved quantities, and the ability of a method to do this can be predicted by examining how closely the boundary of its stability region follows the imaginary axis.

1.6 Prior work

Now with the knowledge of hyperbolic PDEs, the MoC, and RK methods, we will discuss prior work related to the application of RK methods to the MoC, and will introduce the systems of hyperbolic PDEs that will be the focus of the analysis performed in this thesis.

In a recent work [1], the MoC-SE and MoC-ME were applied to a system of hyperbolic PDEs. The system considered is one that arises in the study of the propagation

of light in birefringent optical fibers with Kerr nonlinearity. In its linearized form, the system has many physical applications, including acousto-optical interactions, interaction of crossed beams in photorefractive materials, interaction of counter- and co-propagating light waves in a medium with a periodic refractive index, and in the relativistic field theory. We present the system now, as much of our analysis later on is based on the same system. The system considered was

$$\underline{\mathbf{S}}_t^{\pm} + \underline{\mathbf{S}}_x^{\pm} = \underline{\mathbf{S}}^{\pm} \times \hat{\mathbf{J}}\underline{\mathbf{S}}^{\mp}, \tag{1.49}$$

where $\underline{\mathbf{S}}^{\pm} \equiv [S_1^{\pm}, S_2^{\pm}, S_3^{\pm}]^T$, $\hat{\mathbf{J}} = \mathrm{diag}(1, -1, 2)$, and " $\times$ " is the cross-product. In component form, system (1.49) is:

$$(\partial_t + \partial_x)S_1^+ = S_3^+ S_2^- - 2S_2^+ S_3^-, \tag{1.50a}$$

$$(\partial_t + \partial_x)S_2^+ = 2S_1^+ S_3^- + S_3^+ S_1^-, \tag{1.50b}$$

$$(\partial_t + \partial_x)S_3^+ = -(S_1^+ S_2^- + S_2^+ S_1^-), \tag{1.50c}$$

$$(\partial_t - \partial_x)S_1^- = S_3^- S_2^+ - 2S_2^- S_3^+, \tag{1.50d}$$

$$(\partial_t - \partial_x)S_2^- = 2S_1^- S_3^+ + S_3^- S_1^+, \tag{1.50e}$$

$$(\partial_t - \partial_x)S_3^- = (\partial_t + \partial_x)S_3^+. \tag{1.50f}$$

A system similar to (1.50) which governs the propagation of two intense counter-propagating beams in an isotropic optical fiber was studied in [5]. In the numerical experiments conducted in that paper, the nondimensional time $t = 1$ corresponds to approximately 10 nanoseconds, in which time the pulsed beams propagate the length of a fiber whose approximate length is 2 meters. In this thesis, we will present many

experimental results which integrate system (1.50) up to the nondimensional time $t = 500$. Following the preceding discussion, this corresponds to integrating (1.50) along a 1-kilometer fiber.

Another system that the MoC-ME has been applied to [6] is the massive Gross–Neveu model in the relativistic field theory. This system can be written as:

$$u_t + u_x = i(|v|^2 u + v^2 u^*) - iv, \tag{1.51a}$$

$$v_t - v_x = i(|u|^2 v + u^2 v^*) - iu, \tag{1.51b}$$

where " $*$ " is the complex conjugate. System (1.51) will be featured in our analysis as well. Evidently, system (1.51) is in the form (1.6), and (1.50) also has that form, where $u_{1,2}, f_{1,2}$ are 3-component vectors. This fact will be useful to us later on.

System (1.51) admits two conserved quantities: the Hamiltonian,

$$H = \int \left[-i(u^* u_x - v^* v_x) - \frac{1}{2}(uv^* + u^* v)^2 + (uv^* + u^* v) \right] \, dx, \tag{1.52}$$

and charge,

$$Q = \int \left(|u|^2 + |v|^2 \right) \, dx. \tag{1.53}$$

In later chapters, we will examine the effect that the MoC has on H and Q.

2 Second-order MoC schemes

The purpose of this section is to describe the formulation of first- and second-order MoC schemes, and to provide the details of the von Neumann stability analysis.

2.1 Existing first- and second-order MoC schemes

As mentioned in Sec. 1.4, MoC schemes have already been developed based on first- and second-order RK methods. The focus of this thesis will be on the development of MoC schemes for hyperbolic equations with two **straight-line and crossing** characteristics. We present now how one can do this using the ME method.

Suppose we have a system of two hyperbolic PDEs, such as (1.6),

$$w_{1t} + cw_{1x} = f_1(w_1, w_2), \tag{1.6a}$$

$$w_{2t} - cw_{2x} = f_2(w_1, w_2), \tag{1.6b}$$

with characteristics (1.5):

$$\xi_1 = x - ct \quad \text{and} \quad \xi_2 = x + ct. \tag{1.5}$$

We will work with a rescaled t such that $\mathbf{c = 1}$ **in what follows**. Following the

derivation of (1.16), one can write (1.6) as

$$w_{1t} = f_1(w_1, w_2) \quad \text{on} \quad \xi_1 = \text{const}, \tag{2.1a}$$

$$w_{2t} = f_2(w_1, w_2) \quad \text{on} \quad \xi_2 = \text{const}. \tag{2.1b}$$

The numerical solution for the schemes that we will present to solve (2.1) is computed on the stencil shown in Fig. 2.1. Since on this grid, $\Delta x = c\Delta t$ and $c \equiv 1$, we denote

$$\Delta x = \Delta t \equiv h.$$

We first illustrate how to solve system (2.1) with accuracy $O(h)$ (the SE method,

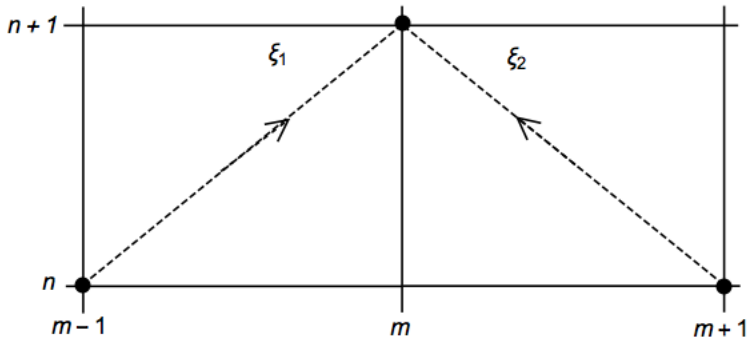

Figure 2.1: Stencil for the numerical solution of (2.1)

(1.29)), and then with accuracy $O(h^2)$ (the ME method, (1.30)). The purpose of illustrating these simple cases is to explain how the solution of the PDE system (2.1) differs from a similar ODE system where w_1, w_2 would have been evolving along **the same** characteristic.

The SE method for the ODE $y'(t) = f(y,t)$ is given by:

$$k_1 = f(Y_n, t_n), \tag{1.29a}$$

$$Y_{n+1} = Y_n + hk_1. \tag{1.29b}$$

When applying scheme (1.29) to the MoC to solve (2.1), we compute the change in W_1 and W_2 along each characteristic ξ_1, ξ_2 **separately**. When we compute along ξ_1, we treat W_1 as Y. The **key point** is that we also treat W_2 **as a function of time**, which we approximate from (2.1b).

To demonstrate this, suppose we want to compute W_1 at the node $(x = m, t = n+1)$ in Fig. 2.1 (we denote this as $(W_1)_m^{n+1}$). As stated in the previous paragraph, **to compute the change in W_1, we compute along ξ_1**. From Fig. 2.1, we see that ξ_1 **is propagating from the node** $(m-1, n)$. Thus, when computing k_1, **all arguments of f_1 must be evaluated at this same node,** $(m-1, n)$. Let k_{11} denote the k_1 which will be used in the computation of W_1. In general, in what follows, the notation k_{ij} will stand for the "stage derivative" k_j in the RK method computed for the variable W_i. By (1.29a), we will have an equation of the form

$$k_{11} = f_1\left((W_1)_{m-1}^n, t_n\right). \tag{2.2}$$

Now, recall the **key point** that along ξ_1 we treat W_2 as a function of time, and that **all arguments of f_1 must be evaluated at this same node**. Thus, (2.2) should be computed by:

$$k_{11} = f_1\left((W_1)_{m-1}^n, (W_2)_{m-1}^n\right). \tag{2.3a}$$

In a similar fashion, one computes k_{21} (the k_1 which will be used in the computation of W_2) by:

$$k_{21} = f_{\text{r.h.s. of (2.1b)}}\left((W_2)^n_{m+1}, t_n\right) \equiv f_2\left((W_1)^n_{m+1}, (W_2)^n_{m+1}\right). \tag{2.3b}$$

Notice that in k_{21}, all arguments of f_2 are evaluated at the node from which ξ_2 propagates, and that W_1 is treated as a function of time.

It is then straightforward to compute $(W_{1,2})^{n+1}_m$ by (1.29b); one simply treats the Y_n term as the node from which the characteristic $\xi_{1,2}$ is propagating, and uses the already computed $k_{\{1,2\}1}$.

$$(W_1)^{n+1}_m = (W_1)^n_{m-1} + hk_{11}, \tag{2.3c}$$

$$(W_2)^{n+1}_m = (W_2)^n_{m+1} + hk_{21}. \tag{2.3d}$$

Equations (2.3) form the MoC-SE. Analogous to the ODE, this method has global accuracy $O(h)$ and hence local accuracy $O(h^2)$.

We now present the MoC-ME. Recall the ME method (1.30) for the ODE $y'(t) = f(t, y)$:

$$k_1 = f(Y_n, t_n), \tag{1.30a}$$

$$k_2 = f(Y_n + hk_1, t_{n+1}), \tag{1.30b}$$

$$Y_{n+1} = Y_n + \frac{h}{2}(k_1 + k_2). \tag{1.30c}$$

The MoC-ME will be computed on the same stencil as the MoC-SE. First, the computation of $k_{\{1,2\}1}$ is performed exactly how it was done in the MoC-SE.

Next, one must compute $k_{\{1,2\}2}$ by (1.30b). We demonstrate how this is done for k_{12}. Since ξ_1 propagates from $(x = m-1, t = n)$, and W_2 is treated as a function of time along the characteristic $\xi_1 = \text{const}$, the term t_{n+1} corresponds to W_2 at the **next** time level. Along the characteristic, this is $(W_2)_m^{n+1}$. Therefore, we **need** the point $(W_2)_m^{n+1}$, but **we do not yet have it**.

The resolution is simple. The MoC-ME needs to have local accuracy $O(h^3)$. However, each of $k_{1,2}$ in (1.30) only needs to be computed with local accuracy $O(h^2)$, since according to (1.30c), they will each be multiplied by h, yielding local accuracy $O(h^3)$. Therefore, we only need to compute k_2 with accuracy $O(h^2)$, which in turn means that we need an $O(h^2)$-accurate $(W_2)_m^{n+1}$. Fortunately, **we know how to compute this $O(h^2)$-accurate solution by MoC-SE** (2.3d).

Let $\overline{W}$ denote the MoC-SE solution. Then, by (1.30b), we have:

$$k_{12} = f_1\left((W_1)_{m-1}^n + hk_{11}, (\overline{W}_2)_m^{n+1}\right). \tag{2.4a}$$

Similarly, when computing k_{22}, one only needs the SE approximation of W_1. Then, one can compute:

$$k_{22} = f_2\left((\overline{W}_1)_m^{n+1}, (W_2)_{m+1}^n + hk_{21}\right). \tag{2.4b}$$

Finally, the computation of (1.30c) is straightforward:

$$(W_1)_m^{n+1} = (W_1)_{m-1}^n + \frac{h}{2}(k_{11} + k_{12}), \tag{2.4c}$$

$$(W_2)_m^{n+1} = (W_2)_{m+1}^n + \frac{h}{2}(k_{21} + k_{22}). \tag{2.4d}$$

Equations (2.4), along with (2.3{a,b}) for $k_{\{1,2\}1}$ and (2.3{c,d}) for $\overline{W}_{1,2}$, form the MoC-ME.

Method (2.3) is the MoC-ME, an existing second-order method which has been the focus of prior research. In this thesis, before considering higher-order methods, we develop another second-order accurate MoC scheme.

2.2 The MoC Scheme using the Midpoint Method

The ME method is a well-known second-order RK method used for the numerical integration of ODEs. As mentioned in Sec. 1.4, another such method is the Midpoint (MP) method, whose Butcher tableau is:

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

For the ODE $y'(t) = f(y,t)$, the MP method can be written as:

$$k_1 = f(Y_n, t_n), \tag{2.5a}$$

$$k_2 = f\left(Y_n + \frac{h}{2}k_1, t_{n+\frac{1}{2}}\right), \tag{2.5b}$$

$$Y_{n+1} = Y_n + hk_2. \tag{2.5c}$$

We proceed in developing an MoC scheme based off method (2.5) for two reasons. The first reason is to provide another method with local accuracy $O(h^3)$ that can be

used in higher-order MoC schemes. Recall that for the $O(h^3)$ local accurate MoC-ME, we needed an $O(h^2)$ solver. As one might expect, for methods that we present later on that have local accuracy $O(h^4)$, we will need a solver of local accuracy $O(h^3)$.

The second reason is to present a feature of many MoC schemes which involves the computation of a point that is **not** located at a grid point on the stencil. The importance of this feature will be explained later on.

Before presenting the MoC-MP, let us make the following remark about notations: For the remainder of this thesis, when describing the general formulation of an MoC scheme, we will use the following conventions: the solution which propagates along characteristic $\xi_1(\equiv \xi^+)$ will be referred to as Y^+, and that which propagates along $\xi_2(\equiv \xi^-)$ will be called Y^-. (The superscripts reflect the direction in which the solution propagates.) The corresponding f_1 and f_2 will be f^+ and f^-. Finally, the corresponding i^{th} stage derivatives will be denoted k_i^+ and k_i^-.

The MoC-MP will be computed on the stencil shown in Fig. 2.2.

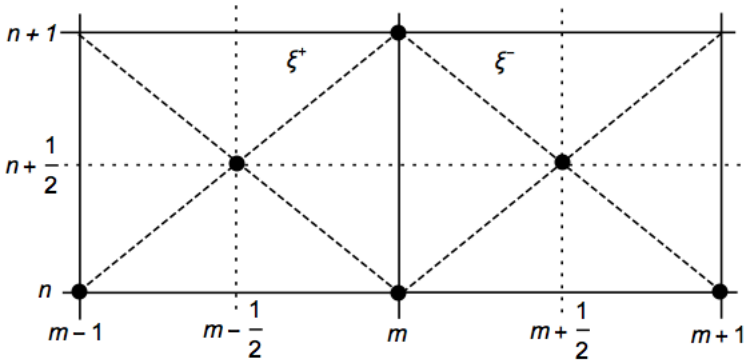

*Figure 2.2: Stencil for the MoC-MP. Solid lines represent actual grid lines, and dashed lines represent the characteristics. Dotted lines are **not** on the actual grid; we include them to reference a point which we compute that does not lie on the grid.*

We now proceed with our formulation of the MoC-MP. We will explain how one

computes the point $(Y^+)_m^{n+1}$, and the explanation for how one computes $(Y^-)_m^{n+1}$ follows the same procedure.

To compute $(Y^+)_m^{n+1}$, we only need to look at the left-most panel of Fig. 2.2. The first stage derivative, k_1^+, is computed the same way as in the MoC-SE and MoC-ME. We state the formulation here to exhibit the new notations.

$$k_1^+ = f^+\left((Y^+)_{m-1}^n, (Y^-)_{m-1}^n\right). \tag{2.6a}$$

As we have established, when computing along ξ^+, Y^- is treated as a function of time. Therefore, when computing k_2^+, we need the point $(Y^-)_{m-1/2}^{n+1/2}$ (see 2.5b). We compute this point as follows:

$$(Y^-)_{m-1/2}^{n+1/2} = (Y^-)_m^n + \frac{h}{2} f^-\left((Y^+)_m^n, (Y^-)_m^n\right). \tag{2.6b}$$

This is the SE method with stepsize $h/2$, and thus has the required local accuracy $O(h^2)$. This is the **first instance** where we have computed **a point that is not** a grid point. This gives us all of the required ingredients to compute k_2^+:

$$k_2^+ = f^+\left((Y^+)_{m-1}^n + \frac{h}{2}k_1^+, (Y^-)_{m-1/2}^{n+1/2}\right). \tag{2.6c}$$

Finally, by (2.5c), we have:

$$(Y^+)_m^{n+1} = (Y^+)_{m-1}^n + hk_2^+. \tag{2.6d}$$

When computing $(Y^-)_m^{n+1}$, one only needs to examine the right-most panel of Fig.

2.2. Following similar logic as above, one obtains:

$$k_1^- = f^-\left((Y^+)_{m+1}^n, (Y^-)_{m+1}^n\right), \tag{2.6e}$$

$$(Y^+)_{m+1/2}^{n+1/2} = (Y^+)_m^n + \frac{h}{2} f^+\left((Y^+)_m^n, (Y^-)_m^n\right), \tag{2.6f}$$

$$k_2^- = f^-\left((Y^+)_{m+1/2}^{n+1/2}, (Y^-)_{m+1}^n + \frac{h}{2}k_1^-\right), \tag{2.6g}$$

$$(Y^-)_m^{n+1} = (Y^-)_{m+1}^n + hk_2^-. \tag{2.6h}$$

Together, equations (2.6) form the MoC-MP.

2.3 Stability analysis

In Sec. 1.5, we performed the stability analysis of numerical methods for ODEs. In this section, we describe and perform the stability analysis of the MoC for a system of hyperbolic PDEs. The system for which we will present the details of our analysis is System (1.50), found in Sec. 1.6.

2.3.1 Linearization of System (1.50)

Before we begin the stability analysis, it will be useful for us to linearize system (1.50). Indeed, we explain in the next subsection that linearization is the first step of the subsequent stability analysis.

Linearization is always performed on the background of some known solution. It can be shown that the system (1.50) has the constant solution

$$S_{1,3}^{\pm} = 0, \quad S_2^{\pm} = \pm 1. \tag{2.7}$$

To linearize (1.50), we set

$$S_j^{\pm} = S_{j0}^{\pm} + s_j^{\pm}, \quad j = 1, 2, 3, \tag{2.8}$$

where $S_{j0}^{\pm}$ are the components of the solution (2.7) and $s_j^{\pm}$ are small ($\ll 1$) perturbations. Substituting (2.8) into (1.50a) yields:

$$\begin{aligned}(\partial_t + \partial_x)s_1^+ &= s_3^+(s_2^- - 1) - 2s_3^-(s_2^+ + 1) \\ &= -s_3^+ - 2s_3^- + O(s_3^+ s_2^-) + O(s_3^- s_2^+).\end{aligned} \tag{2.9}$$

Since the $O(s_3^{\pm} s_2^{\mp})$ terms are much smaller than the other two terms, they can be ignored. Performing a similar analysis for all of the equations in (1.50) gives us:

$$(\partial_t + \partial_x)s_1^+ = -s_3^+ - 2s_3^-, \tag{2.10a}$$

$$(\partial_t + \partial_x)s_2^+ = 0, \tag{2.10b}$$

$$(\partial_t + \partial_x)s_3^+ = s_1^+ - s_1^-, \tag{2.10c}$$

$$(\partial_t - \partial_x)s_1^- = s_3^- + 2s_3^+, \tag{2.10d}$$

$$(\partial_t - \partial_x)s_2^- = 0, \tag{2.10e}$$

$$(\partial_t - \partial_x)s_3^- = s_1^+ - s_1^-. \tag{2.10f}$$

Written in matrix form, system (2.10) is:

$$\mathbf{s}_t + \mathbf{\Sigma}\mathbf{s}_x = \mathbf{P}\mathbf{s}, \tag{2.11a}$$

$$(s_2^{\pm})_t \pm (s_2^{\pm})_x = 0, \tag{2.11b}$$

where in (2.11a), $\mathbf{s} = (s_1^+, s_3^+, s_1^-, s_3^-)^T$ and

$$\mathbf{\Sigma} = \text{diag}(I, -I), \quad \mathbf{P} = \begin{pmatrix} P^{++} & P^{+-} \\ P^{-+} & P^{--} \end{pmatrix}, \tag{2.12}$$

and in (2.12)

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad P^{++} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad P^{+-} = -\begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix},$$
$$P^{--} = -P^{++}, \; P^{-+} = -P^{+-}. \tag{2.13}$$

2.3.2 General idea behind the von Neumann stability analysis

The procedure we will follow to analyze the stability of the MoC schemes in this thesis is the von Neumann stability analysis. The von Neumann analysis is applied to systems of the form

$$\mathbf{y}' + \mathbf{A}\mathbf{y}_x = \mathbf{f}(\mathbf{y}), \tag{2.14}$$

where $\mathbf{y}' \equiv \mathbf{y}_t$ and $\mathbf{f}$ is a nonlinear vector function. As in the stability analysis of ODEs, our goal is to determine how the error behaves as one integrates in time. Thus, the first step is to linearize $\mathbf{f}$, as we did for the model problem $y' = \lambda y$. The result of this linearization is:

$$\tilde{\mathbf{y}}' + \mathbf{A}\tilde{\mathbf{y}}_x = \mathbf{B}\tilde{\mathbf{y}}, \tag{2.15}$$

where $\mathbf{B}$ is the Jacobian matrix corresponding to $\mathbf{f}$, and $\tilde{\mathbf{y}} \equiv \mathbf{y} - \mathbf{y}_0$, where $\mathbf{y}_0$ is the known solution whose small perturbations, $\tilde{\mathbf{y}}$, we are analyzing. In what follows we omit the tilde, as the "full" $\mathbf{y}$ of Eq. (2.14) will not appear again.

To enable the von Neumann analysis, which is essentially based on the spatial discrete Fourier transform, we assume

$$\mathbf{y}(x_m, t_n) = e^{ikx_m}\rho^n\mathbf{u} = e^{ikmh}\rho^n\mathbf{u}, \tag{2.16}$$

with $k \in [-k_{\max}, k_{\max})$ and $k_{\max} = \pi/h$ and $\mathbf{u}$ an eigenvector of a matrix which will appear later on. Additionally, we must impose periodic boundary conditions, meaning

$$(\mathbf{y})_{-1}^n \equiv (\mathbf{y})_M^n, \quad (\mathbf{y})_{M+1}^n \equiv (\mathbf{y})_0^n,$$

where $M+1$ is the number of spatial grid points and $m = 0, 1, \ldots, M$. Substituting (2.16) into (2.15), one obtains an equation for ρ^n of the form

$$\rho^{n+1} = \mathbf{\Phi}(z)\rho^n, \quad z = kh, \tag{2.17}$$

where $z \in [-\pi, \pi)$, as follows from the range for k stated after (2.16). Examples of $\mathbf{\Phi}(z)$ will be given in the next subsection and also in later chapters. Following the analysis in Sec. 1.5, one solves (2.17) for ρ, obtaining a solution which depends on z that tells us how the error behaves in the method. For the method to be stable, we will require, similarly to (1.43) and (1.47), that $|\rho| \leq 1$ for all $z \in [-\pi, \pi)$. Through

(2.16), we can write (2.17) equivalently as:

$$(\mathbf{y})_m^{n+1} = \mathbf{\Phi}(z)(\mathbf{y})_m^n. \tag{2.18}$$

Equations of the form (2.18) are those which we seek in the von Neumann analysis.

2.3.3 Simple example of the von Neumann analysis (MoC-SE)

We will first illustrate how one applies the von Neumann analysis to the MoC-SE, (2.3), when used to solve (1.50). In what follows, we will use these notations: boldfaced letters with no underline refer to the 4×1 vector of all components (e.g. $\mathbf{k_1} = (k_{11}^+, k_{13}^+, k_{11}^-, k_{13}^-)^T$). Boldfaced letters with an underline and superscript " $+$ " or " $-$ " refer to the 2×1 vector of components corresponding to the superscript (e.g. $\underline{\mathbf{k}_\mathbf{1}}^+ = (k_{11}^+, k_{13}^+)^T$). We also introduce the convention that stage derivatives will be indexed at the node *from* which they propagate. Note that when analyzing the stability of (1.50), we ignore the s_2 components, since the r.h.s. of (2.10b) and (2.10e) is zero and will therefore have no effect on the stability of any method.

Applying (2.3a) and (2.3b) to the linearized system (2.11) yields:

$$\begin{pmatrix} \left(\underline{\mathbf{k}_\mathbf{1}}^+\right)_{m-1}^n \\ \left(\underline{\mathbf{k}_\mathbf{1}}^-\right)_{m+1}^n \end{pmatrix} = \begin{pmatrix} P^{++} & P^{+-} \\ \mathcal{O} & \mathcal{O} \end{pmatrix} (\mathbf{s})_{m-1}^n + \begin{pmatrix} \mathcal{O} & \mathcal{O} \\ P^{-+} & P^{--} \end{pmatrix} (\mathbf{s})_{m+1}^n, \tag{2.19}$$

where $\mathcal{O}$ is the 2×2 zero matrix. We now substitute (2.16) into (2.19) to obtain:

$$\begin{aligned}
\begin{pmatrix} \left(\underline{\mathbf{k}}_{\mathbf{1}}{}^{+}\right)_{m-1}^{n} \\ \left(\underline{\mathbf{k}}_{\mathbf{1}}{}^{-}\right)_{m+1}^{n} \end{pmatrix} &= \begin{pmatrix} P^{++} & P^{+-} \\ \mathcal{O} & \mathcal{O} \end{pmatrix} \mathrm{diag}(e^{ik(m-1)hI})\rho^n + \begin{pmatrix} \mathcal{O} & \mathcal{O} \\ P^{-+} & P^{--} \end{pmatrix} \mathrm{diag}(e^{ik(m+1)hI})\rho^n \\
&= e^{ikmh}\rho^n \left[\begin{pmatrix} P^{++} & P^{+-} \\ \mathcal{O} & \mathcal{O} \end{pmatrix} \mathrm{diag}(e^{-ikhI}) + \begin{pmatrix} \mathcal{O} & \mathcal{O} \\ P^{-+} & P^{--} \end{pmatrix} \mathrm{diag}(e^{ikhI}) \right] \\
&= \left[\begin{pmatrix} P^{++} & P^{+-} \\ \mathcal{O} & \mathcal{O} \end{pmatrix} \mathrm{diag}(e^{-izI}) + \begin{pmatrix} \mathcal{O} & \mathcal{O} \\ P^{-+} & P^{--} \end{pmatrix} \mathrm{diag}(e^{izI}) \right] (\mathbf{s})_m^n \\
&= \mathbf{QP}\ (\mathbf{s})_m^n \equiv \mathcal{K}_1\ (\mathbf{s})_m^n, \qquad (2.20)
\end{aligned}$$

where

$$\mathbf{Q} = \exp[-iz\mathbf{\Sigma}]. \qquad (2.21)$$

We follow a similar process when applying (2.3c) and (2.3d) to (2.11) to obtain:

$$(\mathbf{s})_m^{n+1} = \mathbf{Q}(\mathbf{s})_m^n + h\mathcal{K}_1(\mathbf{s})_m^n = [\mathbf{Q}(\mathbf{I} + h\mathbf{P})]\,(\mathbf{s})_m^n, \qquad (2.22)$$

where in (2.20) $\mathbf{I}$ is the 4×4 identity matrix. Equation (2.22) is in the desired form (2.18), telling us that after a large enough number of iterations N, the size of the perturbations $(\mathbf{s})_m^N$ will be proportional to $|\lambda_{\max}|^N$, where $\lambda_{\max}$ is the largest in magnitude eigenvalue of

$$\mathbf{\Phi}(z) = \mathbf{Q}(\mathbf{I} + h\mathbf{P}). \qquad (2.23)$$

(See Appendix B for a more in-depth discussion of this.) We plot $\lambda_{\max}$ of this matrix for $z \in [0, \pi]$ in Fig. 2.3; the plot for $z \in [-\pi, 0]$ is symmetric relative to $z = 0$ and therefore is not plotted, here and everywhere below.

Equations (2.18), (2.21), and (2.23) illustrate the statement, made in Sec. 1.3, that MoC schemes preserve the linear dispersion relation (1.20) for system (2.11). Indeed, (2.18) implies that the frequency ω of the numerical solution is an eigenvalue of $-\ln(\mathbf{\Phi})/(ih)$. Then, (2.21) and (2.23) imply that for $k \gg 1$,

$$\omega = -\left(\ln \mathbf{Q} + O(h)\right)/(ih) = \pm k\left(1 + O(1/|k|)\right),$$

which agrees with (1.20).

In the von Neumann analysis of the MoC-ME (2.4), one obtains the stability matrix

$$\mathbf{\Phi}(z) = \frac{1}{2}\left[\mathbf{Q} + (\mathbf{I} + h\mathbf{P})\mathbf{Q}(\mathbf{I} + h\mathbf{P})\right]. \tag{2.24}$$

Some details of the derivation of (2.24) are found in [1]. Conceptually similar steps of a derivation are presented for the MoC-MP scheme in the next subsection. The largest eigenvalues for matrix (2.24) are plotted in Fig. 2.3. As explained in Appendix B, the largest eigenvalues imply that the instability growth rate is $O(h)$.

In Fig. 2.4, the results of this analysis are verified numerically. Appendix B discusses the numerical simulation, and how it agrees with the von Neumann analysis, in greater detail.

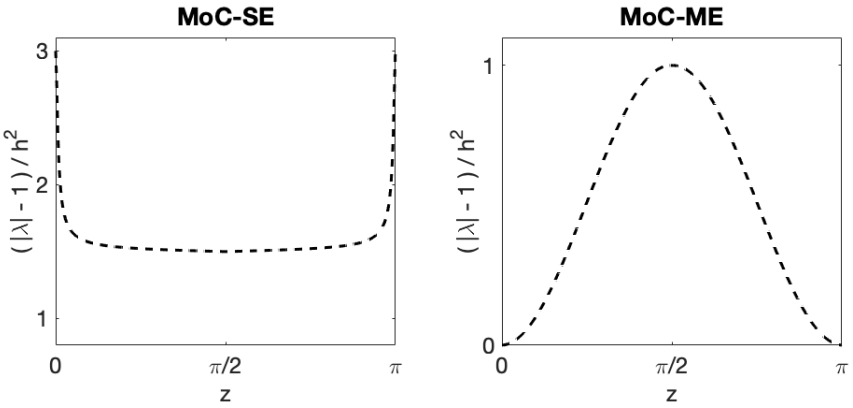

Figure 2.3: Maximum eigenvalue of (2.23) (left) and (2.24) (right) for $z \in [0, \pi]$ and $h = 0.01$. For the MoC-SE, the value on the y-axis is larger than 1 for all z, meaning that this method is unstable, with the largest instability occurring at $z = 0$ and $z = \pi$. For the MoC-ME, the "worst-case" eigenvalue occurs at $z = \pi/2$, with the value on the y-axis equal to 1. This means that a milder instability occurs for this method at modes in the middle of the Fourier spectrum. Both methods have a growth rate of $O(h^2)$ in the instability. Appendix B contains more quantitative details.

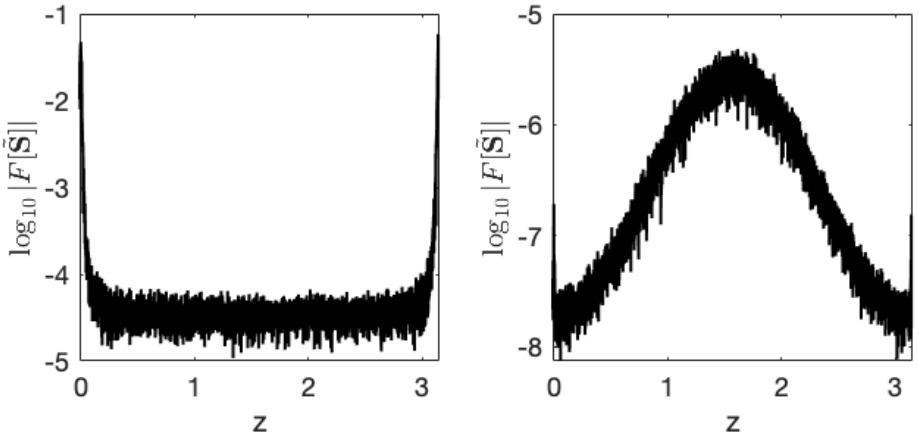

Figure 2.4: Logarithm of the Fourier spectrum of the error when solving (1.50) with the MoC-SE (left) and MoC-ME (right) at $t = 500$. In both cases, the stepsize used was $h = 0.01$. Both plots qualitatively agree with the results predicted by the von Neumann analysis, shown and described in Fig. 2.3; also see Appendix B. In Fig. 2.3, the worst-case eigenvalue for the MoC-SE is shown to be approximately 3 times larger than that for the MoC-ME. This agrees with the ratio of the error in the plots shown above; see Appendix B.

2.3.4 von Neumann stability analysis of the MoC-MP

We now perform the von Neumann stability analysis of the MoC-MP. Applying (2.6a) and (2.6e) to (2.11) and following the general procedure of the analysis yields, as it did in the MoC-SE,

$$\mathcal{K}_1 \equiv \begin{pmatrix} \left(\underline{\mathbf{k_1}}^{+}\right)_{m-1}^{n} \\ \left(\underline{\mathbf{k_1}}^{-}\right)_{m+1}^{n} \end{pmatrix} = \mathbf{QP}\ (\mathbf{s})_m^n. \tag{2.25}$$

Next, we apply (2.6b) and (2.6f) to (2.11) to obtain:

$$\begin{pmatrix} (\underline{\mathbf{s}}^{+})_{m+1/2}^{n+1/2} \\ (\underline{\mathbf{s}}^{-})_{m-1/2}^{n+1/2} \end{pmatrix} = (\mathbf{s})_m^n + \frac{h}{2}\mathbf{P}(\mathbf{s})_m^n = \left(\mathbf{I} + \frac{h}{2}\mathbf{P}\right)(\mathbf{s})_m^n \equiv \mathcal{S}_1(\mathbf{s})_m^n. \tag{2.26}$$

Because (2.26) is automatically written in terms of $(\mathbf{s})_m^n$, there is no need to make the substitution (2.16). To simplify the analysis when applying (2.6c) and (2.6g) to (2.11), we define

$$\left(\mathring{Y}^{+}\right)_m = (Y^{+})_{m-1}^{n} + \frac{h}{2}(k_1^{+})_{m-1}^{n}, \tag{2.27a}$$

$$\left(\mathring{Y}^{-}\right)_m = (Y^{-})_{m+1}^{n} + \frac{h}{2}(k_1^{-})_{m+1}^{n}. \tag{2.27b}$$

Then, (2.6c) and (2.6g) can be written as:

$$(k_2^{+})_{m-1}^{n} = f^{+}\left(\left(\mathring{Y}^{+}\right)_m, (Y^{-})_{m-1/2}^{n+1/2}\right), \tag{2.28a}$$

$$(k_2^{-})_{m+1}^{n} = f^{-}\left((Y^{+})_{m+1/2}^{n+1/2}, \left(\mathring{Y}^{-}\right)_m\right). \tag{2.28b}$$

Linearization of (2.27) yields, similarly to (2.22):

$$(\mathring{\mathbf{s}})_m = \left[\mathbf{Q}\left(\mathbf{I} + \frac{h}{2}\mathbf{P}\right)\right](\mathbf{s})_m^n \equiv \mathcal{S}_2(\mathbf{s})_m^n. \tag{2.29}$$

Now applying (2.28a) and (2.28b) to (2.11) and linearizing gives us:

$$\begin{aligned}\begin{pmatrix}(\underline{\mathbf{k}}_{\mathbf{2}}{}^{+})_{m-1}^n \\ (\underline{\mathbf{k}}_{\mathbf{2}}{}^{-})_{m+1}^n\end{pmatrix} &= \begin{pmatrix}P^{++} & \mathcal{O} \\ \mathcal{O} & P^{--}\end{pmatrix}(\mathring{\mathbf{s}})_m + \begin{pmatrix}\mathcal{O} & P^{+-} \\ P^{-+} & \mathcal{O}\end{pmatrix}\begin{pmatrix}(\underline{\mathbf{s}}^{+})_{m+1/2}^{n+1/2} \\ (\underline{\mathbf{s}}^{-})_{m-1/2}^{n+1/2}\end{pmatrix} \\ &= \left[\mathbf{P}_{\text{diag}}\mathcal{S}_2 + \mathbf{P}_{\text{offdiag}}\mathcal{S}_1\right](\mathbf{s})_m^n \equiv \mathcal{K}_2(\mathbf{s})_m^n.\end{aligned} \tag{2.30}$$

Finally, according to (2.6d) and (2.6h) we have:

$$(\mathbf{s})_m^{n+1} = \begin{pmatrix}(\underline{\mathbf{s}})_{m-1}^n \\ (\underline{\mathbf{s}})_{m+1}^n\end{pmatrix} + h\begin{pmatrix}(\underline{\mathbf{k}}_{\mathbf{2}}{}^{+})_{m-1}^n \\ (\underline{\mathbf{k}}_{\mathbf{2}}{}^{-})_{m+1}^n\end{pmatrix}. \tag{2.31}$$

Upon making the substitutions (2.16) and (2.30) and simplifying, we obtain:

$$(\mathbf{s})_m^{n+1} = \left[\mathbf{Q} + h\mathcal{K}_2\right](\mathbf{s})_m^n. \tag{2.32}$$

Making the appropriate substitutions, one finds the explicit form of the stability matrix:

$$\mathbf{\Phi}(z) = \mathbf{Q} + h\left[(\mathbf{P}_{\text{diag}}\mathbf{Q} + \mathbf{P}_{\text{offdiag}})\left(\mathbf{I} + \frac{h}{2}\mathbf{P}\right)\right]. \tag{2.33}$$

The largest eigenvalues of (2.33), along with the results of a numerical simulation, are shown in Fig. 2.5. From the plot and the analysis in Appendix B, one can see

that the growth rate of the instability in the MoC-MP is $O(h)$, the same as in the MoC-SE and MoC-ME.

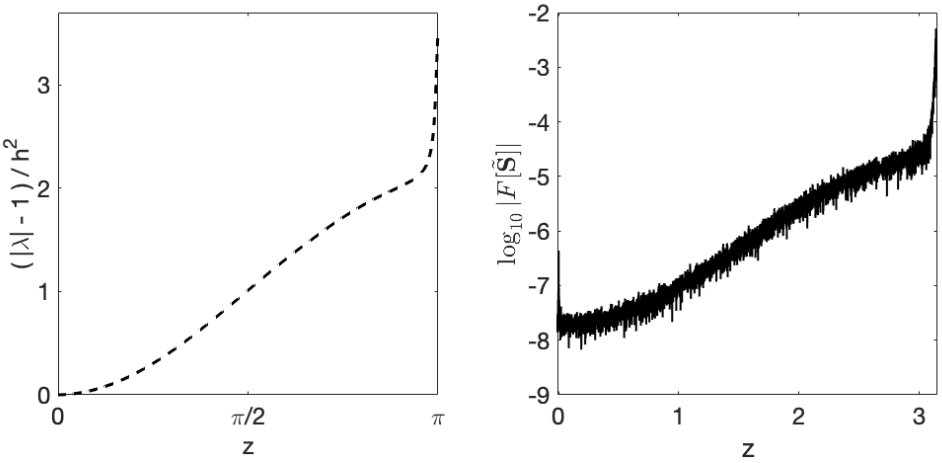

Figure 2.5: The left pane shows the largest eigenvalue of (2.33) over the Fourier spectrum with $h = 0.01$. The strongly unstable are the highest Fourier harmonics with $z \lesssim \pi$. The right pane shows the logarithm of the Fourier spectrum of error at $t = 350$ that results from integrating (2.10) with the MoC-MP. If one were to integrate to $t = 500$ as we did with the MoC-SE and MoC-ME, the solution would be completely destroyed.

2.4 Concluding remarks

We now comment on similarities and differences between the MoC-ME and MoC-MP that motivate the rest of the work presented in this thesis. The ME and MP are both second-order RK methods, and therefore have the same stability characteristics when it comes to solving ODEs. This might lead one to suspect that they share similar stability characteristics when used in the MoC. However, as shown in Sec. 2.3, this is not the case, as the MoC-ME has a mild instability for modes in the center of the Fourier spectrum, while the MoC-MP has a stronger instability at the end of the spectrum.

In Chapter 3, we will present MoC schemes based on third- and fourth-order RK methods, both of which will have a growth rate in their instability that is $O(1)$, rather than $O(h)$. In further chapters, we will shift our focus from explicit RK methods to the so-called pseudo Runge–Kutta methods, whose stability characteristics when used in the MoC will be more akin to those of the MoC-ME.

3 The MoC Scheme using classical RK methods

In Sec. 1.3.2, we explained that only first- and second order-accurate explicit MoC schemes have been reported thus far for systems with two crossing characteristics. In Chapter 2, we described these methods: the MoC-SE, MoC-ME, and MoC-MP. The main focus of this chapter will be to introduce an MoC scheme which uses the classical third-order RK method (cRK3) as the ODE solver. In principle, this is a third-order accurate MoC scheme, but as will be shown, this scheme possesses undesirable qualities in terms of numerical stability. We will also present variations of the cRK3 that can be used in the MoC, and will briefly discuss the construction and results of a fourth-order scheme.

3.1 The classical third-order RK method

The cRK3 method has the Butcher tableau:

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

For clarity, we will write out each stage of the method:

$$k_1 = f(Y_n, t_n), \tag{3.1a}$$

$$k_2 = f\left(Y_n + \frac{h}{2}k_1, t_{n+1/2}\right), \tag{3.1b}$$

$$k_3 = f\left(Y_n - hk_1 + 2hk_2, t_{n+1}\right), \tag{3.1c}$$

$$Y_{n+1} = Y_n + \frac{h}{6}\left(k_1 + 4k_2 + k_3\right). \tag{3.1d}$$

Following the analysis in Sec. 1.5, the cRK3 is stable for

$$\left|1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6}\right| \leq 1. \tag{3.2}$$

The corresponding stability region is plotted in Fig. 3.1, along with the region for the ME.

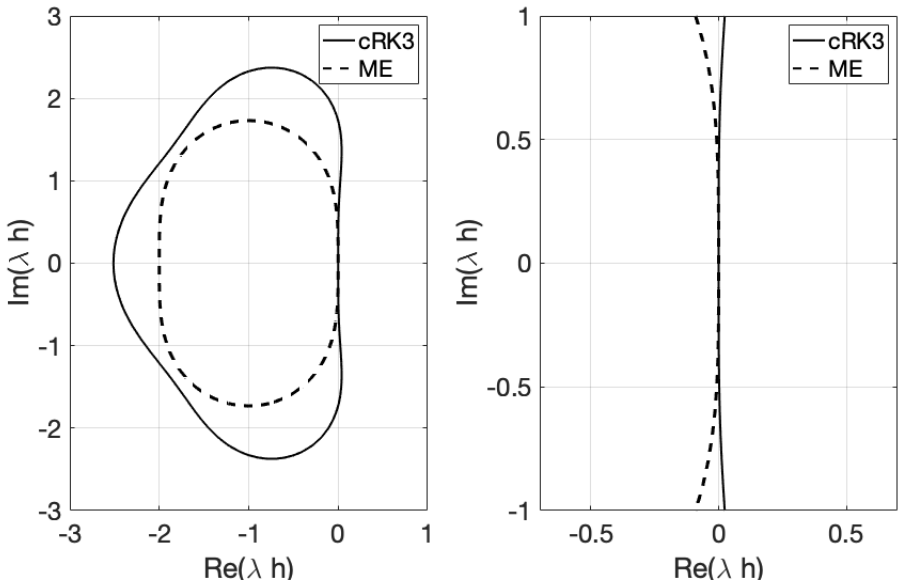

Figure 3.1: Stability regions of the cRK3 and ME. The right pane shows that the cRK3 maintains a higher degree of tangency to the imaginary axis than the ME.

The motivation to develop an MoC scheme using the cRK3 is twofold: first, since the cRK3 is third-order accurate, one will obtain a more accurate numerical solution than one will obtain using a second-order scheme; and second, the cRK3 should be expected to do a better job of preserving some of the conserved quantities admitted by a system of hyperbolic equations. This follows from Fig. 3.1 and the discussion in Sec. 1.5.3.

3.2 The MoC using the cRK3

We now present how one can formulate the MoC using the cRK3. The stencil for this method will be the same stencil used for the MoC-MP. We insert the figure below for the reader's convenience.

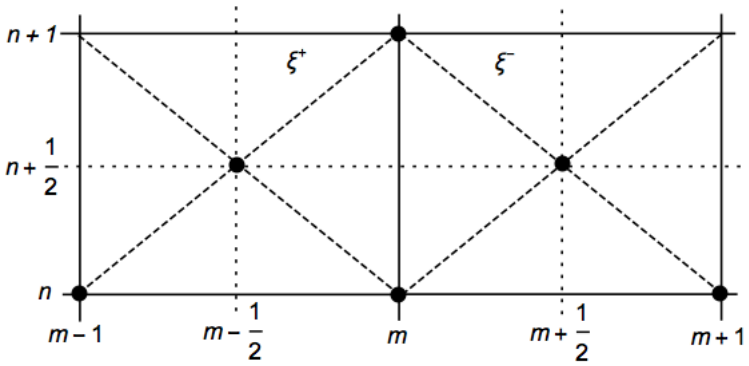

Figure 3.2: Stencil for the MoC-cRK3

Following the analysis in Sec. 2, (3.1a) leads to the following equations in the MoC setup:

$$(k_1^+)_{m-1}^n = f^+\left((Y^+)_{m-1}^n, (Y^-)_{m-1}^n\right), \tag{3.3a}$$

$$(k_1^-)_{m+1}^n = f^-\left((Y^+)_{m+1}^n, (Y^-)_{m+1}^n\right). \tag{3.3b}$$

Eventually, to compute the cRK3 solution at time level $n+1$, we will need auxiliary solutions at that time level with local accuracy $O(h^2)$ and $O(h^3)$. We compute these by the MoC-SE and MoC-ME, respectively. As this will be a common feature of many methods which we develop, we will denote solutions computed by MoC-SE with an overline and solutions computed by MoC-ME with a double-dot.

$$\left(\overline{Y}^{+}\right)_m^{n+1} = (Y^+)_{m-1}^n + h(k_1^+)_{m-1}^n, \tag{3.3c}$$

$$\left(\overline{Y}^{-}\right)_m^{n+1} = (Y^-)_{m+1}^n + h(k_1^-)_{m+1}^n, \tag{3.3d}$$

$$\left(\ddot{Y}^{+}\right)_m^{n+1} = \frac{1}{2}\left[(Y^+)_{m-1}^n + \left(\overline{Y}^{+}\right)_m^{n+1} + hf^+\left(\left(\overline{Y}^{+}\right)_m^{n+1}, \left(\overline{Y}^{-}\right)_m^{n+1}\right)\right], \tag{3.3e}$$

$$\left(\ddot{Y}^{-}\right)_m^{n+1} = \frac{1}{2}\left[(Y^-)_{m+1}^n + \left(\overline{Y}^{-}\right)_m^{n+1} + hf^-\left(\left(\overline{Y}^{+}\right)_m^{n+1}, \left(\overline{Y}^{-}\right)_m^{n+1}\right)\right]. \tag{3.3f}$$

Equations (3.3e) and (3.3f) are different than Eqs. (2.4), which was our formulation of the MoC-ME in Chapter 2. However, it is not difficult to show that the formulations are equivalent. It may be useful to point out that while the (m,n)-index of $k_i^{\pm}$ is the index of the node <u>from</u> which k_i is constructed, the index of $\overline{Y}^{\pm}$ or $\ddot{Y}^{\pm}$ is that of the node <u>at</u> which this auxiliary solution is <u>obtained</u>.

New issues with the MoC-cRK3 begin to arise with the calculation of the second stage derivative, k_2. Let us explain this for k_2^+. According to (3.1b), treating Y^- as a function of time along ξ^+, we need:

$$(k_2^+)_{m-1}^n = f^+\left((Y^+)_{m-1}^n + \frac{h}{2}(k_1^+)_{m-1}^n, (Y^-)_{m-1/2}^{n+1/2}\right). \tag{3.3g}$$

Recall that in the MoC-MP, we encountered the same term $(Y^-)_{m-1/2}^{n+1/2}$, and had to compute it with local accuracy $O(h^2)$. For similar reasons, in the MoC-cRK3 we

must compute this term with local accuracy $O(h^3)$, which thus is new compared to the MoC-MP. One way to compute this point is to first find the point $(\ddot{Y}^-)_{m-1}^{n+1}$ by the MoC-ME, which is a second-order scheme and thus has the local error $O(h^3)$, as desired. Then, using this point and the available solution $(Y^-)_m^n$, one can interpolate the desired term using a Taylor series. We demonstrate this process below.

Expanding the desired term in a third-order Taylor series gives us:

$$\begin{aligned}(Y^-)_{m-1/2}^{n+1/2} &= (Y^-)_m^n + \frac{h}{2}\left[(Y^-)_m^n\right]' + \frac{(h/2)^2}{2}\left[(Y^-)_m^n\right]'' + O(h^3) \\ &= (Y^-)_m^n + \frac{h}{2} f^-((Y^+)_m^n, (Y^-)_m^n) + \frac{h^2}{8}\left[(Y^-)_m^n\right]'' + O(h^3). \end{aligned} \tag{3.4a}$$

Thus, to approximate $(Y^-)_{m-1/2}^{n+1/2}$ with the desired accuracy $O(h^3)$, we need to know each of the three terms on the r.h.s. of (3.4a). We know the first two terms but not the third. To obtain the yet unknown $[(Y^-)_m^n]''$, we expand the term that we have already computed with local accuracy $O(h^3)$, $(\ddot{Y}^-)_{m-1}^{n+1}$, in a Taylor series:

$$(\ddot{Y}^-)_{m-1}^{n+1} = (Y^-)_m^n + h f^-((Y^+)_m^n, (Y^-)_m^n) + \frac{h^2}{2}\left[(Y^-)_m^n\right]'' + O(h^3). \tag{3.4b}$$

Solving (3.4b) for $[(Y^-)_m^n]''$, substituting the result into (3.4a), and simplifying yields:

$$(Y^-)_{m-1/2}^{n+1/2} = \frac{3}{4}(Y^-)_m^n + \frac{1}{4}(\ddot{Y}^-)_{m-1}^{n+1} + \frac{h}{4} f^-((Y^+)_m^n, (Y^-)_m^n). \tag{3.3h}$$

Therefore, one can compute (3.3g) with the desired accuracy using (3.3h). By the same token, the second stage derivative for the negative component,

$$(k_2^-)_{m+1}^n = f^-\left((Y^+)_{m+1/2}^{n+1/2}, (Y^-)_{m+1}^n + \frac{h}{2}(k_1^-)_{m+1}^n\right), \tag{3.3i}$$

can be computed using:

$$(Y^+)^{n+1/2}_{m+1/2} = \frac{3}{4}(Y^+)^n_m + \frac{1}{4}(\ddot{Y}^-)^{n+1}_{m+1} + \frac{h}{4}f^+((Y^+)^n_m, (Y^-)^n_m). \tag{3.3j}$$

According to (3.1c), to compute $(k_3^{\pm})^n_{m\mp1}$, we need the solution at time level $n+1$ with local accuracy $O(h^2)$. Conveniently, this has already been computed by the MoC-ME in (3.3{e,f}), so we have:

$$(k_3^+)^n_{m-1} = f^+\left((Y^+)^n_{m-1} - h((k_1^+)^n_{m-1} - 2(k_2^+)^n_{m-1}), (\ddot{Y}^-)^{n+1}_m\right), \tag{3.3k}$$

$$(k_3^-)^n_{m+1} = f^-\left((\ddot{Y}^+)^{n+1}_m, (Y^-)^n_{m+1} - h((k_1^-)^n_{m+1} - 2(k_2^-)^n_{m+1})\right). \tag{3.3l}$$

Finally, by (3.1d),

$$(Y^+)^{n+1}_m = (Y^+)^n_{m-1} + \frac{h}{6}\left((k_1^+)^n_{m-1} + 4(k_2^+)^n_{m-1} + (k_3^+)^n_{m-1}\right), \tag{3.3m}$$

$$(Y^-)^{n+1}_m = (Y^-)^n_{m+1} + \frac{h}{6}\left((k_1^-)^n_{m+1} + 4(k_2^-)^n_{m+1} + (k_3^-)^n_{m+1}\right). \tag{3.3n}$$

Equations (3.3) form the MoC-cRK3. Notice that, when designing this method, we had three degrees of freedom: choosing how to compute $(Y^{\pm})^{n+1/2}_{m\mp1/2}$, choosing which second-order method to use in this computation, and choosing which second-order method to use in the computation of k_3. We will exploit these degrees of freedom later on in this chapter.

3.3 von Neumann analysis of the MoC-cRK3

By looking at Fig. 3.1, one might make the correct assumption that the cRK3 would be a better method to use when solving ODEs than the ME if one hoped to preserve stability. One, however, cannot apply this logic when using the cRK3 in the MoC, as we will show in this subsection.

Let us analyze the stability of the cRK3 when applied to the system (1.50). Following the procedure of the von Neumann analysis presented in Chapter 2, we find the stability matrix

$$\mathbf{\Phi}(z) = \mathbf{Q} + \frac{h}{6}\left[\mathcal{K}_1 + 4\mathcal{K}_2 + \mathcal{K}_3\right], \tag{3.4}$$

where $\mathbf{Q}$ is defined in (2.21), $\mathcal{K}_{\{1,2,3\}}$ are given by

$$\mathcal{K}_1 = \mathbf{Q}\mathbf{P}, \tag{3.5a}$$

$$\mathcal{K}_2 = \mathbf{P}_{\text{diag}}\left(\mathbf{Q} + \frac{h}{2}\mathcal{K}_1\right) + \mathbf{P}_{\text{offdiag}}\mathcal{S}_1, \tag{3.5b}$$

$$\mathcal{K}_3 = \mathbf{P}_{\text{diag}}\mathcal{S}_2 + \mathbf{P}_{\text{offidag}}\mathcal{S}_3, \tag{3.5c}$$

$\mathbf{P}$ is defined in (2.12),

$$\mathcal{S}_1 = \frac{1}{8}\left[8\mathbf{I} + (4\mathbf{I} + h\mathbf{P})h\mathbf{P}\right], \tag{3.5d}$$

$$\mathcal{S}_2 = \mathbf{Q} - h\left(\mathcal{K}_1 - 2\mathcal{K}_2\right), \tag{3.5e}$$

$$\mathcal{S}_3 = \frac{1}{2}\left[\mathbf{Q} + (\mathbf{I} + h\mathbf{P})\mathbf{Q}(\mathbf{I} + h\mathbf{P})\right], \tag{3.5f}$$

and $\mathbf{I}$ is the 4×4 identity matrix. The von Neumann stability plot is shown in Fig.

3.3.

Following the analysis of Appendix B, one finds that the growth rate of the instability in the MoC-cRK3 is $O(1)$. Let us explain why this is so. In the equation

$$|\lambda| = 1 + ch^{\alpha}, \tag{3.6}$$

we seek the exponent α. Fig. 3.3 tells us

$$\frac{|\lambda| - 1}{h_1^2} = 16 \quad \text{and} \quad \frac{|\lambda| - 1}{h_2^2} = 8, \tag{3.7}$$

where $h_1 = 0.01$ and $h_2 = 0.02$. Using (3.6), one finds from (3.7) that $ch_1^{\alpha}/h_1^2 = 16$ and $ch_2/h_2^2 = 8$, or $h_1^{2-\alpha}/h_2^{2-\alpha} = 16/8$, whence $\alpha = 1$. Therefore,

$$|\lambda|^n = |\lambda|^{t/h} = (1 + ch)^{t/h} \approx e^{ct}, \tag{3.8}$$

meaning that the growth rate in the error of the solution is $c = O(1)$. Indeed, following the analysis of Appendix B, for $h = 0.01$ one has $c \approx 0.17$. Due to this large (i.e., $O(1)$)) growth rate, any numerical solution will turn into complete garbage rather quickly. This is shown in Fig. 3.4.

3.4 Alternative formulations of the MoC-cRK3

At the end of Sec. 3.2, we mentioned that one has many degrees of freedom when designing the MoC-cRK3. In this subsection, we take advantage of these freedoms and present alternative formulations of the MoC-cRK3, and then comment on the stability of the resulting methods.

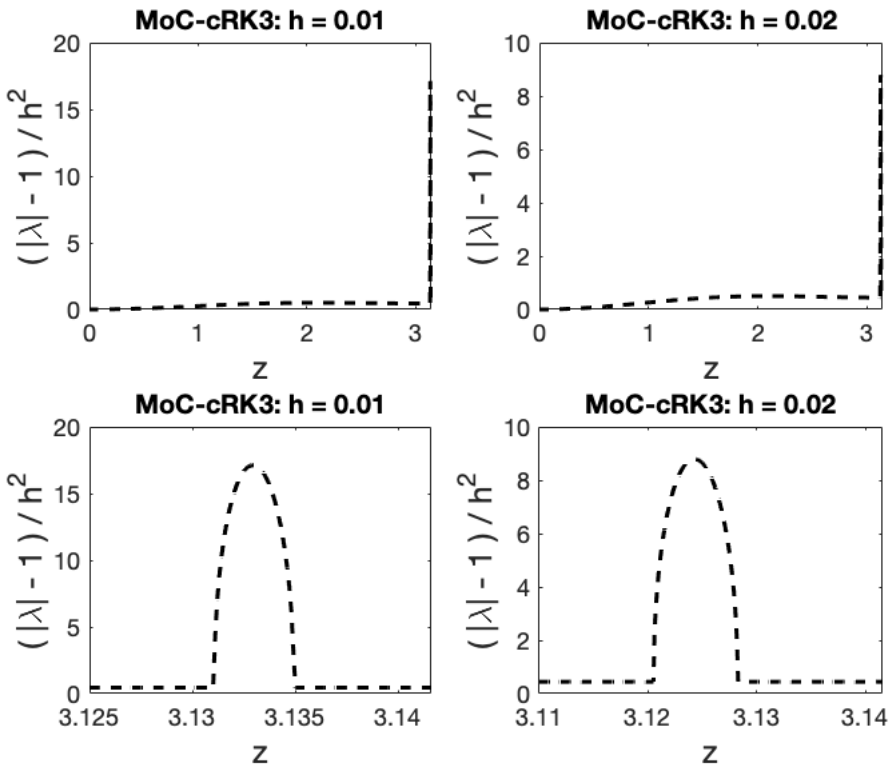

Figure 3.3: von Neumann stability plot for the cRK3 using $h = 0.01$ and $h = 0.02$. In both cases, we see a large instability at the harmonics with $z \lesssim \pi$. This is similar to the behavior we observed in the MoC-MP, except here the instability is significantly larger.

3.4.1 Second-order solver used in the MoC-cRK3

Recall that in the computation of k_2 at time level n (see (3.1b),(3.3g),(3.3i)), we needed a solution at time level $n + 1/2$ with local accuracy $O(h^3)$. The computation of this point required the solution at time level $n + 1$ with local accuracy $O(h^3)$, which was used again in k_3. In our previous formulation of the MoC-cRK3, we computed the $O(h^3)$-local accurate solution at level $n + 1$ by the MoC-ME. Any second-order method would have been appropriate for this computation, though, such as the MoC-MP. When the MoC-MP is used, we refer to the method as MoC-

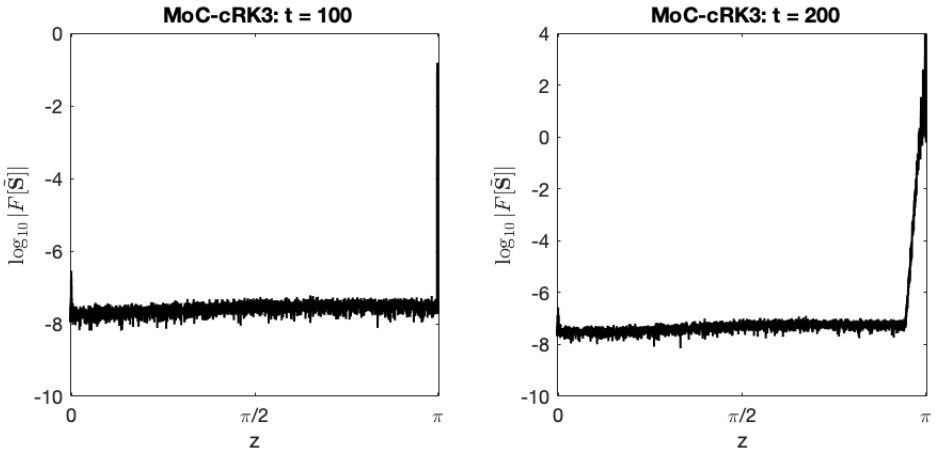

Figure 3.4: Fourier spectrum of the error produced by the MoC-cRK3 when solving (1.50) using $h = 0.01$. The left pane shows the results at $t = 100$, and the right pane shows the results at $t = 200$. In the left pane, notice the instability at $z \lesssim \pi$, which agrees with the von Neumann analysis (Fig. 3.3). By $t = 200$, this instability has grown quite large, and has begun to infect the spectrum at nearby harmonics. The maximum error in the solution here is much greater than the exact solution itself, so the solution is essentially destroyed. By $t = 250$, this error will explode and become too large for the machine to handle.

MPA-cRK3, where “MPA" stands for “Midpoint-Assisted." (Following this notation, the scheme we presented in Sec. 3.2 would be called the MoC-MEA-cRK3. After this chapter, this will be the only MoC-RK3 scheme we use again, and thus will omit the -MEA when referring to this scheme.)

The equations for the MoC-MPA-cRK3 are the same as Eqs. (3.3), except we replace (3.3{c,d,e,f}) with the MoC-MP. This looks like:

$$\left(\overline{Y}^{+}\right)_{m+1/2}^{n+1/2} = (Y^{+})_{m}^{n} + \frac{h}{2} f^{+}\left((Y^{+})_{m}^{n}, (Y^{-})_{m}^{n}\right), \tag{3.6a}$$

$$\left(\overline{Y}^{-}\right)_{m-1/2}^{n+1/2} = (Y^{-})_{m}^{n} + \frac{h}{2} f^{-}\left((Y^{+})_{m}^{n}, (Y^{-})_{m}^{n}\right), \tag{3.6b}$$

$$\left(\ddot{Y}^{+}\right)_{m}^{n+1} = (Y^{+})_{m-1}^{n} + h f^{+}\left(\left(\overline{Y}^{+}\right)_{m-1/2}^{n+1/2}, \left(\overline{Y}^{-}\right)_{m-1/2}^{n+1/2}\right), \tag{3.6c}$$

$$\left(\ddot{Y}^{-}\right)_{m}^{n+1} = (Y^{-})_{m+1}^{n} + h f^{-}\left(\left(\overline{Y}^{+}\right)_{m+1/2}^{n+1/2}, \left(\overline{Y}^{-}\right)_{m+1/2}^{n+1/2}\right). \tag{3.6d}$$

(Eqs. (3.6) are equivalent to the MoC-MP derived in Chapter 2.) The MoC-MP solution is given by $(\ddot{Y}^{\pm})_m^{n+1}$, and one uses this exactly how is written in the remainder of Eqs. (3.3). The results of the von Neumann stability analysis of the MoC-MPA-cRK3 will be shown later on.

3.4.2 Other RK3 schemes

Thus far, we have only presented third-order MoC schemes based on the *classical* RK3, (3.1). One can, however, use any RK3 scheme. Any explicit RK3 scheme has the form

$$\begin{array}{c|ccc} 0 & & & \\ b_2 & a_{21} & & \\ b_3 & a_{31} & a_{32} & \\ \hline & c_1 & c_2 & c_3 \end{array} \tag{3.7}$$

If one takes b_2 and b_3 as free parameters, it can be shown [7] that the remaining coefficients can be found by solving the six equations

$$a_{21} = b_2, \tag{3.8a}$$

$$a_{31} + a_{32} = b_3, \tag{3.8b}$$

$$b_2 c_2 + b_3 c_3 = \frac{1}{2}, \tag{3.8c}$$

$$b_2^2 c_2 + b_3^2 c_3 = \frac{1}{3}, \tag{3.8d}$$

$$a_{32} b_2 c_3 = \frac{1}{6}, \tag{3.8e}$$

$$c_1 + c_2 + c_3 = 1. \tag{3.8f}$$

When designing MoC schemes, it is convenient to have b_2 and b_3 equal 1/2 or 1 (otherwise, the numerical grid becomes very complicated). Taking $b_2 = 1/2$ and $b_3 = 1$ yields the cRK3. Taking $b_2 = 1$ and $b_3 = 1/2$ yields the following scheme, known as the Strong Stability Preserving RK3 (SSPRK3).

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array} \tag{3.9}$$

Deriving the MoC-SSPRK3 follows the same process as the MoC-cRK3: one must first compute solutions with local accuracy $O(h^3)$ at time levels $n+1$ and $n+1/2$, then appropriately evaluate the functions and combine terms. Later, we will show the results of the von Neumann analysis of the MoC-MEA-SSPRK3 and MoC-MPA-SSPRK3.

3.4.3 Computation of the solution at time level $n + 1/2$

In all of the MoC-RK3 schemes presented thus far, when computing the solution at time level $n+1$, we have needed an $O(h^3)$-local accurate solution at the time level $n + 1/2$. As described in Sec. 3.2, we interpolated this point through the use of a Taylor series. This is, however, only one of the many ways this can be done.

Recall that in the MoC-MP (a second-order method), we faced a similar problem where we needed the solution at time level $n + 1/2$ with local accuracy $O(h^2)$. This was accomplished by using the MoC-SE (a first-order method) with stepsize $h/2$. In complete analogy, in MoC-RK3 schemes, we can use any second-order method with

stepsize $h/2$ to compute the desired point.

For example, using the MoC-ME with stepsize $h/2$, one has:

$$\left(\ddot{Y}^{+}\right)_{m+1/2}^{n+1/2} = \frac{1}{2}\left[(Y^{+})_{m}^{n} + \frac{h}{2}f^{+}\left(\left(\overline{Y}^{+}\right)_{m+1/2}^{n+1/2}, \left(\overline{Y}^{-}\right)_{m+1/2}^{n+1/2}\right)\right], \tag{3.10a}$$

$$\left(\ddot{Y}^{-}\right)_{m-1/2}^{n+1/2} = \frac{1}{2}\left[(Y^{-})_{m}^{n} + \frac{h}{2}f^{-}\left(\left(\overline{Y}^{+}\right)_{m-1/2}^{n+1/2}, \left(\overline{Y}^{-}\right)_{m-1/2}^{n+1/2}\right)\right], \tag{3.10b}$$

where the $\left(\overline{Y}^{\pm}\right)^{n+1/2}$ are identical to those computed in (3.6a) and (3.6b). One could also, of course, use the MoC-MP with stepsize $h/2$. We will present results using the MoC-ME in the MoC-cRK3, and will refer to the resulting method as MoC-cRK3-ME/2

3.4.4 von Neumann stability analysis of other MoC-RK3 schemes

In this subsection, we present the results of the von Neumann stability analysis of four of the aforementioned methods: MoC-MPA-cRK3, MoC-MEA-SSPRK3, MoC-MPA-SSPRK3, and MoC-cRK3-ME/2. We omit the details of the stability analysis, as they are similar to those given in Sec. 3.3. The von Neumann stability plots are shown in Fig. 3.5.

Based on the results shown in Fig. 3.5, we see that there is no obvious advantage in using one type of MoC-RK3 scheme over another. Due to the growth rate of the instability being $O(1)$, the numerical solution of (1.50) will be destroyed after a relatively short computation time if these methods are used.

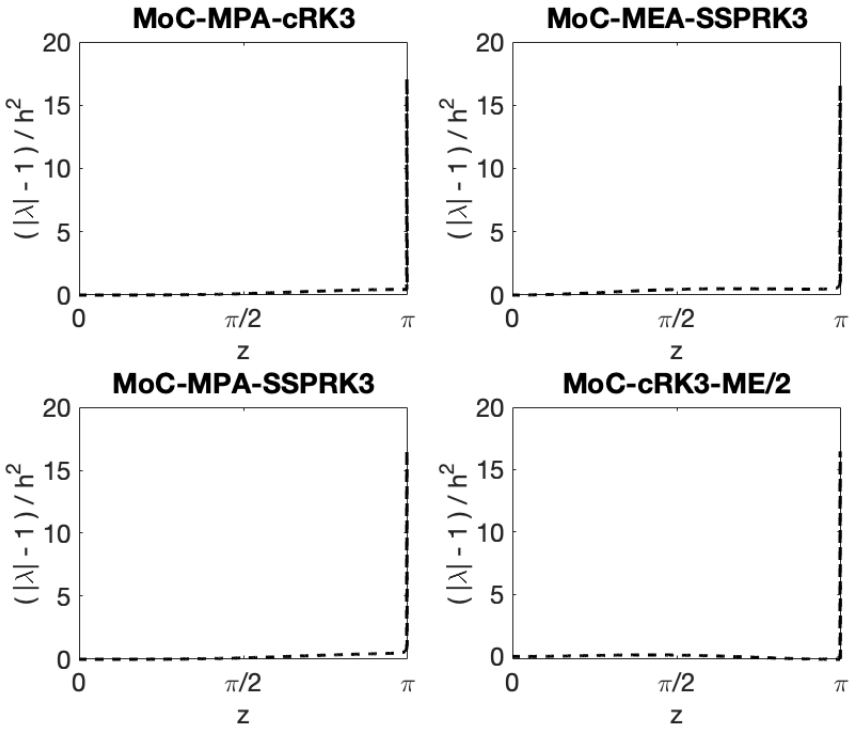

Figure 3.5: von Neumann stability plots of the other MoC-RK3 schemes, using $h = 0.01$. They are all qualitatively similar to the stability plot for the MoC-cRK3 (Fig. 3.4), with a large instability at $z \lesssim \pi$. These instabilities all have the growth rate $O(1)$.

3.5 The MoC using the classical RK4

In this section, we will briefly describe an MoC scheme using the classical fourth-order RK (cRK4) method as the ODE solver. The brevity of our discussion here is due to the fact that the MoC-cRK4 exhibits the same strong instability as the MoC-cRK3 and thus possesses no advantage over the latter scheme for long-time computation.

The cRK4 is perhaps the most well-known explicit RK method. Its Butcher

tableau is given by:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{3.11}$$

Examining the left-most column of the tableau, one can deduce that in the MoC-cRK4, one will need the solutions $(Y^{\pm})^{n+1/2}$ and $(Y^{\pm})^{n+1}$ with local accuracy $O(h^4)$. Methods to find the $(Y^{\pm})^{n+1/2}$ solution include interpolation using a fourth-order Taylor series, or calculation using an MoC-RK3 scheme with stepsize $h/2$. The latter method requires the solution $(Y^{\pm})^{n+1/4}$, so in practice the former method is easier to implement. To compute $(Y^{\pm})^{n+1}$, one simply uses their favorite MoC-RK3 scheme and then interpolates similarly to (3.3h) and (3.3j). Appendix C shows this interpolation.

Figure 3.6 shows the results of the von Neumann stability analysis applied to the MoC-cRK4 using Taylor series interpolation and MoC-cRK3. These plots are similar to those produced in the analysis of the MoC-cRK3, so we conclude that the MoC-cRK4 is similarly strongly unstable.

3.6 Concluding remarks

One of the take home-messages of Chapters 2 and 3 is this: MoC schemes using explicit RK methods as the ODE solver tend to be unstable. Therefore, we abandon all hope of developing a stable MoC scheme based on any higher-order explicit RK

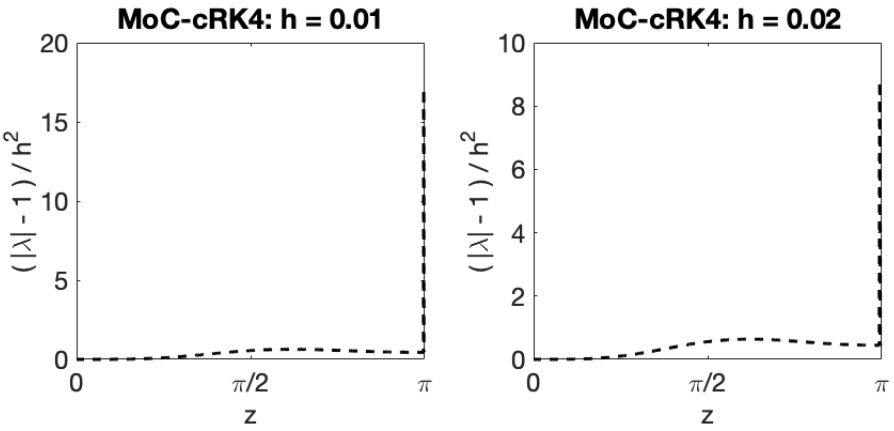

Figure 3.6: von Neumann stability plots of the MoC-cRK4 with different values of h. We see the same type of instability that arises in the MoC-RK3 schemes, as well as a growth in the instability that is $O(1)$.

schemes. Fortunately, this does not mean that the main objective of this thesis has been compromised. In Chapter 4, we will discuss pseudo-RK methods, which are explicit methods based on RK and multistep methods, in the ODE context. In Chapters 5 and 6, we will show that stable third- and fourth-order MoC schemes can be developed using pseudo-RK methods as the ODE solver.

For the developments in the subsequent chapters, it is important to reflect on the question: *Why* did the MoC-RK3 and MoC-cRK4 schemes essentially fail (to be stable)? We do *not* know an answer to this. However, *from our extensive experimentation* with various versions of these schemes, we have *surmised* that the culprit may be the fact that they all required evaluation of the solution at a virtual node such as $(m+1/2, n+1/2)$. We have *accepted this hypothesis* and therefore seek methods that require solution values *only at the actual nodes* (like the MoC-ME). This leads us to consider the pseudo-RK methods, described next.

4 Pseudo-Runge–Kutta Methods for ODEs

In this chapter, we will present the idea behind pseudo-Runge–Kutta (pRK) schemes for the numerical integration of ODEs. As we will show in Chapter 5, the MoC-pRK schemes for hyperbolic PDEs (1.6) are free of the strong instability that we discovered and described in Chapter 3.

To begin, we will present third- and fourth-order pRK schemes first proposed in [8]. Then, we will explain a modified version of these methods described in [9]. After this, we will derive our own fourth-order pRK scheme. Finally, we will investigate the stability of these methods and present some numerical results.

4.1 Framework of pRK schemes

The main idea behind pRK schemes is simple. Recall that in explicit RK schemes, one computes the numerical solution at the next step, Y_{n+1}, by strategically combining certain function evaluations involving the available solution, Y_n. In pRK schemes, one advances to Y_{n+1} by using the information available at Y_n and that which is available at previous nodes (e.g. Y_{n-1}). The Butcher tableau of a pRK scheme takes the form

below.

$$
\begin{array}{c|cccc}
0 & & & & \\
b_2 & a_{21} & & & \\
b_3 & a_{31} & a_{32} & & \\
\vdots & \vdots & \vdots & \ddots & \\
b_m & a_{m1} & a_{m2} & \cdots & a_{m(m-1)} \\
\hline
 & c_{11} & c_{12} & \cdots & c_{1m} \\
 & c_{21} & c_{22} & \cdots & c_{1m} \\
 & \vdots & \vdots & \ddots & \vdots \\
 & c_{s1} & c_{s2} & \cdots & c_{sm}
\end{array}
$$

The upper part of this tableau has the same meaning as usual (see Sec. 1.4) and governs how one computes the stage derivatives $k_i,\ i = 1, \ldots, m$. The lower portion involving the terms $c_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, s$, has the following meaning:

$$Y_{n+1} = Y_n + h \sum_{j=1}^{s} \sum_{i=1}^{m} c_{ji} (k_i)_{n-s+1} \tag{4.1}$$

where $(k_i)_{n-s+1}$ refers to stage derivative i computed with the solution Y_{n-s+1}. In this thesis, s will never be greater than 3.

4.2 pRK methods of Byrne and Lambert

The first people to propose the framework discussed in Sec. 4.1 were G. Byrne and R. Lambert in [8]. In this section, we will present their third- and fourth-order methods.

4.2.1 The third-order pRK scheme of Byrne and Lambert

Byrne and Lambert's third-order scheme involves the computation of two stage derivatives at the current step and the use of these derivatives found in the previous step. Thus, the tableau has the following form:

$$\begin{array}{c|cc} 0 & & \\ b_2 & a_{21} & \\ \hline & c_{11} & c_{12} \\ & c_{21} & c_{22} \end{array} \tag{4.2}$$

The coefficients in method (4.2) are entirely determined by b_2 and are given by:

$$a_{21} = b_2, \tag{4.3a}$$

$$c_{11} = \frac{-5 + 18b2}{12b_2}, \tag{4.3b}$$

$$c_{12} = \frac{5}{12b_2}, \tag{4.3c}$$

$$c_{21} = \frac{5 - 6b_2}{12b_2}, \tag{4.3d}$$

$$c_{22} = -\frac{5}{12b_2}. \tag{4.3e}$$

When developing an MoC scheme, it is beneficial for the user to have a method with $b_2 = 1$. Indeed, according to (1.27b), the stage derivative k_2 is evaluated at $t_n + b_2 h$. Thus, using $b_2 = 1$ obviates the need to evaluate the solution outside of the actual nodes. As noted in Sec. 3.6, this is precisely the property that we seek.

According to Eqs. (4.3), method (4.2) with $b_2 = 1$ is:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{13}{12} & \frac{5}{12} \\ & \frac{-1}{12} & \frac{-5}{12} \end{array} \tag{4.4}$$

We will refer to method (4.4) as *the* pRK3. For clarity, the equations of the pRK3 for the ODE $y' = f(Y, t)$ are:

$$(k_1)_n = f(Y_n, t_n), \tag{4.5a}$$

$$(k_2)_n = f(Y_n + h(k_1)_n, t_n + h), \tag{4.5b}$$

$$Y_{n+1} = Y_n + \frac{h}{12}\left(13(k_1)_n + 5(k_2)_n - (k_1)_{n-1} - 5(k_2)_{n-1}\right). \tag{4.5c}$$

Notice that to obtain a third-order pRK scheme, one only needs two stage derivatives, as opposed to an RK scheme where one needed three stages (see Secs. 3.1 and 3.4.2). In the numerical methods examined in this thesis, it will always be the case that the order of a pRK method will be given by $m + s - 1$, where m and s are defined in (4.1). (This also applies to the RK methods studied earlier, since in their case s was equal to 1.)

Let us also comment upon how one may start a pRK method. In the pRK3, for example, one needs the solution at the current and previous stage in order to advance to the next stage. Thus, one cannot advance to Y_1 from the initial condition Y_0 by the pRK. Since this one step needs to be computed with local accuracy $O(h^3)$, one can use any third-order local accurate method, such as ME, to take this step.

4.2.2 The fourth-order pRK scheme of Byrne and Lambert

Byrne and Lambert's fourth-order scheme involves three stages, and therefore will require the use of stage values found at the previous step. The tableau has the form

$$\begin{array}{c|ccc} 0 & & & \\ b_2 & a_{21} & & \\ b_3 & a_{31} & a_{32} & \\ \hline & c_{11} & c_{12} & c_{13} \\ & c_{21} & c_{22} & c_{23} \end{array} \tag{4.6}$$

The coefficients are entirely determined by b_2 and b_3 and are given by:

$$a_{21} = b_2, \tag{4.7a}$$

$$a_{32} = \frac{2b_3(b_3 - b_2)}{b_2(4 - 5b_2)}, \tag{4.7b}$$

$$a_{31} = b_3 - a_{32}, \tag{4.7c}$$

$$c_{11} = \frac{4 - 5b_2 - 5b_3 + 18b_2b_3}{12b_2b_3}, \tag{4.7d}$$

$$c_{12} = \frac{4 - 5b_3}{12b_2(b_2 - b_3)}, \tag{4.7e}$$

$$c_{13} = \frac{5b_2 - 4}{12b_3(b_2 - b_3)}, \tag{4.7f}$$

$$c_{21} = 1 - c_{11}, \tag{4.7g}$$

$$c_{22} = -c_{12}, \tag{4.7h}$$

$$c_{23} = -c_{13}. \tag{4.7i}$$

As mentioned in the previous subsection, for the sake of the MoC it would be ideal to have $b_2 = b_3 = 1$. However, this is impossible according to Eqs. (4.7). Later in this chapter, we will develop a fourth-order pRK scheme that only involves two stages and $b_2 = 1$, as opposed to the three-stage method (4.6). This method is more favorable in the MoC, and thus we will not mention the fourth-order methods of Byrne and Lambert again.

4.3 pRK methods of Nakashima

A modification of Byrne and Lambert's pRK framework is given by M. Nakashima in [9]. Many examples of the Nakashima methods are given in [10], [11]. Nakashima methods follow the same idea as pRK methods, but they use the values of Y_{n-1} and $(k_1)_{n-1}$ in the computation of $(k_2)_n$. In this section, we present third- and fourth-order Nakashima methods.

4.3.1 The third-order pRK scheme of Nakashima

The third-order Nakashmia pRK method (NpRK3) has the form:

$$(k_1)_n = f(Y_n, t_n), \tag{4.8a}$$

$$(k_2)_n = f\Big(Y_n + \Lambda(Y_n - Y_{n-1}) + h(a_{20}(k_1)_{n-1} + a_{21}(k_1)_n), t_n + b_2 h\Big), \tag{4.8b}$$

$$Y_{n+1} = Y_n + h\Big(c_{11}(k_1)_n + c_{12}(k_2)_n + c_{21}(k_1)_{n-1}\Big). \tag{4.8c}$$

The meaning of the $Y_n - Y_{n-1}$ term (4.8b) is the following: according to (4.8c), one has:

$$Y_n = Y_{n-1} + h(c_{11}(k_1)_{n-1} + c_{12}(k_2)_{n-1} + c_{21}(k_1)_{n-2}).$$

Thus, the term $Y_n - Y_{n-1}$ is adding stage derivatives from the previous *two* steps in the computation of $(k_2)_n$. This allows one to avoid the use of $(k_2)_{n-1}$ when computing Y_{n+1}, as opposed to the pRK methods discussed in Sec. 4.2.

In what follows, we will enforce the condition that $b_2 = 1$ so that the method can be readily used in the MoC: see Sec. 3.6. Imposing this condition, one obtains the free parameter a_{20} and the coefficients

$$a_{21} = 2 + a_{20}, \tag{4.9a}$$

$$\Lambda = -1 - 2a_{20}, \tag{4.9b}$$

$$c_{11} = \frac{2}{3}, \tag{4.9c}$$

$$c_{12} = \frac{5}{12}, \tag{4.9d}$$

$$c_{21} = -\frac{1}{12}. \tag{4.9e}$$

Unlike the Byrne and Lambert pRK methods, the Nakashima methods give the user a free parameter. This free parameter can affect stability properties of the method, so it must be chosen wisely. This will be discussed later on in this chapter.

4.3.2 The fourth-order pRK scheme of Nakashima

The fourth-order Nakashima pRK method (NpRK4) is similar to the NpRK3, except one uses the value $(k_1)_{n-2}$ when computing Y_{n+1}. The method has the form:

$$(k_1)_n = f(Y_n, t_n), \tag{4.10a}$$

$$(k_2)_n = f\Big(Y_n + \Lambda(Y_n - Y_{n-1}) + h(a_{20}(k_1)_{n-1} + a_{21}(k_1)_n), t_n + b_2 h\Big), \tag{4.10b}$$

$$Y_{n+1} = Y_n + h\Big(c_{11}(k_1)_n + c_{12}(k_2)_n + c_{21}(k_1)_{n-1} + c_{22}(k_2)_{n-1} + c_{31}(k_1)_{n-2}\Big). \tag{4.10c}$$

When one imposes $b_2 = 1$, either one obtains the free parameter Λ and the equations

$$a_{20} = \frac{-1-\Lambda}{2}, \tag{4.11a}$$

$$a_{21} = \frac{3-\Lambda}{2}, \tag{4.11b}$$

$$c_{11} = \frac{7}{6}, \tag{4.11c}$$

$$c_{12} = \frac{3}{8}, \tag{4.11d}$$

$$c_{21} = -\frac{5}{24}, \tag{4.11e}$$

$$c_{22} = -\frac{3}{8}, \tag{4.11f}$$

$$c_{31} = \frac{1}{24} \tag{4.11g}$$

or one obtains the free parameter c_{22} and the equations

$$a_{20} = 2, \tag{4.12a}$$

$$a_{21} = 4, \tag{4.12b}$$

$$\Lambda = -5, \tag{4.12c}$$

$$c_{11} = \frac{19 - 24c_{22}}{24}, \tag{4.12d}$$

and the same constants $c_{12,21,31}$ given by (4.11{d,e,g}). We will discuss later how the free parameters can be chosen wisely.

4.4 A fourth-order pRK scheme following the framework of (4.1)

In this section, we seek a fourth-order pRK scheme that follows more closely the framework presented in Sec. 4.1 and the idea of Byrne and Lambert than that of Nakashima. In Sec. 4.2.1, we stated that the order of pRK methods in this thesis are given by $m + s - 1$. This means that we should be able to discover a fourth-order method by taking $m = 2$ and $s = 3$. We would also like to impose the condition

$b_2 = 1$. Therefore, we seek a method of the form:

$$\begin{array}{c|cc} 0 & & \\ 1 & a_{21} & \\ \hline & c_{11} & c_{12} \\ & c_{21} & c_{22} \\ & c_{31} & c_{32} \end{array} \tag{4.13}$$

The equations of this method will look like:

$$(k_1)_n = f(Y_n, t_n), \tag{4.14a}$$

$$(k_2)_n = f\Big(Y_n + a_{21}h(k_1)_n, t_n + h\Big), \tag{4.14b}$$

$$\begin{aligned} Y_{n+1} = Y_n + h\Big(&c_{11}(k_1)_n + c_{12}(k_2)_n + c_{21}(k_1)_{n-1} + c_{22}(k_2)_{n-1} + \\ &c_{31}(k_1)_{n-2} + c_{32}(k_2)_{n-2}\Big). \end{aligned} \tag{4.14c}$$

Our goal is to find the coefficients in (4.13) and (4.14) which yield an $O(h^5)$ local-accurate scheme. In other words, we require

$$y_{n+1} - Y_{n+1} = O(h^5), \tag{4.15}$$

where y_n is the true solution of the ODE $y' = f(y, t)$. To do this, we will expand each term on the l.h.s. of (4.15) in a Taylor series and find conditions on the unknown coefficients so that (4.15) is satisfied.

Expanding the first term on the l.h.s. of (4.15) in a Taylor series yields:

$$y_{n+1} = y_n + hy'_n + \frac{h^2}{2}y''_n + \frac{h^3}{6}y^{(3)}_n + \frac{h^4}{24}y^{(4)}_n + O(h^5). \tag{4.16}$$

Using the fact that $y' = f(y,t)$ and the chain rule for functions of two variables, one finds

$$y'' = f_y f + f_t, \tag{4.17a}$$

$$y^{(3)} = f_{yy}f^2 + f_t f_y + f(f_y^2 + 2f_{ty}) + f_{tt}, \tag{4.17b}$$

$$\begin{aligned} y^{(4)} = {} & f_{yyy}f^3 + f_t f_y^2 + 3f_t f_{ty} + f^2(4f_y f_{yy} + 3f_{tyy}) + \\ & f_{tt}f_y + f\left(f_y^3 + 5f_y f_{ty} + 3(f_t f_{yy} + f_{tty})\right). \end{aligned} \tag{4.17c}$$

Equations (4.17) can be substituted into (4.16) to find an expression for the first term on the l.h.s. of (4.15).

To find an expression for the second term on the l.h.s. of (4.15), one assumes that

$$Y_m = y_n \quad \text{for } m \le n. \tag{4.18}$$

Then, using *Mathematica*, one has (omitting the subscript n on the r.h.s.):

$$(k_1)_n = f, \tag{4.19a}$$

$$\begin{aligned} (k_2)_n = {} & \frac{1}{6}h(h(a_{21}{}^3 f^3 h f_{yyy} + 3a21f(a_{21}f(hf_{yyt} + f_{yy}) + \\ & hf_{ytt} + 2f_{yt}) + hf_{ttt} + 3f_{tt}) + 6a_{21}ff_y + 6f_t) + f + O(h^4), \end{aligned} \tag{4.19b}$$

$$(k_1)_{n-1} = -\frac{1}{6}h^3(f(3f_tf_{yy} + 3f_{tty} + f(3f_{tyy} + 4f_yf_{yy} + ff_{yyy}) + 5f_yf_{yt} + f_y^3) + f_t(f_y^2 + 3f_{yt}) + f_{tt}f_y + f_{ttt}) + \frac{1}{2}h^2\left(f_tf_y + f_{tt} + f\left(f_y^2 + 2f_{yt} + ff_{yy}\right)\right) - h\left(f_t + ff_y\right) + f + O(h^4), \tag{4.19c}$$

$$(k_2)_{n-1} = \frac{1}{6}h(f_y(h(-2(3(a_{21}-2)a_{21}+2)f^2hf_{yy} + (3a_{21}-1)h(f_{tt} + 2ff_{yt}) + (3-6a_{21})f_t) + 6(a_{21}-1)f) + (a_{21}-1)fh((a_{21}-1)^2f^2hf_{yyy} + 3f_{yy}((1-2a_{21})hf_t + (a_{21}-1)f)) + (3a_{21}-1)fh^2f_y^3 + hf_y^2 + ((3a_{21}-1)hf_t - 6a_{21}f + 3f)) + f + O(h^4), \tag{4.19d}$$

$$(k_1)_{n-2} = 2h^2\left(f^2f_{yy} + f_tf_y + f_{tt} + f\left(f_y^2 + 2f_{yt}\right)\right) - \frac{4}{3}h^3(f^3f_{yyy} + f^2(3f_{tyy} + 4f_yf_{yy}) + f(3(f_tf_{yy} + f_{tty}) + 5f_yf_{yt} + f_y^3) + f_t(f_y^2 + 3f_{yt}) + f_{tt}f_y + f_{ttt}) - 2h(f_t + ff_y) + f + O(h^4), \tag{4.19e}$$

$$(k_2)_{n-2} = \frac{1}{2}h^2((a_{21}-2)^2f^2f_{yy} - 4(a_{21}-1)f_y(f_t + ff_y) - 2(a_{21}-2)ff_{yt} + f_{tt}) + \frac{1}{6}h^3((a_{21}-2)^3f^3f_{yyy} - 3(a_{21}-2)^2f^2f_{tyy} + 4(3a_{21}-2)f_y(f_tf_y + f_{tt} + f(f_y^2 + 2f_{yt} + ff_{yy})) + 12(a_{21}-1)f_{yt}(f_t + ff_y) - 12(a_{21}-2)(a_{21}-1)ff_{yy}(f_t + ff_y) + 3(a_{21}-2)ff_{tty} - f_{ttt}) + h((a_{21}-2)ff_y - f_t) + f + O(h^4). \tag{4.19f}$$

One substitutes Eqs. (4.19) into (4.14c) and requires that the entire expression on the l.h.s. of (4.15) be $O(h^5)$. One obtains a linear system for the coefficients whose

solution corresponds to the tableau

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{37}{24} & \frac{3}{8} \\ & \frac{-7}{12} & \frac{-3}{4} \\ & \frac{1}{24} & \frac{3}{8} \end{array} \tag{4.20}$$

We refer to method (4.20) as the pRK4.

4.5 Numerical stability of pRK methods

In this section, we discuss the numerical stability of the pRK3, pRK4, NpRK3, and NpRK4. The stability analysis follows that of Sec. 1.5.

4.5.1 Stability of the pRK3

To analyze the stability of the pRK3, we first apply the model problem $y' = \lambda y$ to Eqs. (4.5). After doing this, one obtains:

$$Y_{n+1} - \left(1 + \frac{3}{2}h\lambda + \frac{5}{12}(h\lambda)^2\right) Y_n + \left(\frac{1}{2}h\lambda + \frac{5}{12}(h\lambda)^2\right) Y_{n-1} = 0. \tag{4.21}$$

One then makes the substitution $Y_n = \rho^n Y_0$ to obtain the polynomial equation

$$\rho^2 - \left(1 + \frac{3}{2}h\lambda + \frac{5}{12}(h\lambda)^2\right) \rho + \left(\frac{1}{2}h\lambda + \frac{5}{12}(h\lambda)^2\right) = 0. \tag{4.22}$$

Eq. (4.22) has the solution

$$\rho_{1,2} = \frac{1}{24}\left(5(h\lambda)^2 + 18h\lambda + 12 \pm \sqrt{25(h\lambda)^4 + 180(h\lambda)^3 + 204(h\lambda)^2 + 144h\lambda + 144}\right). \tag{4.23}$$

Therefore, the pRK3 is stable when $|\rho_1| \leq 1$ and $|\rho_2| \leq 1$. The corresponding stability region is shown in Fig. 4.1, along with the stability region of the cRK3.

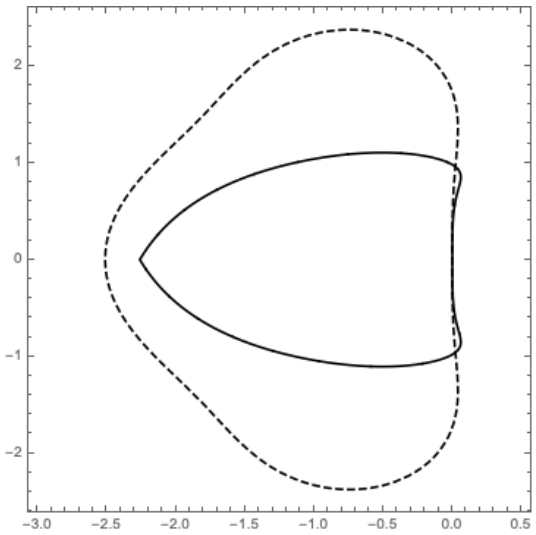

Figure 4.1: Stability region of the pRK3 (solid) and cRK3 (dashed) in the $Re(h\lambda) - Im(h\lambda)$ plane. Although both are third-order methods, the cRK3 has a significantly larger stability region, as well as a higher degree of tangency to the imaginary axis. Therefore, if one were integrating ODEs, the cRK3 would be the preferred method, as one could use a larger stepsize and would have better preservation of conserved quantities.

Although the cRK3 may be a better choice for solving ODEs than the pRK3, as we have seen in Chapter 3, the MoC becomes unstable when the cRK3 is used. We will see in Chapter 5 that this is *not* the case for the pRK3.

4.5.2 Stability of the pRK4

To perform the stability analysis of the pRK4 (4.20), we follow the procedure outlined above to obtain the stability polynomial

$$\phi(h\lambda) = \rho^3 - \left(1 + \frac{23}{12}h\lambda + \frac{3}{8}(h\lambda)^2\right)\rho^2 + \left(\frac{4}{3}h\lambda + \frac{3}{4}(h\lambda)^2\right)\rho - \left(\frac{5}{12}(h\lambda) + \frac{3}{8}(h\lambda)^2\right). \tag{4.24}$$

$\phi(h\lambda)$ has three roots, $\rho_{1,2,3}$, which we will not write but which can be found easily with, e.g., *Mathematica*. The corresponding stability region is shown in Fig. 4.2.

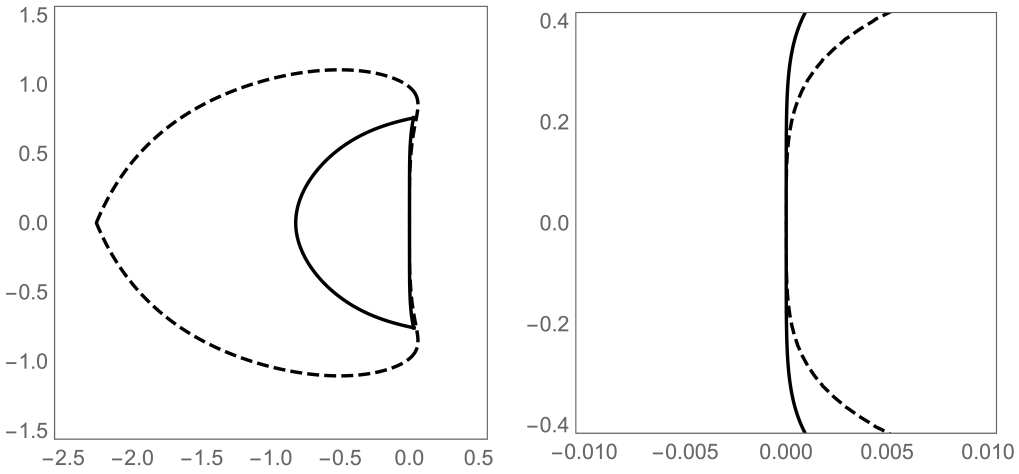

Figure 4.2: Stability region of the pRK4 (solid) and pRK3 (dashed). In the left pane, one can see that the area of the region for the pRK4 is significantly smaller than that of the pRK3. This is a common feature of multistep methods, including the well-known Adams methods. The right pane shows, however, that the pRK4 has a higher degree of tangency to the imaginary axis than the pRK3. Therefore, for non-stiff ODEs, the pRK4 is expected to preserve conserved quantities better than the pRK3.

4.5.3 Stability of the NpRK3

Recall that in the NpRK3, one has the free parameter a_{20} (see Eqs. (4.8) and (4.9)). In principle, one has carte blanche over the choice of a_{20} without running the risk of compromising the order of the method. In practice, however, one must select a_{20} wisely in order to optimize the stability properties of the method, which for our purposes is the degree of tangency the stability region has to the imaginary axis.

The stability polynomial for the NpRK3 is given by:

$$\phi(h\lambda) = \rho^2 - \left(1 + \frac{h\lambda}{6}(4 - 5a_{20}) + \frac{5(h\lambda)^2}{12}(2 + a_{20})\right)\rho - \left(\frac{h\lambda}{6}(2 + 5a_{20}) + \frac{5a_{20}}{12}(h\lambda)^2\right). \tag{4.25}$$

In Fig. 4.3, we plot the stability region corresponding to the NpRK3 with $a_{20} = 1.4$ along with the pRK3. This choice of a_{20} will be explained soon in the text.

Let us now elaborate on the statement made at the beginning of this subsection, as well as the reasoning behind our choice of $a_{20} = 1.4$ in the analysis above. The stability region of the NpRK3 changes with the choice of free parameter a_{20}. This is shown to be the case in Fig. 4.4, where we plot the stability region of the NpRK3 with three different values of a_{20}. The main point of doing this is to emphasize the fact that a_{20} will affect the preservation of conserved quantities in the NpRK3. This will be explored more in Appendix D.

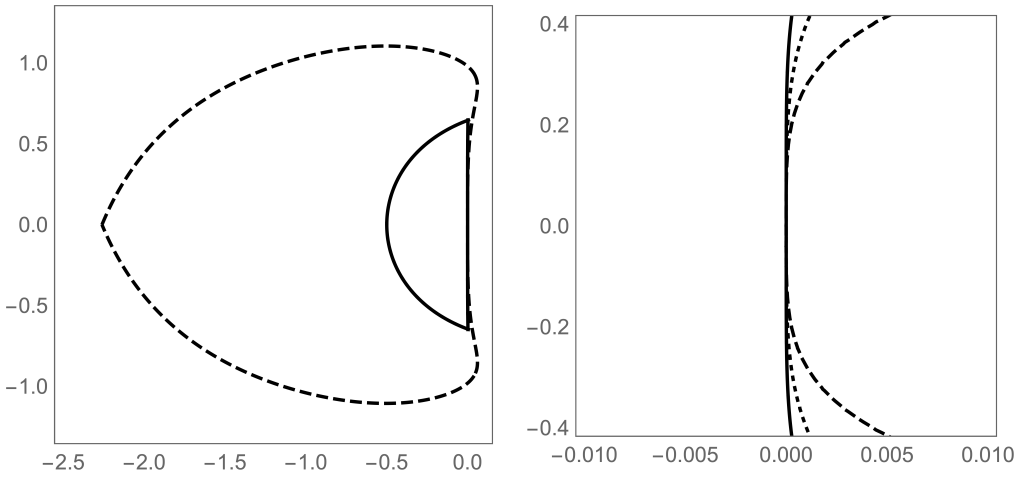

Figure 4.3: Stability regions of the NpRK3 (solid), pRK3 (dashed), and cRK3 (dotted, right pane only). The NpRK3 has a much smaller region than the pRK3, as shown in the left pane. In the right pane, we see that the NpRK3 has a higher degree of tangency to the imaginary axis than both the pRK3 and cRK3. Therefore, the NpRK3 is a good choice for a third-order method if one can afford a small stepsize and one is hoping to preserve conserved quantities.

4.5.4 Stability of the NpRK4

In the NpRK4, one has the free parameter Λ or c_{22}. Our analysis here will focus on the latter case; the details of the former case are similar.

In the case where c_{22} is free, one obtains the stability polynomial:

$$\phi(h\lambda) = \rho^3 - \left(1 - \frac{17h\lambda}{24} - c_{22}h\lambda + \frac{3(h\lambda)^2}{2}\right)\rho^2 + \left(4c_{22}h\lambda - \frac{5h\lambda}{3} - \frac{3(h\lambda)^2}{4} - 4c_{22}(h\lambda)^2\right)\rho - \left(\frac{h\lambda}{24} + 5c_{22}h\lambda + 2c_{22}(h\lambda)^2\right). \tag{4.26}$$

The choice of c_{22} has a similar effect on stability that the choice of a_{20} had in the NpRK3. This will be explored in Appendix D. In Fig. 4.5, we plot the corresponding

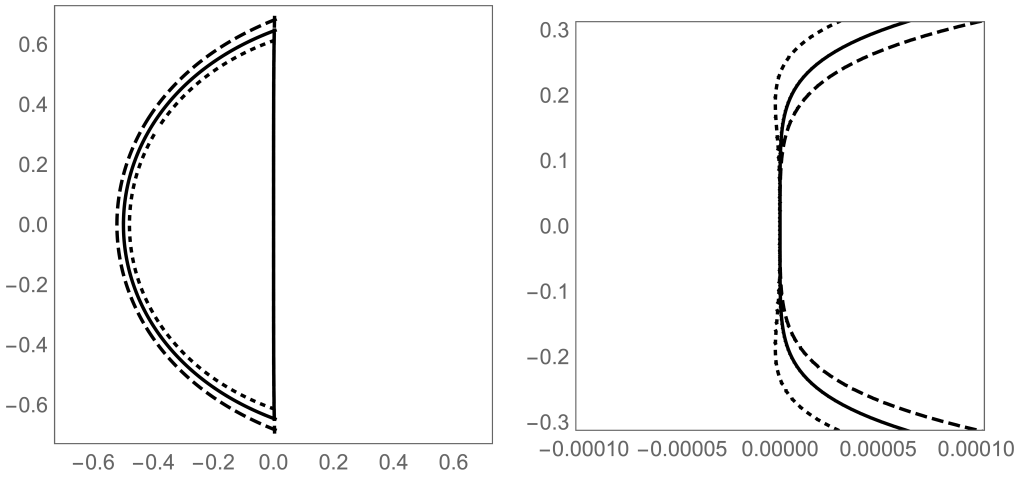

Figure 4.4: Stability regions of the NpRK3 with $a_{20} = 1.3$ (dashed), $a_{20} = 1.4$ (solid), and $a_{20} = 1.5$ (dotted). The right pane shows the plots very close to the imaginary axis (notice the horizontal and vertical scales). The meaning of these plots is the following: for larger values of $h\lambda$, one may want to choose $a_{20} = 1.3$, due to the slightly larger region of stability shown in the left pane. On the other hand, for small values of $h\lambda$, one should choose $a_{20} = 1.4$ in order to receive a higher degree of tangency to the imaginary axis.

stability region for $c_{22} = -0.15$.

4.5.5 Comparison of pRK methods with Adams methods

In this subsection, we compare the stability regions of the pRK methods with a well-known family of multistep methods, the Adams–Bashforth (AB) methods. The reason for doing this is to explain why, in certain situations, pRK methods may be a better choice than the well-known AB methods.

The third-order AB method (AB3) is given by:

$$Y_{n+1} = Y_n + \frac{h}{12}\Big(23f(Y_n, t_n) - 16f(Y_{n-1}, t_{n-1}) + 5f(Y_{n-2}, t_{n-2})\Big). \tag{4.27}$$

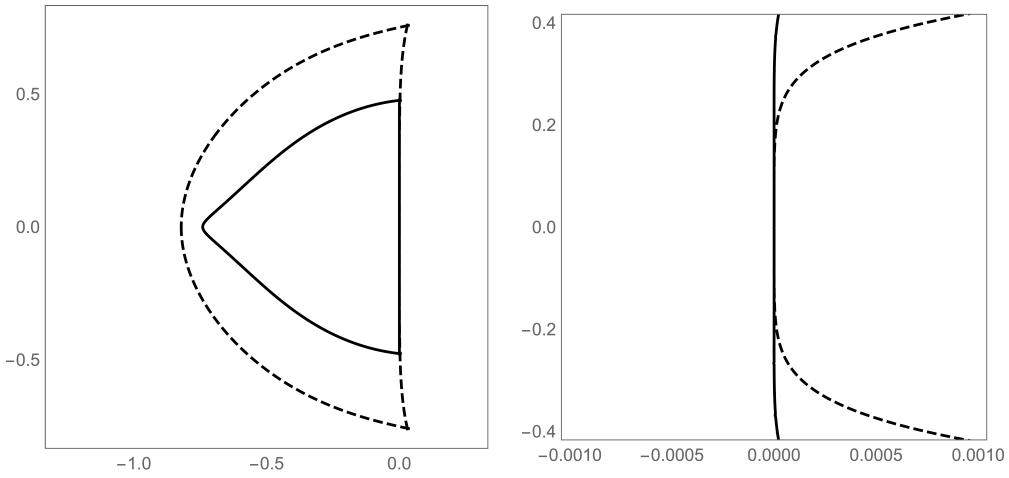

Figure 4.5: The left pane shows the stability regions of the NpRK4 with $c_{22} = -0.15$ (solid) and the pRK4 (dashed). This plot follows the behavior shown in Fig. 4.3: stability regions of NpRK methods can be smaller than those of pRK methods. The right pane shows the NpRK4 (solid) and pRK4 (dashed) close to the imaginary axis. This plot tells us that for small values of $h\lambda$, the NpRK4 should preserve conserved quantities to a degree better than the pRK4.

In Fig. 4.6, we compare the stability regions of the pRK3 and AB3.

The fourth-order AB method (AB4) is given by:

$$Y_{n+1} = Y_n + \frac{h}{24}\left(55f(Y_n, t_n) - 59f(Y_{n-1}, t_{n-1}) + 37f(Y_{n-2}, t_{n-2}) - 9f(Y_{n-3}, t_{n-2})\right). \tag{4.28}$$

In Fig. 4.7, we compare the stability regions of the pRK4 and AB4.

Since we use the MoC to solve hyperbolic equations which admit conserved quantities, we are interested in methods which will do the best job of preserving these quantities. According to Figs. 4.6 and 4.7, the pRK methods can accomplish this task better than the AB methods, and thus we will choose to employ pRK methods in our upcoming MoC schemes.

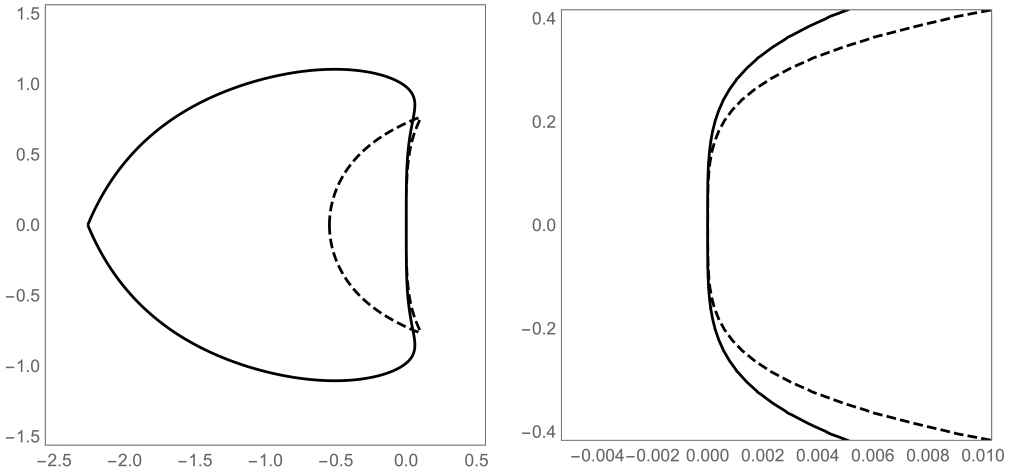

Figure 4.6: Stability regions of the pRK3 (solid) and AB3 (dashed). The left pane shows that the region for the pRK3 is much larger, and the right pane shows that the pRK3 also has a slightly higher degree of tangency to the imaginary axis than the AB3. Thus, the pRK3 (and therefore, NpRK3) should preserve conserved quantities better than the AB3.

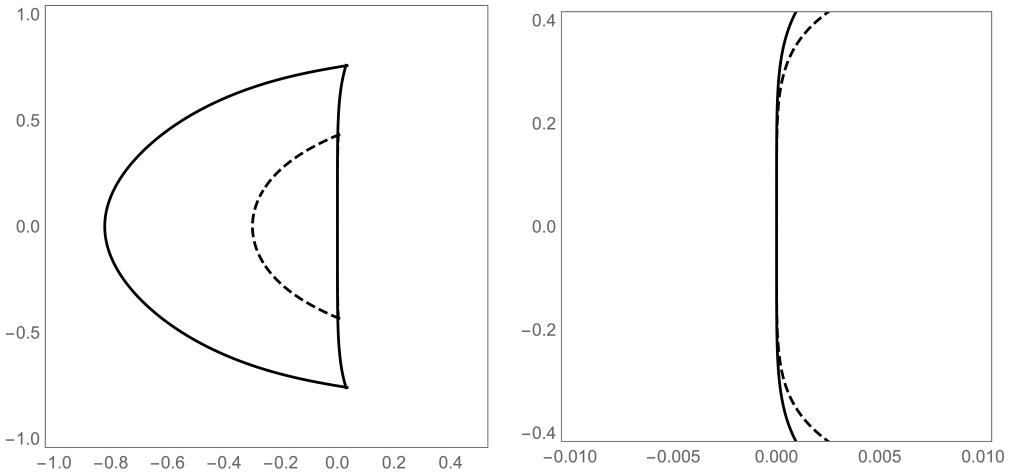

Figure 4.7: Stability regions of the pRK4 (solid) and AB4 (dashed). We draw similar conclusions from these plots as we did for the plots in Fig. 4.6.

Another reason we prefer pRK methods over AB methods is because the number of previous solutions needed in AB methods is greater than the number needed in

pRK methods. For example, the pRK3 requires Y_n and Y_{n-1}, while the AB3 requires these solutions in addition to Y_{n-2}. When solving an ODE, this is not a huge problem. However, the implementation of MoC schemes becomes increasingly complicated as the number previous solutions needed increases. This will be seen in Chapters 5 and 6 when we discuss the implementation of MoC-pRK schemes.

4.6 Numerical results

In this section, we present numerical results obtained from using the aforementioned methods to solve ODEs. These results confirm what we have seen in the stability plots of Sec. 4.5 and predict behavior that we will see later when using these methods in the MoC.

The two subsections focus on two slightly different aspects of the preservation of conserved quantities by a numerical method. In Sec. 4.6.1 we will mainly focus on how well the methods preserve the conserved quantities for a given h, while in Sec. 4.6.2 we will focus on how this preservation property scales with h.

4.6.1 Kepler two-body problem

The first problem we consider is the Kepler two-body problem:

$$q'' = -\frac{q}{(q^2+r^2)^{3/2}}, \quad r'' = -\frac{r}{(q^2+r^2)^{3/2}}, \tag{4.29}$$

where q and r are the Cartesian coordinates of a particular radius vector relative to the center of mass of two particles in each other's gravitational field. This problem has constants of motion which include the Hamiltonian, H, and angular momentum,

A. The equations for these constants are given by:

$$H = \frac{1}{2}(Q^2 + R^2) - \frac{1}{\sqrt{q^2 + r^2}}, \tag{4.30}$$

$$A = qR - rQ, \tag{4.31}$$

where Q and R are the velocities corresponding to q and r, respectively. The results of integrating (4.29) with the initial condition corresponding to an elliptical orbit with eccentricity $\epsilon = 0.6$ using the cRK3, pRK3, NpRK3 ($a_{20} = 1.4$), cRK4, pRK4, and NpRK4 ($c_{22} = -0.15$) are shown in Figs. 4.8 and 4.9. It should be noted that while the values of the numerical error shown in Figs. 4.8 and 4.9 are specific to the particular value of the orbits eccentricity, they illustrate the general trend observed among the methods used.

A discussion about the dependence of the error on the free parameters of the NpRK methods is found in Appendix D.

4.6.2 Anharmonic oscillator

The second problem we consider are the equations of an anharmonic oscillator:

$$y_1' = y_2, \tag{4.32a}$$

$$y_2' = -y_1(y_1 - 1)(y_1 + 2). \tag{4.32b}$$

To make (4.30) into an IVP, we impose the initial condition

$$y_1(0) = 0, \quad y_2(0) = 1. \tag{4.32c}$$

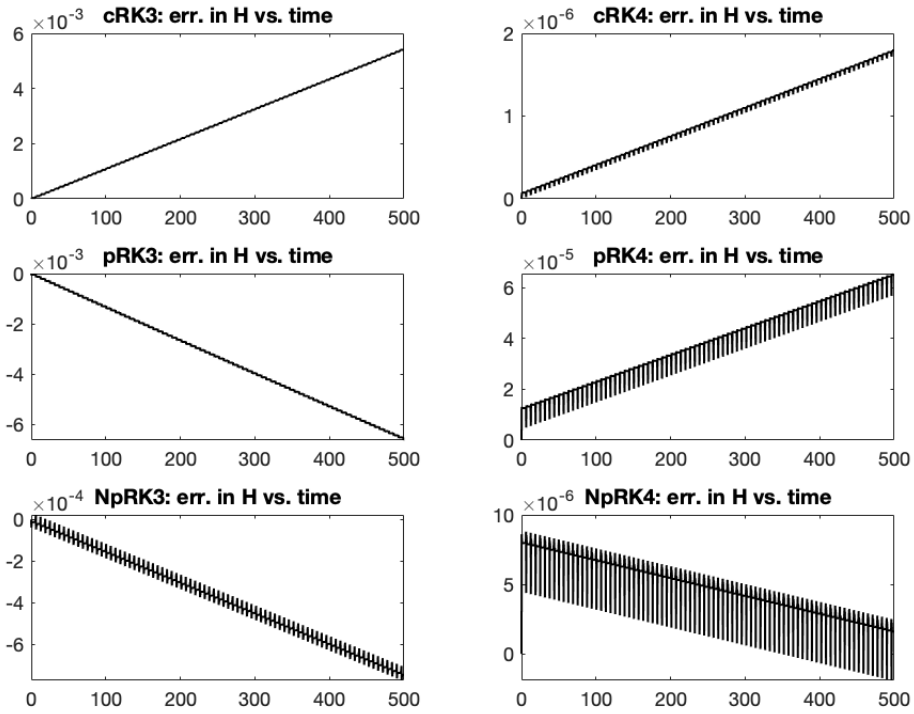

Figure 4.8: Error in the Hamiltonian vs. time produced from integrating the Kepler problem up to $t = 500$ with stepsize $h = 0.01$. In Sec. 4.5, we found that among the methods mentioned there, the method with the worst degree of tangency to the imaginary axis for small λh was the pRK3, and the best was the NpRK4, with the NpRK3 and pRK4 in the middle. This is exactly what we see in this plot: the pRK3 does the worst job of conserving the Hamiltonian and the NpRK4 does the best job (comparable to the cRK4).

We will solve IVP (4.32) with the cRK3, pRK3, cRK4, and pRK4 with different values of h and record the change in the Hamiltonian,

$$H = \frac{1}{2}y_2^2 - y_1^2 + \frac{1}{3}y_1^3 + \frac{1}{4}y_1^4. \tag{4.33}$$

The results are shown in Table 1. Note that we omit the results obtained by the NpRK methods, as our goal here is to show the degree in which the conserved quantities are

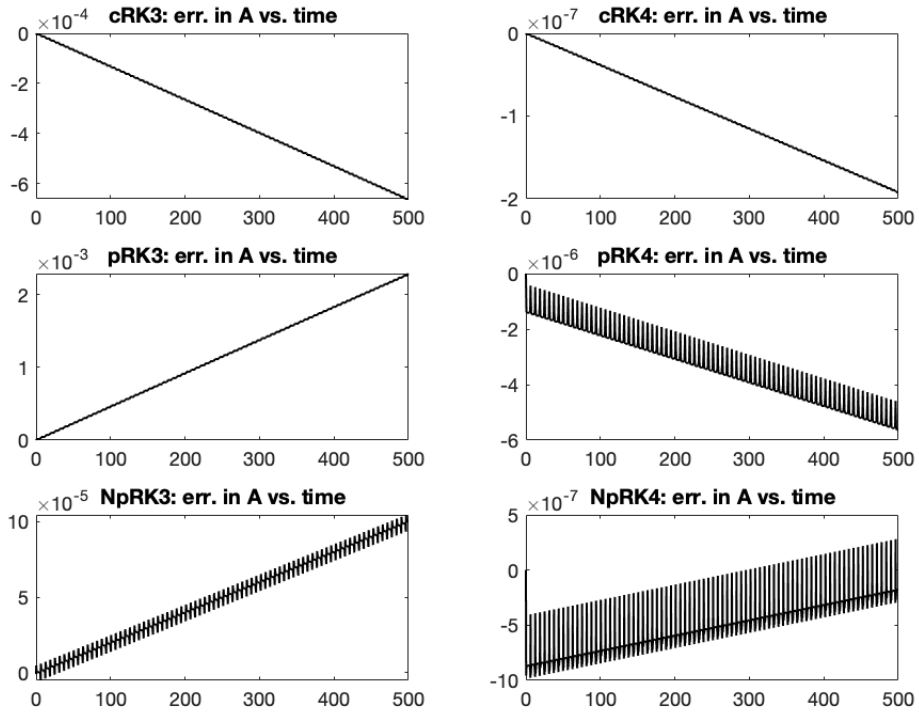

Figure 4.9: Error in angular momentum vs. time produced from integrating the Kepler problem, as for Fig. 4.8. Similarly to what was written in the caption of Fig. 4.6, these results verify what we observed in the stability plots in Sec. 4.5.

preserved by third- and fourth-order methods.

These results allow us to estimate the order in which the Hamiltonian is preserved. Namely, they tell us that the cRK3 and pRK3 preserve the Hamiltonian with an order of 3, and the cRK4 and pRK4 do so with an order of 5. This is because when we halve the stepsize, we obtain ratios in the change in the Hamiltonian of approximately 2^3 and 2^5. We will see this type of behavior later on when we use pRK schemes in the MoC.

	cRK3	pRK3	cRK4	pRK4
ΔH for $h = 0.01$	5.9×10^{-3}	2.7×10^{-2}	1.2×10^{-6}	4.2×10^{-5}
ΔH for $h = 0.005$	7.4×10^{-4}	3.4×10^{-3}	3.8×10^{-8}	1.3×10^{-6}
Ratio	8.0	7.9	31.6	32.3

Table 4.1: Results of solving IVP (4.32) up to $t = 1000$ with four methods. Recorded is the change in the Hamiltonian, ΔH, for a given method and stepsize. The fourth row gives the ratio of the numbers in the second and third rows.

4.7 Concluding remarks

In this Chapter, we have presented four ODE solvers based on the pRK framework for use in the MoC: the third-order pRK3 and NpRK3, and the fourth-order pRK4 and NpRK4. These methods all share the property that $b_2 = 1$ in the second stage derivative, and can therefore be implemented with ease on our MoC grid without employing any virtual nodes. We have also investigated the stability of these methods for ODEs and seen to which order we can expect them to preserve the conserved quantities that will be of interest to us later on.

The remaining chapters of this thesis will focus on the use of pRK schemes as the ODE solver in the MoC. We will apply the von Neumann stability analysis to MoC schemes which use third- and fourth-order pRK methods, then verify the analysis numerically. We will see that the resulting third- and fourth-order MoC schemes do not have a large instability, as they did when they were based on explicit RK methods.

5 The MoC using third-order pRK schemes

In Chapter 4, we presented pRK schemes for ODEs, including the third-order pRK3 and NpRK3. In this chapter, we will develop a third-order MoC scheme which uses the pRK3 as the ODE solver. We will show that the resulting method, the MoC-pRK3, has stability properties similar to the MoC-ME. We will also discuss for the first time in this thesis the implications of using nonreflecting boundary conditions in an MoC scheme, rather than the periodic boundary conditions which the von Neumann analysis and our numerical simulations have assumed thus far. This will be done in the context of the MoC-pRK3. Chapter 6 will follow a similar structure as Chapter 5 but with a focus on fourth-order pRK schemes.

5.1 Formulation of the MoC-pRK3

Recall the pRK3 scheme for ODEs from Chapter 4:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{13}{12} & \frac{5}{12} \\ & \frac{-1}{12} & \frac{-5}{12} \end{array} \tag{4.4}$$

For the ODE $y' = f(Y_n, t_n)$, tableau (4.4) has the following meaning:

$$(k_1)_n = f(Y_n, t_n), \tag{4.5a}$$

$$(k_2)_n = f(Y_n + h(k_1)_n, t_n + h), \tag{4.5b}$$

$$Y_{n+1} = Y_n + \frac{h}{12}\left(13(k_1)_n + 5(k_2)_n - (k_1)_{n-1} - 5(k_2)_{n-1}\right). \tag{4.5c}$$

We now give the equations for the MoC-pRK3 which advance the solution from time level n to $n+1$. The stencil for the method is shown in Fig. 5.1.

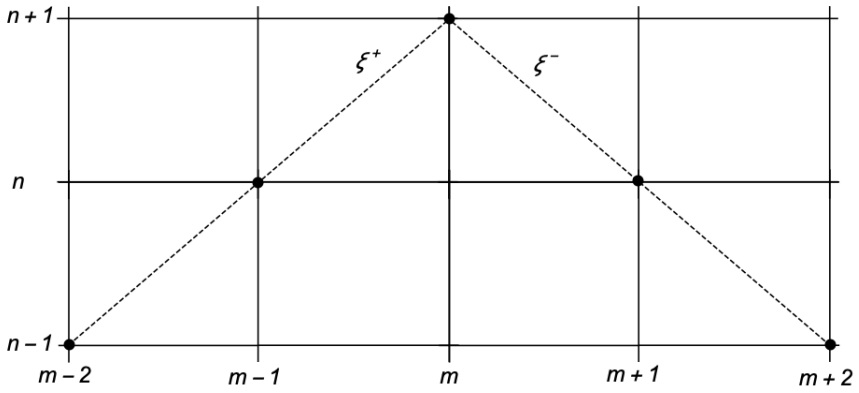

Figure 5.1: Stencil for the MoC-pRK3 for systems admitting two straight-line and crossing characteristics, ξ^+ and ξ^-. The solution Y^+ is integrated along ξ^+, which propagates from left to right, and the solution Y^- is integrated along ξ^-, which propagates from right to left. Recall that we enforce a square grid ($\Delta t = \Delta x = h$), and thus in order to obtain either of $(Y^\pm)_m^{n+1}$, one needs information of both $(Y^\pm)_{m\mp1}^{n}$ and of both $(Y^\pm)_{m\mp2}^{n-1}$.

In the MoC-pRK3, the equations for k_1 are no different than they have been in previous methods:

$$(k_1^+)_{m-1}^n = f^+\left((Y^+)_{m-1}^n, (Y^-)_{m-1}^n\right), \tag{5.1a}$$

$$(k_1^-)_{m+1}^n = f^-\left((Y^+)_{m+1}^n, (Y^-)_{m+1}^n\right). \tag{5.1b}$$

When computing $(k_2)^n_{m\mp1}$, one needs the points $(Y^\pm)^{n+1}_m$ with local accuracy $O(h^3)$, just as in the MoC-cRK3. We accomplish this task with the MoC-ME:

$$\left(\overline{Y}^+\right)^{n+1}_m = (Y^+)^n_{m-1} + h(k_1^+)^n_{m-1}, \tag{5.1c}$$

$$\left(\overline{Y}^-\right)^{n+1}_m = (Y^-)^n_{m+1} + h(k_1^-)^n_{m+1}, \tag{5.1d}$$

$$\left(\ddot{Y}^+\right)^{n+1}_m = \frac{1}{2}\left[(Y^+)^n_{m-1} + \left(\overline{Y}^+\right)^{n+1}_m + hf^+\left(\left(\overline{Y}^+\right)^{n+1}_m, \left(\overline{Y}^-\right)^{n+1}_m\right)\right], \tag{5.1e}$$

$$\left(\ddot{Y}^-\right)^{n+1}_m = \frac{1}{2}\left[(Y^-)^n_{m+1} + \left(\overline{Y}^-\right)^{n+1}_m + hf^-\left(\left(\overline{Y}^+\right)^{n+1}_m, \left(\overline{Y}^-\right)^{n+1}_m\right)\right]. \tag{5.1f}$$

k_2 is then computed by:

$$(k_2^+)^n_{m-1} = f^+\left(\left(\overline{Y}^+\right)^{n+1}_m, \left(\ddot{Y}^-\right)^{n+1}_m\right), \tag{5.1g}$$

$$(k_2^-)^n_{m+1} = f^-\left(\left(\ddot{Y}^+\right)^{n+1}_m, \left(\overline{Y}^-\right)^{n+1}_m\right). \tag{5.1h}$$

The solution at the next level is then:

$$(Y^+)^{n+1}_m = (Y^+)^n_{m-1} + \frac{h}{12}\left(13(k_1^+)^n_{m-1} + 5(k_2^+)^n_{m-1} - (k_1^+)^{n-1}_{m-2} - 5(k_2^+)^{n-1}_{m-2}\right), \tag{5.1i}$$

$$(Y^-)^{n+1}_m = (Y^-)^n_{m+1} + \frac{h}{12}\left(13(k_1^-)^n_{m+1} + 5(k_2^-)^n_{m+1} - (k_1^-)^{n-1}_{m+2} - 5(k_2^-)^{n-1}_{m+2}\right). \tag{5.1j}$$

Note that $(k^\pm_{1,2})^{n-1}_{m\mp2}$ were computed in the previous step by the same procedure.

Equations (5.1) form the MoC-pRK3. This scheme is more straightforward than our other third-order scheme, the MoC-cRK3, since we do not need to compute any points that are not on the stencil.

5.2 von Neumann stability analysis of the MoC-pRK3

With the MoC-pRK3 formulated, we would like to know if the method is stable. We will answer this question by performing the von Neumann stability analysis on the background of system (1.50), which we have seen in Chapters 2 and 3. Recall the goal of the von Neumann analysis is to write the numerical method in the form

$$(\mathbf{s})_m^{n+1} = \mathbf{\Phi}(z)(\mathbf{s})_m^n, \quad z = kh, \tag{5.2}$$

where $\mathbf{s}$ is a vector of small perturbations from the true solution and $\mathbf{\Phi}$ is a matrix.

Previously, $\mathbf{s}$ had the form $(\mathbf{s})_m^n \equiv ((\underline{\mathbf{s}}^+)_m^n, (\underline{\mathbf{s}}^-)_m^n)^T$. Now, in the context of the MoC-pRK3, the solution at level $n+1$ depends on the solutions at level n **and** at level $n-1$. To reflect this in the von Neumann analysis, we seek an equation of the form:

$$\left((\mathbf{s})_m^{n+1}, (\mathbf{s})_m^n\right)^T = \mathbf{\Phi}(z)\left((\mathbf{s})_m^n, (\mathbf{s})_m^{n-1}\right)^T. \tag{5.3}$$

Let us commence with the analysis. Conceptually, the analysis is identical to that which we performed in the previous chapters. However, the vector on the l.h.s. of (5.3) is now of length 8, rather than 4. Thus, most of the matrices we encounter in this analysis will be of size 4×8.

Equations (5.1{a,b}) applied to (2.11), the linearized version of (1.50), yield:

$$\begin{pmatrix} \underline{\mathbf{k}}_1^+ \\ \underline{\mathbf{k}}_1^- \end{pmatrix}_m^n = [\mathbf{P}, \mathcal{O}]\hat{\mathbf{s}} \equiv \mathcal{K}_{11}\ \hat{\mathbf{s}}, \tag{5.4a}$$

$$\begin{pmatrix} \underline{\mathbf{k}}_1^+ \\ \underline{\mathbf{k}}_1^- \end{pmatrix}_m^{n-1} = [\mathcal{O}, \mathbf{P}]\hat{\mathbf{s}} \equiv \mathcal{K}_{12}\ \hat{\mathbf{s}}, \tag{5.4b}$$

where $\mathbf{P}$ is defined in (2.12), $\mathcal{O}$ is the 4×4 zero matrix, and $\hat{\mathbf{s}}$ is the 8×1 vector

$$\hat{\mathbf{s}} = ((\mathbf{s})_m^n, (\mathbf{s})_m^{n-1})^T.$$

Next, Eqs. (5.1{c,d}) applied to (2.11), along with the substitution (2.16), give us:

$$\begin{pmatrix} \overline{\underline{\mathbf{s}}}^+ \\ \overline{\underline{\mathbf{s}}}^- \end{pmatrix}_m^{n+1} = \Big[\mathbf{Q}\Big([\mathbf{I}, \mathcal{O}] + h\mathcal{K}_{11}\Big)\Big]\ \hat{\mathbf{s}} \equiv \mathcal{S}_{11}\ \hat{\mathbf{s}}, \tag{5.4c}$$

$$\begin{pmatrix} \overline{\underline{\mathbf{s}}}^+ \\ \overline{\underline{\mathbf{s}}}^- \end{pmatrix}_m^{n} = \Big[\mathbf{Q}\Big([\mathcal{O}, \mathbf{I}] + h\mathcal{K}_{12}\Big)\Big]\ \hat{\mathbf{s}} \equiv \mathcal{S}_{12}\ \hat{\mathbf{s}}, \tag{5.4d}$$

where $\mathbf{Q}$ is defined in (2.21) and $\mathbf{I}$ is the 4×4 identity matrix. (5.4{c,d}) share a similar structure with (2.22), the matrix we obtained for the MoC-SE.

Applying (5.1{e,f}) to (2.11) gives us an analogue of the MoC-ME stability matrix:

$$\begin{pmatrix} \ddot{\underline{\mathbf{s}}}^{+} \\ \ddot{\underline{\mathbf{s}}}^{-} \end{pmatrix}_{m}^{n+1} = \Big[\frac{1}{2}\Big(\ [\mathbf{Q},\mathcal{O}] + \mathcal{S}_{11} + h\mathcal{K}_{11}\mathcal{M}_{11}\ \Big)\Big]\ \hat{\mathbf{s}} \equiv \mathcal{S}_{21}\ \hat{\mathbf{s}}, \tag{5.4e}$$

$$\begin{pmatrix} \ddot{\underline{\mathbf{s}}}^{+} \\ \ddot{\underline{\mathbf{s}}}^{-} \end{pmatrix}_{m}^{n} = \Big[\frac{1}{2}\Big(\ [\mathcal{O},\mathbf{Q}] + \mathcal{S}_{12} + h\mathcal{K}_{12}\mathcal{M}_{12}\ \Big)\Big]\ \hat{\mathbf{s}} \equiv \mathcal{S}_{22}\ \hat{\mathbf{s}}, \tag{5.4f}$$

where

$$\mathcal{M}_{11} = \begin{bmatrix} \mathcal{S}_{11} \\ \mathcal{O}_{4\times 8} \end{bmatrix}, \quad \mathcal{M}_{12} = \begin{bmatrix} \mathcal{O}_{4\times 8} \\ \mathcal{S}_{12} \end{bmatrix}. \tag{5.5}$$

After applying (5.1{g,h}) to (2.11), we obtain:

$$\begin{pmatrix} (\underline{\mathbf{k}}_2^{+})_{m-1}^{n} \\ (\underline{\mathbf{k}}_2^{-})_{m+1}^{n} \end{pmatrix} = \Big([\mathbf{P}_{\text{diag}},\mathcal{O}]\mathcal{M}_{11} + [\mathbf{P}_{\text{offdiag}},\mathcal{O}]\mathcal{M}_{21}\Big)\hat{\mathbf{s}} \equiv \mathcal{K}_{21}\ \hat{\mathbf{s}}, \tag{5.4g}$$

$$\begin{pmatrix} (\underline{\mathbf{k}}_2^{+})_{m-1}^{n-1} \\ (\underline{\mathbf{k}}_2^{-})_{m+1}^{n-1} \end{pmatrix} = \Big([\mathcal{O},\mathbf{P}_{\text{diag}}]\mathcal{M}_{12} + [\mathcal{O},\mathbf{P}_{\text{offdiag}}]\mathcal{M}_{22}\Big)\hat{\mathbf{s}} \equiv \mathcal{K}_{22}\ \hat{\mathbf{s}}, \tag{5.4h}$$

where

$$\mathcal{M}_{21} = \begin{bmatrix} \mathcal{S}_{21} \\ \mathcal{O}_{4\times 8} \end{bmatrix}, \quad \mathcal{M}_{22} = \begin{bmatrix} \mathcal{O}_{4\times 8} \\ \mathcal{S}_{22} \end{bmatrix}. \tag{5.6}$$

Equations (5.1{i,j}) applied to (2.11) yield:

$$\begin{pmatrix} \underline{\mathbf{s}}^{+} \\ \underline{\mathbf{s}}^{-} \end{pmatrix}_{m}^{n+1} = \Big([\mathbf{Q}, \mathcal{O}] + \frac{h}{12} \left(13\mathbf{Q}\mathcal{K}_{11} + 5\mathcal{K}_{21} - \mathbf{Q}^2\mathcal{K}_{12} - 5\mathbf{Q}\mathcal{K}_{22} \right) \Big) \hat{\mathbf{s}} \equiv \mathcal{N}_1 \, \hat{\mathbf{s}}. \quad (5.4\text{i})$$

Note that $\mathcal{K}_{11}$ and $\mathcal{K}_{12}$ are defined with the index m, as are the quantities in $\hat{\mathbf{s}}$. According to (5.1{i,j}) we need k_1 at the nodes $(m-1, n)$ and $(m-2, n-1)$. This explains why $\mathcal{K}_{11}$ and $\mathcal{K}_{12}$ are multiplied on the left by $\mathbf{Q}$ and $\mathbf{Q}^2$. A similar explanation applies to the presence of $\mathbf{Q}$ on the left of $\mathcal{K}_{22}$.

Equation (5.4i) gives us an expression for $(\mathbf{s})_m^{n+1}$ in the l.h.s. of (5.3) in terms of $\hat{\mathbf{s}}$. The expression for $(\mathbf{s})_m^n$ in terms of $\hat{\mathbf{s}}$ is simple:

$$(\mathbf{s})_m^n = [\mathbf{I}, \mathcal{O}] \left((\mathbf{s})_m^n, (\mathbf{s})_m^{n-1} \right)^T \equiv \mathcal{N}_2 \, \hat{\mathbf{s}}. \quad (5.4\text{j})$$

Therefore, the stability matrix $\mathbf{\Phi}$ in (5.3) is given by:

$$\mathbf{\Phi}(z) = \begin{bmatrix} \mathcal{N}_1 \\ \mathcal{N}_2 \end{bmatrix}. \quad (5.5)$$

In Fig. 5.2, we show the von Neumann stability plot corresponding to $\mathbf{\Phi}(z)$.

5.3 Numerical verification of the von Neumann analysis

Following the analysis of Appendix B, at time t, the maximum of the logarithm of the error in the Fourier spectrum for the MoC-pRK3 when solving system (1.50) will

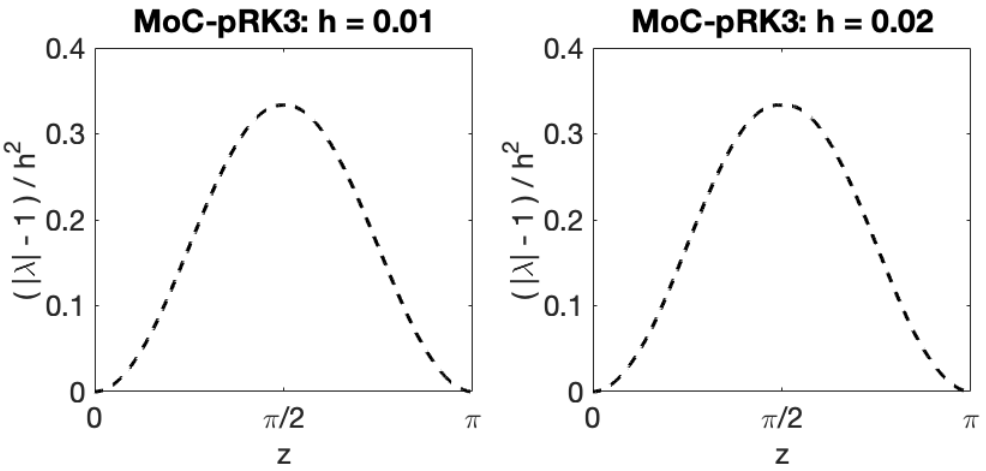

Figure 5.2: Largest eigenvalue of the stability matrix for the MoC-pRK3 for $h = 0.01$ *and* $h = 0.02$*. The plots are very similar for these values of* h*, so by an analysis similar to that of Appendix B, we have* $|\lambda_{\max}| \approx 1 + O(h^2)$*, which means that the growth rate in the error is* $O(h)$*. Similar to the MoC-ME, the largest instability occurs for harmonics near the middle of the spectrum. However, the maximum growth rate of harmonics with* $|k| \sim k_{\max}/2$ *is about three times smaller than that for the MoC-ME.*

be:

$$\mathrm{err}_{\mathrm{pRK3}} = \frac{cht}{\ln 10} + \log_{10} W_0, \tag{5.6}$$

where W_0 is the initial error in the Fourier spectrum and c satisfies

$$|\lambda_{\max}| = 1 + ch^2. \tag{5.7}$$

From Fig. 5.2, one can find that $c \approx 1/3$. Therefore, in our simulation with $\log_{10} W_0 \approx -7.8$, we should expect the logarithm of the maximum error to be approximately -7. This is shown to be the case in Fig. 5.3.

In the numerical simulation, at $t = 500$, the base-10 logarithm of the error has grown by approximately 0.8. Recall that in the MoC-ME, the growth constant c was

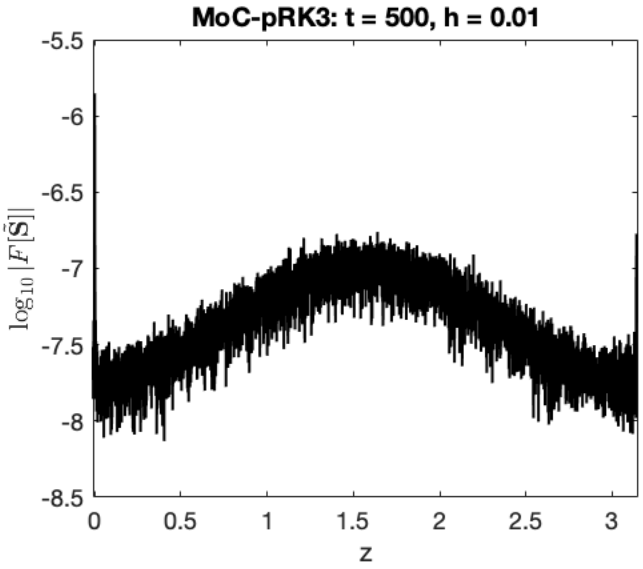

Figure 5.3: Logarithm Fourier spectrum of the error in the solution obtained by solving (1.50) with the MoC-pRK3 up to $t = 500$ with $h = 0.01$. The logarithm of the maximum error is close to -7, as predicted.

1. In the MoC-pRK3, $c = 1/3$. This means that the error in the MoC-ME should increase by around 3 times as much, in the logarithmic scale, the error in the MoC-pRK3. This is what we have observed: the growth in the logarithm of the error in the MoC-ME was 2.2, roughly 3 times larger than 0.8. From this, we see that although the instability in the MoC-ME and MoC-pRK3 occurs for similar harmonics and with a similar shape, it will destroy the solution in the MoC-ME sooner than it will in the MoC-pRK3.

5.4 Implementation of the MoC-pRK3 scheme with periodic boundary conditions

In the von Neumann analysis and simulation performed to verify it, we assume periodic boundary conditions (b.c.), meaning:

$$(Y^{\pm})^n_{-1} \equiv (Y^{\pm})^n_M, \quad (Y^{\pm})^n_{M+1} \equiv (Y^{\pm})^n_0, \tag{5.8}$$

where $M+1$ is the number of spatial grid points and $m = 0, 1, \ldots, M$. The same convention applies to all of the stage derivatives $k_i^{\pm}$ which we compute. In the MoC schemes discussed so far in this thesis, it is straightforward to implement periodic b.c. using (5.8). Implementing them in a multistep scheme, such as the MoC-pRK3, must be done with care. To see why this is so, let us examine the stencil for the MoC-pRK3 near the left boundary (Fig. 5.4).

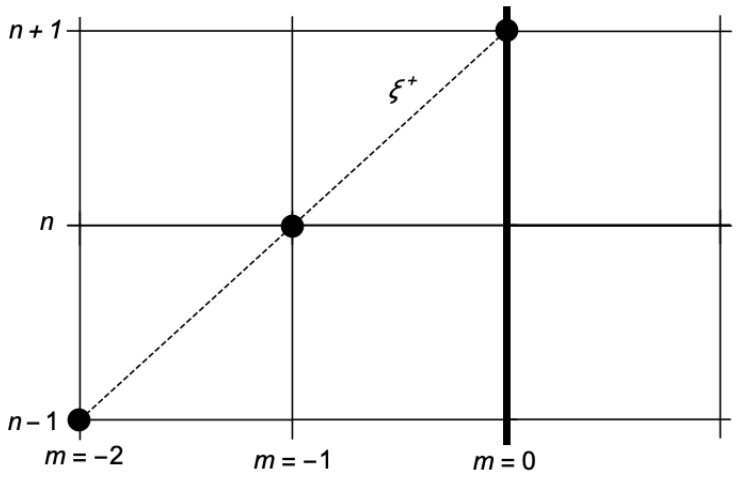

Figure 5.4: Stencil for the MoC-pRK3 near the left boundary (the thick line, $m = 0$).

By (5.1i), the solution $(Y^+)_0^{n+1}$ is given by:

$$(Y^+)_0^{n+1} = (Y^+)_{-1}^n + \frac{h}{12}\left(13(k_1^+)_{-1}^n + 5(k_2^+)_{-1}^n - (k_1^+)_{-2}^{n-1} - 5(k_2^+)_{-2}^{n-1}\right). \tag{5.9}$$

The values with the index $(m = -1, n)$ can be found as prescribed in (5.8). To maintain periodic b.c., the values with index $(m = -2, n-1)$ can be found by:

$$(Y^\pm)_{-2}^{n-1} \equiv (Y^\pm)_{M-1}^{n-1}. \tag{5.10a}$$

Similarly, one needs values at the node $(m = M+2, n-1)$ when computing $(Y^-)_M^{n+1}$. These values can be found by:

$$(Y^\pm)_{M+2}^{n-1} \equiv (Y^\pm)_1^{n-1}. \tag{5.10b}$$

Code implementing the MoC-pRK3 with periodic b.c. will be given in Appendix F.

5.5 The MoC-NpRK3 scheme and stability

The NpRK3 scheme (4.8) with $b_2 = 1$ for the ODE $y' = f(Y,t)$ is given by:

$$(k_1)_n = f(Y_n, t_n), \tag{5.11a}$$

$$(k_2)_n = f\left(Y_n + \Lambda(Y_n - Y_{n-1}) + h(a_{20}(k_1)_{n-1} + a_{21}(k_1)_n), t_n + h\right), \tag{5.11b}$$

$$Y_{n+1} = Y_n + h\left(c_{11}(k_1)_n + c_{12}(k_2)_n + c_{21}(k_1)_{n-1}\right), \tag{5.11c}$$

where all of the coefficients are completely determined by the free parameter a_{20} via (4.9). The MoC-NpRK3 has the same stencil as the MoC-pRK3 (Fig. 5.1), and the

equations for the scheme are given by:

$$(\widetilde{Y}^{\pm})_m^n = (Y^{\pm})_m^n + \Lambda\left((Y^{\pm})_m^n - (Y^{\pm})_{m\mp1}^{n-1}\right) + h\left(a_{20}(k_1^{\pm})_{m\mp1}^{n-1} + a_{21}(k_1^{\pm})_m^n\right), \quad (5.12a)$$

$$(k_2^{+})_{m-1}^n = f^{+}\left((\widetilde{Y}^{+})_{m-1}^n, (\ddot{Y}^{-})_m^{n+1}\right), \quad (5.12b)$$

$$(k_2^{-})_{m+1}^n = f^{-}\left((\ddot{Y}^{+})_m^n, (\widetilde{Y}^{-})_{m+1}^n\right), \quad (5.12c)$$

$$(Y^{\pm})_m^{n+1} = (Y^{\pm})_{m\mp1}^n + h\left(c_{11}(k_1^{\pm})_{m\mp1}^n + c_{12}(k_2^{\pm})_{m\mp1}^n + c_{21}(k_1^{\pm})_{m\mp2}^{n-1}\right), \quad (5.12d)$$

where $k_1^{\pm}$ and $\ddot{Y}^{\pm}$ are computed the same way as they are in (5.1{a,b,e,f}).

We will omit the technical details of the von Neumann analysis of the MoC-NpRK3, since they are very similar to those given in Sec. 5.2. We will, however, present the results of it and the numerical verification (see Fig. 5.5). In the following analysis, we use $a_{20} = 7/5 = 1.4$, the value which optimizes the stability region for the ODE. Although we saw in Appendix D that the optimal choice of a_{20} in the PDE case is problem dependent, the von Neumann stability plots for different a_{20} are virtually indistinguishable.

For the remainder of this thesis, our examination of third-order MoC schemes will focus on the MoC-pRK3. Although the growth rate of the instability in the MoC-NpRK3 is smaller than that in the MoC-pRK3, both growth rates are small enough that either method could be used for a realistic integration time.

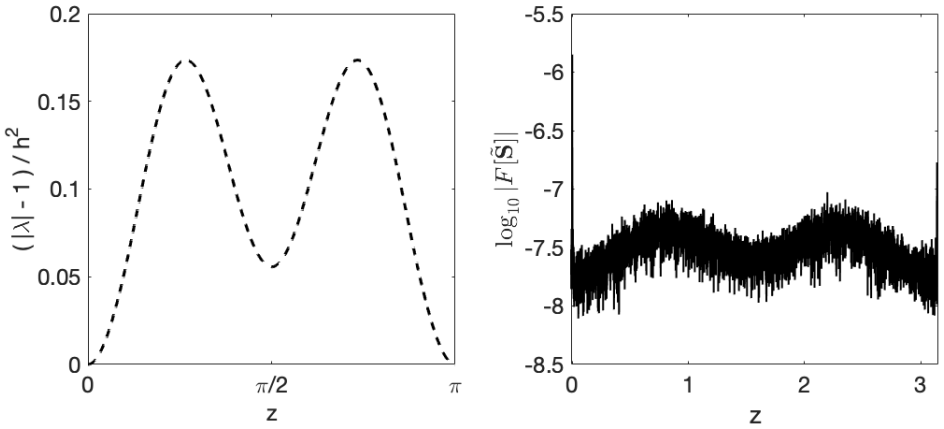

Figure 5.5: von Neumann stability plot of the NpRK3 (left) and numerical verification (right), both with $h = 0.01$. Like the MoC-ME and MoC-pRK3, the growth rate of the instability is $O(h)$ in this method, with the growth constant $c \approx 0.17$, i.e. twice as small as for the MoC-pRK3. Unlike the MoC-ME and MoC-pRK3, the largest instability for the MoC-NpRK3 occurs for harmonics near $z = \pi/4$ and $z = 3\pi/4$, rather than $z = \pi/2$. The figure on the right was created by integrating system (1.50) up to $t = 500$ with the MoC-NpRK3. We see that the two "peaks" predicted by the stability plot have begun to form.

5.6 MoC schemes using nonreflecting boundary conditions

So far in this thesis, when analyzing the stability of an MoC scheme and verifying the analysis numerically, we have assumed periodic b.c. in our systems of PDEs. Another type of b.c. that can be used are nonreflecting b.c. When nonreflecting b.c. are used, the boundary values are computed at every time step by a known function. This means:

$$(Y^+)_0^n = F^+(n), \quad (Y^-)_M^n = F^-(n), \tag{5.13}$$

where $F^{\pm}$ are given. The numerical stability of the MoC-SE and MoC-ME using nonreflecting boundary conditions has been studied in [12]. In that paper, it was shown that the MoC-ME becomes stable when nonreflecting b.c. are used. In this chapter, we present results which the constant nonreflecting b.c. given by:

$$F^{\pm}_{1,3} = 0, \quad F^{\pm}_{2} = \pm 1, \tag{5.14}$$

where subscripts $1, 2, 3$ refer to the components $Y^{\pm}_{1,2,3} \equiv S^{\pm}_{1,2,3}$ of the solution (2.7) of our model problem (1.50). The logarithm of the Fourier spectrum of the error in the MoC-ME case is shown in Fig. 5.6.

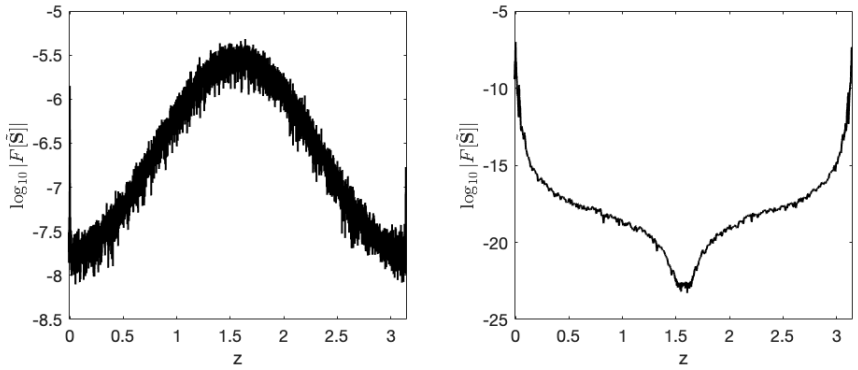

Figure 5.6: Numerical solution of system (1.50) using the MoC-ME with periodic b.c. (left) and nonreflecting b.c. (right) up to $t = 500$ with $h = 0.01$. As we saw in Chapter 2, there is a mild instability near $z = \pi/2$ when periodic b.c. are used whose growth rate is $O(h)$. When nonreflecting b.c. are used, this instability vanishes, and the error in the middle of the spectrum decays.

The question we would now like to answer is: will nonreflecting b.c. eliminate the instability in MoC-pRK schemes? In this chapter, we will answer this question for the MoC-pRK3.

5.7 Implementing nonreflecting b.c. in the MoC-pRK3

In Sec. 5.4, we explained how some care must be taken when implementing the MoC-pRK3 with periodic b.c. Even more care is needed when implementing this scheme with nonreflecting b.c. We will explain this process for Y^+ by advancing through the stencil.

We obtain the solution $(Y^+)_m^{n=1}$ from the initial condition. As described in Chapter 4, the first step of the pRK3 can be taken by the ME. Thus, the first step of the MoC-pRK3 is taken by the MoC-ME, giving us $(Y^+)_m^2$. The first time we use the MoC-pRK3 scheme is when we are computing the solution at time level $n = 3$. The corresponding stencil is shown in Fig. 5.7.

Computation of the solution at the boundary is straightforward, as these values are prescribed by (5.13). As such, the value $(Y^+)_0^3$ comes for free.

The next value we compute is $(Y^+)_1^3$. In the MoC-pRK3, this requires the solution and stage derivatives at $(Y^+)_0^2$ and $(Y^+)_{-1}^1$. The former value is available to us from the MoC-ME; the latter value, however, is not. Note that we need this value with at least local accuracy $O(h^3)$. This is because $(Y^+)_1^3$ must be computed with local accuracy $O(h^4)$, and thus all values on the r.h.s. of (5.1i) must be $O(h^4)$. The terms required at level $n = 2$ are multiplied by h in (5.1i), so they must be at least $O(h^3)$.

The resolution is to extrapolate the value $(Y^+)_{-1}^1$ with the desired accuracy. This process is explained in Appendix E. In this case, one can compute:

$$(Y^+)_{-1}^1 = 3(Y^+)_0^1 - 3(Y^+)_1^1 + (Y^+)_2^1 + O(h^3). \tag{5.15}$$

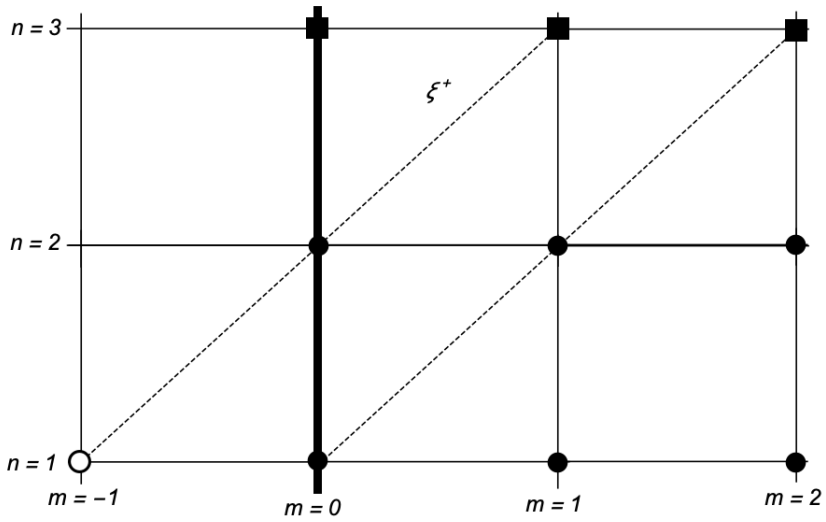

Figure 5.7: Stencil showing the advancement of the solution along ξ^+ to $(Y^+)^3_m$ using the MoC-pRK3. Filled-in circles represent nodes where the solution and stage derivatives are available. Currently, $n = 1$ and $n = 2$ are available for all $0 \leq m \leq M$ from the initial condition and first step with MoC-ME. Filled-in squares represent the nodes we are advancing to; thus, all nodes at level $n = 3$ with $0 \leq m \leq M$ have a filled-in square. Empty circles represent nodes which are not available, but are needed to compute a node with a filled-in square. Currently, the only node like this is $(m = -1, n = 1)$, as the solution and stage derivatives are needed here in order to compute the solution at $(m = 1, n = 3)$.

With this point in hand, we can now compute $(Y^+)^3_1$. The remaining points at level $n = 3$, $(Y^+)^3_m, m = 2, \ldots, M$ are straightforward to compute because all of the values needed at previous time levels are available. A similar routine is, of course, used at the right boundary to compute Y^- at level $n = 3$.

Let us make a very important remark: the extrapolation (5.15) will only be used to compute a point outside of the boundary at level $n = 1$. This is because the extrapolation involved the Lagrange polynomial, which assumes a smooth solution. Thus, if the numerical solution happens to be non-smooth, the extrapolation (5.15) could give a poor approximation. Moving forward, extrapolation outside of the boundary will be computed by another method, which will be described soon.

We now describe how to advance the solution from $n = 3$ to $n = 4$. The corresponding stencil is shown in Fig. 5.8.

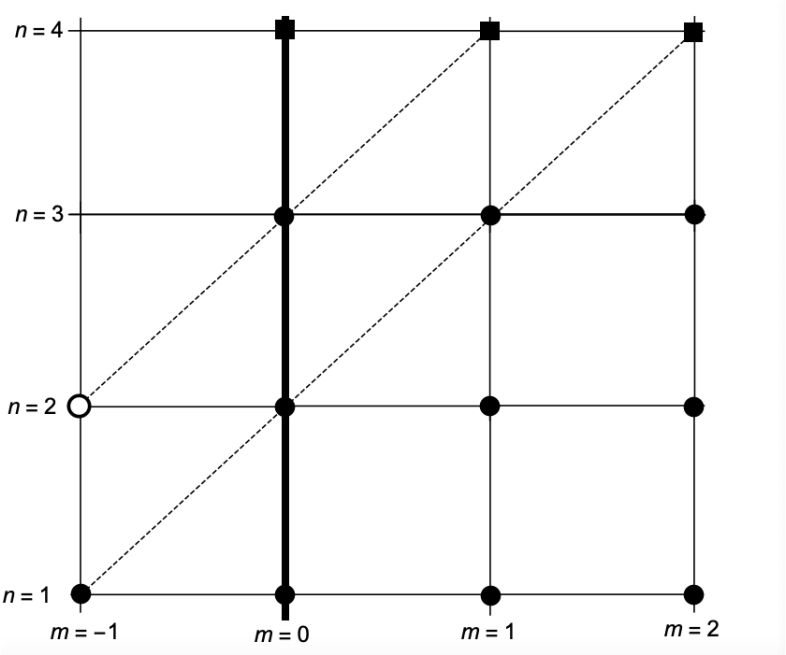

Figure 5.8: Stencil showing the advancement of the solution along ξ^+ to $(Y^+)^4_m$ using the MoC-pRK3. The solution $(Y^+)^4_0$ is supplied by the boundary condition, and $(Y^+)^4_2$ is found by the MoC-pRK3 algorithm using the available solutions at nodes with $(m, n) = (1, 3)$ and $(0, 2)$. To compute $(Y^+)^4_1$, we need the solution at node $(-1, 2)$, which is not yet available.

Our goal now is to explain how one computes the point $(m = -1, n = 2)$ with the desired local accuracy $O(h^3)$. As mentioned above, we will not do this by extrapolation. We can, however, do this with the MoC-ME. We will demonstrate the idea in Fig. 5.9 and then explain it in the text.

According to the rotated stencil in Fig. 5.9, the desired point can be computed

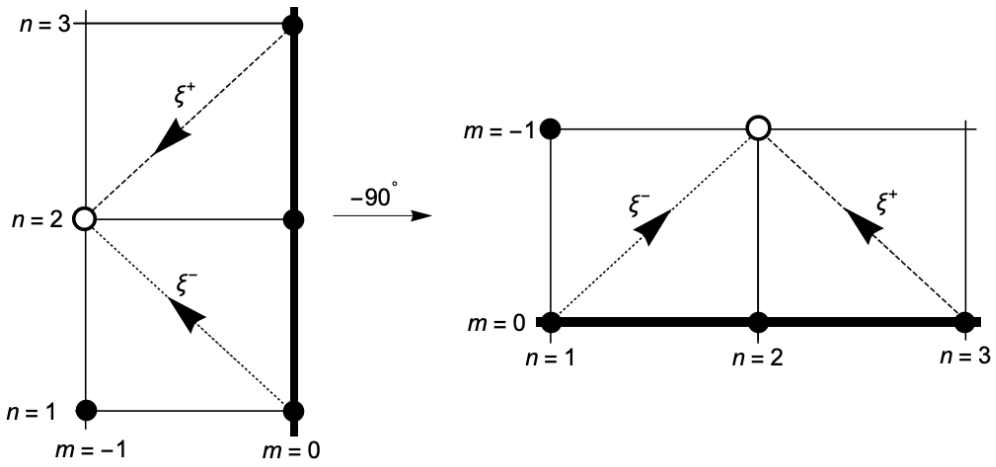

Figure 5.9: Left pane: bottom-left rectangle of the stencil in Fig. 5.8; right pane: the same rectangle rotated by −90 degrees. We include the characteristics ξ^+ (dashed) and ξ^- (dotted), as well as the direction in which they propagate. The point which we are interested in computing is represented by the open circle. When the usual stencil is rotated, we obtain an MoC-ME-like stencil, with the locations of ξ^+ and ξ^- reversed. From the left pane, notice also that in the rotated stencil, ξ^+ propagates in the opposite direction compared to that in the bulk of the grid.

by the MoC-ME as follows:

$$\left(\overline{Y}^{+}\right)_{m=-1}^{n=2} = (Y^{+})_0^3 - hf^{+}\left((Y^{+})_0^3, (Y^{-})_0^3\right), \tag{5.16a}$$

$$\left(\overline{Y}^{-}\right)_{m=-1}^{n=2} = (Y^{-})_0^1 + hf^{-}\left((Y^{+})_0^1, (Y^{-})_0^1\right), \tag{5.16b}$$

$$\left(\ddot{Y}^{+}\right)_{m=-1}^{n=2} = \frac{1}{2}\left[(Y^{+})_0^3 + \left(\overline{Y}^{+}\right)_{-1}^{2} - hf^{+}\left(\left(\overline{Y}^{+}\right)_{-1}^{2}, \left(\overline{Y}^{-}\right)_{-1}^{2}\right)\right], \tag{5.16c}$$

$$\left(\ddot{Y}^{-}\right)_{m=-1}^{n=2} = \frac{1}{2}\left[(Y^{-})_0^1 + \left(\overline{Y}^{-}\right)_{-1}^{2} + hf^{+}\left(\left(\overline{Y}^{+}\right)_{-1}^{2}, \left(\overline{Y}^{-}\right)_{-1}^{2}\right)\right]. \tag{5.16d}$$

This closely resembles the MoC-ME seen many times in this thesis, with two main differences. First, the node to which ξ^+ propagates in (5.16) is the node that ξ^- propagates from in the usual MoC-ME. Likewise, the node from which ξ^- propagates in (5.16) is the node that ξ^+ propagates from in the usual MoC-ME. Second, in

(5.16a) and (5.16c), the function evaluations are multiplied by **negative** h. This reflects the fact, shown in the stencil, that ξ^+ in this scheme propagates away from the desired node, rather than towards it. Thus, we can "reverse" the direction by stepping backwards, i.e. by using the stepsize $-h$.

Equations (5.16) give us the desired values at node $(-1, 2)$, which can then be used to compute the stage derivatives needed to find $(Y^+)_1^4$. On the right boundary, we encounter a similar situation: to find the Y^- values at $(M-1, 4)$, we need values at $(M+1, 2)$. The analogue of (5.16) at the right boundary to accomplish this task is:

$$\left(\overline{Y}^+\right)_{M+1}^2 = (Y^+)_M^1 + hf^+\left((Y^+)_M^1, (Y^-)_M^1\right), \tag{5.17a}$$

$$\left(\overline{Y}^-\right)_{M+1}^2 = (Y^-)_M^3 - hf^-\left((Y^+)_M^3, (Y^-)_M^3\right), \tag{5.17b}$$

$$\left(\ddot{Y}^+\right)_{M+1}^2 = \frac{1}{2}\left[(Y^+)_M^1 + \left(\overline{Y}^+\right)_{M+1}^2 + hf^+\left(\left(\overline{Y}^+\right)_{M+1}^2, \left(\overline{Y}^-\right)_{M+1}^2\right)\right], \tag{5.17c}$$

$$\left(\ddot{Y}^-\right)_{M+1}^2 = \frac{1}{2}\left[(Y^+)_M^3 + \left(\overline{Y}^+\right)_{M+1}^2 - hf^+\left(\left(\overline{Y}^+\right)_{M+1}^2, \left(\overline{Y}^-\right)_{M+1}^2\right)\right]. \tag{5.17d}$$

The solution at all levels $n > 4$ will follow a similar pattern. The pseudocode for the MoC-pRK3 with nonreflecting b.c. is shown in Algorithm 1. In Appendix F, we will include our commented MATLAB code.

5.8 Numerical results of MoC-pRK3 with nonreflecting b.c.

The results of simulating system (1.50) with the MoC-pRK3 using nonreflecting b.c. (5.14) are shown in Fig. 5.10. We see that nonreflecting b.c. in the MoC-pRK3 have

Algorithm 1: MoC-pRK3 algorithm with nonreflecting boundary conditions

while $n < nmax$ **do**
  if $n = 1$ **then**
    $(Y^{\pm})^{1}_{[0,M]}$ given by initial condition;
    Extrapolate $(Y^{\pm})^{1}_{-1,M+1}$ by (5.15);
  else if $n = 2$ **then**
    Compute $(Y^{\pm})^{2}_{[0,M]}$ by MoC-ME;
  else
    Compute $(\ddot{Y}^{\pm})^{n}_{[0,M]}$ by MoC-ME.
     As a byproduct, we obtain the MoC-SE solution $(\overline{Y}^{\pm})^{n}_{[0,M]}$;
    if $n = 3$ **then**
      Compute $(k^{+}_{1,2})^{1}_{[-1,M-2]}$ and $(k^{-}_{1,2})^{1}_{[2,M+1]}$ by (5.1{a,b,g,h});
    else
      Compute $(Y^{\pm})^{n-2}_{-1,M+1}$ by rotated MoC-ME ;
      Set $(k^{+}_{1})^{n-2}_{[-1,M-2]} = \left[f^{+}((Y^{+})^{n-2}_{-1}, (Y^{-})^{n-2}_{-1}), (k^{+}_{1})^{n-1}_{[0,M-2]}\right]$;
      Set $(k^{-}_{1})^{n-2}_{[2,M+1]} = \left[(k^{-}_{1})^{n-1}_{[2,M]}, f^{-}((Y^{+})^{n-2}_{M+1}, (Y^{-})^{n-1}_{M+1})\right]$;
      Compute $(\overline{Y}^{+})^{n-2}_{0}$ and $(\overline{Y}^{-})^{n-2}_{M}$ by MoC-SE using $(Y^{\pm})^{n-2}_{-1,M+1}$;
      Set $(k^{+}_{2})^{n-2}_{[-1,M-2]} = \left[f^{+}((\overline{Y}^{+})^{n-2}_{0}, (Y^{-})^{n-2}_{0}), (k^{+}_{2})^{n-1}_{[0,M-2]}\right]$;
      Set $(k^{-}_{2})^{n-2}_{[2,M+1]} = \left[(k^{-}_{2})^{n-1}_{[2,M]}, f^{-}((Y^{+})^{n-2}_{M}, (\overline{Y}^{-})^{n-2}_{M})\right]$;
    end Set $(k^{+}_{1})^{n-1}_{[0,M-1]} = f^{+}\left((Y^{+})^{n-1}_{[0,M-1]}, (Y^{-})^{n-1}_{[0,M-1]}\right)$;
    Set $(k^{-}_{1})^{n-1}_{[1,M]} = f^{-}\left((Y^{+})^{n-1}_{[1,M]}, (Y^{-})^{n-1}_{[1,M]}\right)$;
    Set $(k^{+}_{2})^{n-1}_{[0,M-1]} = f^{+}\left((\overline{Y}^{+})^{n}_{[1,M]}, (\ddot{Y}^{-})^{n}_{[1,M]}\right)$;
    Set $(k^{-}_{2})^{n-1}_{[1,M]} = f^{-}\left((\ddot{Y}^{+})^{n}_{[0,M-1]}, (\overline{Y}^{-})^{n}_{[0,M-1]}\right)$;
    Compute $(Y^{\pm})^{n}_{[0,M]}$ by MoC-pRK3;
  end
end

a similar effect on the error that they do in the MoC-ME. Thus, a stable, third-order MoC scheme has been established.

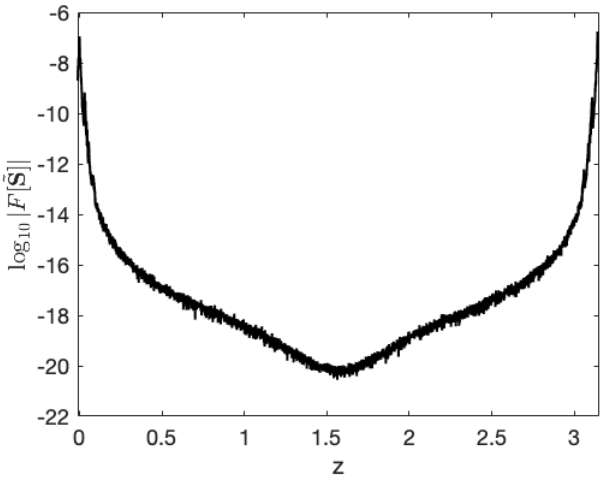

Figure 5.10: Logarithm of the Fourier spectrum of the error obtained by solving (1.50) with the MoC-pRK3 using nonreflecting b.c. (5.14) up to $t = 500$ with $h = 0.01$. One sees that the error decays most drastically near the middle of the spectrum, contrary to what occurs when periodic b.c. are used. However, one may also notice that the middle of the spectrum for the MoC-pRK3 does not "sag" as strongly as it does for the MoC-ME; compare with Fig. 5.6.

In Chapter 6, we will establish a stable, fourth-order MoC scheme using the pRK4 as the ODE solver along with nonreflecting b.c. In Chapter 7, we will present numerical results for both methods verifying their accuracy as well as their ability to preserve conserved quantities.

5.9 An alternative implementation of nonreflecting b.c. in the MoC-pRK3

The implementation of the MoC-pRK3 with nonreflecting b.c. becomes rather complicated due to the need to compute points that lie outside the boundary. However,

this only needs to be done so that one can compute $(Y^+)_1^n$, and similarly $(Y^-)_{M-1}^n$ by the MoC-pRK3 algorithm (see Fig. 5.8). The natural question that one might as is: why not just compute these two values at each level by another third-order method, say MoC-cRK3, to avoid exiting the boundary?

The answer to this question is stability. Recall from Chapter 3 that the MoC-cRK3 can be strongly unstable for harmonics close to π. Although we would only use the MoC-cRK3 to compute the values at two nodes per level, the instability of the MoC-cRK3 eventually pervades the solution. This is shown in Fig. 5.11.

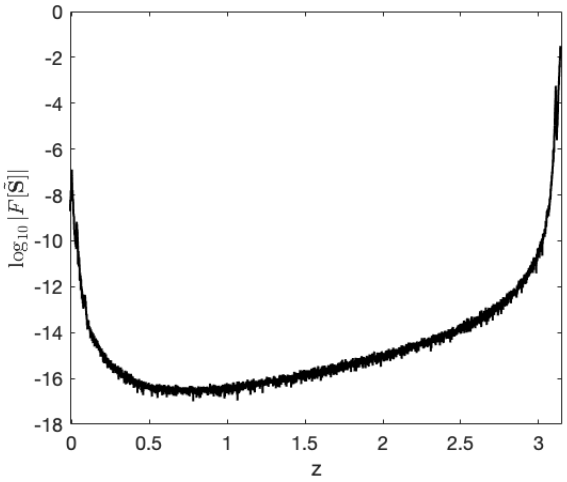

Figure 5.11: Logarithm of the Fourier spectrum of the error obtained by solving (1.50) with the MoC-pRK3 using the cRK3-style implementation of nonreflecting b.c. While the error does decay throughout most of the spectrum, one can see that it is growing for high Fourier harmonics, an emblematic feature of the MoC-cRK3. Soon, this error will be order 1, which will completely destroy the solution.

Based on these considerations, we advocate for the use of the implementation described in Sec. 5.8 to avoid any risk of instability.

6 The MoC using fourth-order pRK schemes

In Chapter 5, we formulated an MoC scheme which used the third-order pRK method developed in Chapter 4. We analyzed the stability of the resulting MoC-pRK3 and discussed how to implement the method using both periodic and nonreflecting boundary conditions. In this Chapter, we will perform the same tasks but now with the fourth-order pRK method. In Chapter 7, we will provide the numerical verification that the MoC-pRK3 and MoC-pRK4 are third- and fourth-order methods, respectively, and that they preserve conserved quantities to the degree predicted in Chapter 4.

6.1 Formulation of the MoC-pRK4

Recall the pRK4 scheme for ODEs from Chapter 4:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{37}{24} & \frac{3}{8} \\ & \frac{-7}{12} & \frac{-3}{4} \\ & \frac{1}{24} & \frac{3}{8} \end{array} \tag{4.20}$$

For the ODE $y' = f(Y_n, t_n)$, method (4.20) has the following meaning:

$$(k_1)_n = f(Y_n, t_n), \tag{6.1a}$$

$$(k_2)_n = f(Y_n + h(k_1)_n, t_n + h), \tag{6.1b}$$

$$Y_{n+1} = Y_n + h\left(\frac{37}{24}(k_1)n + \frac{3}{8}(k_2)_n - \frac{7}{12}(k_1)_{n-1} - \frac{3}{4}(k_2)_{n-1} + \frac{1}{24}(k_1)_{n-2} + \frac{3}{8}(k_2)_{n-2}\right). \tag{6.1c}$$

The stencil for the MoC-pRK4 is shown in Fig. 6.1.

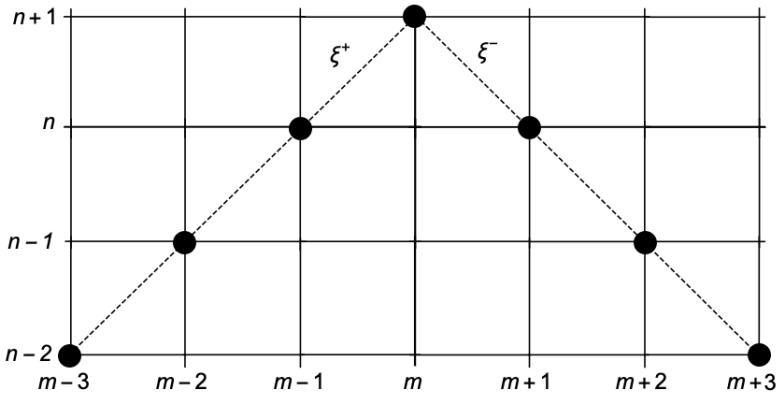

Figure 6.1: Stencil for the MoC-pRK4. To obtain either of $(Y^{\pm})_m^{n+1}$*, one needs information of both* $(Y\pm)_{m\mp1}^{n}$*, of both* $(Y^{\pm})_{m\mp2}^{n-1}$*, and of both* $(Y^{\pm})_{m\mp3}^{n-2}$*.*

The equations for $k_1^{\pm}$ in the MoC-pRK4 are:

$$(k_1^+)_{m-1}^n = f^+\left((Y^+)_{m-1}^n, (Y^-)_{m-1}^n\right), \tag{6.2a}$$

$$(k_1^-)_{m+1}^n = f^-\left((Y^+)_{m+1}^n, (Y^-)_{m+1}^n\right). \tag{6.2b}$$

Recall our convention, stated after Eqs. (3.3), that the indices of the stage derivatives k's are those *from* which the respective k's are computed, while the indices of the

auxiliary solutions $\overline{Y}^{\pm}, \ddot{Y}^{\pm}$, and $\widetilde{Y}^{\pm}$ in (6.2) are those *at* which these solutions are found.

According to Eqs. (6.1b) and (6.1c), to compute $k_2^{\pm}$ in the MoC-pRK4 at level n, one needs $Y^{\mp}$ at level $n+1$ with local accuracy $O(h^4)$ to achieve a local accuracy of $O(h^5)$ (and hence a global accuracy of $O(h^4)$) in the solution (6.1c). Thus, we need a third-order accurate method to estimate those $\overline{Y}^{\pm}$. The third-order accurate MoC schemes which we can use are the MoC-cRK3 and MoC-pRK3. In Chapters 3 and 5, we showed that the former of these methods can be strongly unstable, while the latter is not. Thus, we opt to compute the $O(h^4)$-local accurate solution by the MoC-pRK3. As seen in Chapter 5, this requires the MoC-SE and MoC-ME solutions, given by:

$$\left(\overline{Y}^{+}\right)_m^{n+1} = (Y^{+})_{m-1}^{n} + h(k_1^{+})_{m-1}^{n}, \tag{6.2c}$$

$$\left(\overline{Y}^{-}\right)_m^{n+1} = (Y^{-})_{m+1}^{n} + h(k_1^{-})_{m+1}^{n}, \tag{6.2d}$$

$$\left(\ddot{Y}^{+}\right)_m^{n+1} = \frac{1}{2}\left[(Y^{+})_{m-1}^{n} + \left(\overline{Y}^{+}\right)_m^{n+1} + hf^{+}\left(\left(\overline{Y}^{+}\right)_m^{n+1}, \left(\overline{Y}^{-}\right)_m^{n+1}\right)\right], \tag{6.2e}$$

$$\left(\ddot{Y}^{-}\right)_m^{n+1} = \frac{1}{2}\left[(Y^{-})_{m+1}^{n} + \left(\overline{Y}^{-}\right)_m^{n+1} + hf^{-}\left(\left(\overline{Y}^{+}\right)_m^{n+1}, \left(\overline{Y}^{-}\right)_m^{n+1}\right)\right], \tag{6.2f}$$

as well as the stage derivatives

$$(\ell_2^{+})_{m-1}^{n} = f^{+}\left(\left(\overline{Y}^{+}\right)_m^{n+1}, \left(\ddot{Y}^{-}\right)_m^{n+1}\right), \tag{6.2g}$$

$$(\ell_2^{-})_{m+1}^{n} = f^{-}\left(\left(\ddot{Y}^{+}\right)_m^{n+1}, \left(\overline{Y}^{-}\right)_m^{n+1}\right). \tag{6.2h}$$

The values $\ell_2^{\pm}$ were referred to as $k_2^{\pm}$ in Chapter 5. We use a different notation here as we reserve the k notation for the stage derivatives in the MoC-pRK4. The MoC-pRK3

solution, $\widetilde{Y}^{\pm}$, is then found by:

$$\left(\widetilde{Y}^{+}\right)_{m}^{n+1} = (Y^{+})_{m-1}^{n} + \frac{h}{12}\left(13(k_1^{+})_{m-1}^{n} + 5(\ell_2^{+})_{m-1}^{n} - (k_1^{+})_{m-2}^{n-1} - 5(\ell_2^{+})_{m-2}^{n-1}\right), \quad (6.2i)$$

$$\left(\widetilde{Y}^{-}\right)_{m}^{n+1} = (Y^{-})_{m+1}^{n} + \frac{h}{12}\left(13(k_1^{-})_{m+1}^{n} + 5(\ell_2^{-})_{m+1}^{n} - (k_1^{-})_{m+2}^{n-1} - 5(\ell_2^{-})_{m+2}^{n-1}\right). \quad (6.2j)$$

We can now compute $k_2^{\pm}$ for the MoC-pRK4:

$$(k_2^{+})_{m-1}^{n} = f^{+}\left(\left(\overline{Y}^{+}\right)_{m}^{n+1}, \left(\widetilde{Y}^{-}\right)_{m}^{n+1}\right), \quad (6.2k)$$

$$(k_2^{-})_{m+1}^{n} = f^{-}\left(\left(\widetilde{Y}^{+}\right)_{m}^{n+1}, \left(\overline{Y}^{-}\right)_{m}^{n+1}\right). \quad (6.2l)$$

By Eq. (6.1c), the MoC-pRK4 solution is then given by:

$$(Y^{+})_{m}^{n+1} = (Y^{+})_{m-1}^{n} + h\left(\frac{37}{24}(k_1^{+})_{m-1}^{n} + \frac{3}{8}(k_2^{+})_{m-1}^{n} - \frac{7}{12}(k_1^{+})_{m-2}^{n-1} - \frac{3}{4}(k_2^{+})_{m-2}^{n-1} + \frac{1}{24}(k_1^{+})_{m-3}^{n-2} + \frac{3}{8}(k_2^{+})_{m-3}^{n-2}\right), \quad (6.2m)$$

$$(Y^{-})_{m}^{n+1} = (Y^{-})_{m+1}^{n} + h\left(\frac{37}{24}(k_1^{-})_{m+1}^{n} + \frac{3}{8}(k_2^{-})_{m+1}^{n} - \frac{7}{12}(k_1^{-})_{m+2}^{n-1} - \frac{3}{4}(k_2^{-})_{m+2}^{n-1} + \frac{1}{24}(k_1^{-})_{m+3}^{n-2} + \frac{3}{8}(k_2^{-})_{m+3}^{n-2}\right). \quad (6.2n)$$

As in the MoC-pRK3, the values $k_{1,2}^{\pm}$ at levels $n-1$ and $n-2$ are computed in the previous steps by an identical procedure. Equations (6.2) form the MoC-pRK4.

6.2 von Neumann stability analysis of the MoC-pRK4

We will investigate the stability properties of the MoC-cRK4 by performing the von Neumann stability analysis on the background of system (1.50). In the stability analysis of the MoC-pRK3, we sought an equation of the form:

$$\left((\mathbf{s})_m^{n+1}, (\mathbf{s})_m^n\right)^T = \mathbf{\Phi}(z)\left((\mathbf{s})_m^n, (\mathbf{s})_m^{n-1}\right). \tag{5.3}$$

The need to evolve $(\mathbf{s})_m^{n-1}$ arose from the dependence of the MoC-pRK3 solution on stage derivatives at level $n-1$. In the analysis of the MoC-pRK3, we seek an equation of the form:

$$\begin{pmatrix} (\mathbf{s})_m^{n+1} \\ (\mathbf{s})_m^n \\ (\mathbf{s})_m^{n-1} \\ (\mathbf{s})_m^{n-2} \end{pmatrix} = \mathbf{\Phi}(z) \begin{pmatrix} (\mathbf{s})_m^n \\ (\mathbf{s})_m^{n-1} \\ (\mathbf{s})_m^{n-2} \\ (\mathbf{s})_m^{n-3} \end{pmatrix} \equiv \mathbf{\Phi}(z)\ \hat{\mathbf{s}}. \tag{6.3}$$

Each of the perturbation vectors $\mathbf{s}$ are of length 4, and thus $\mathbf{\Phi}$ must be a 16×16 matrix. The need to include the values of $\mathbf{s}$ at 4 levels will become evident in the forthcoming analysis, but stems from the fact that the MoC-pRK4 requires stage derivatives from two previous steps, as well as the MoC-pRK3 solution at every step (see Eqs. (6.2{i,j}).

Equations (6.2{a,b}) applied to (2.11), the linearized version of (1.50), yield:

$$\left(\mathbf{k}_1^{\pm}\right)_m^n = [\mathbf{P}, \mathcal{O}, \mathcal{O}, \mathcal{O}]\,\hat{\mathbf{s}} \equiv \mathcal{K}_{11}\,\hat{\mathbf{s}}, \tag{6.4a}$$

$$\left(\mathbf{k}_1^{\pm}\right)_m^{n-1} = [\mathcal{O}, \mathbf{P}, \mathcal{O}, \mathcal{O}]\,\hat{\mathbf{s}} \equiv \mathcal{K}_{12}\,\hat{\mathbf{s}}, \tag{6.4b}$$

$$\left(\mathbf{k}_1^{\pm}\right)_m^{n-2} = [\mathcal{O}, \mathcal{O}, \mathbf{P}, \mathcal{O}]\,\hat{\mathbf{s}} \equiv \mathcal{K}_{13}\,\hat{\mathbf{s}}, \tag{6.4c}$$

$$\left(\mathbf{k}_1^{\pm}\right)_m^{n-3} = [\mathcal{O}, \mathcal{O}, \mathcal{O}, \mathbf{P}]\,\hat{\mathbf{s}} \equiv \mathcal{K}_{14}\,\hat{\mathbf{s}}, \tag{6.4d}$$

where $\mathbf{P}$ is defined in (2.12), $\mathcal{O}$ is the 4×4 zero matrix, and $\hat{\mathbf{s}}$ is the 16×1 vector defined in (6.3). Recall that each $\mathbf{k}^{\pm}$ has length 4, so the dimensions on each side of Eqs. (6.4) agree.

According to (6.2{a,b}), we require the MoC-SE solution. The analogue of the MoC-SE stability matrix in the context of the MoC-pRK4 is:

$$(\bar{\mathbf{s}})_m^{n+1} = \Big([\mathbf{Q}, \mathcal{O}, \mathcal{O}, \mathcal{O}] + h\mathbf{Q}\mathcal{K}_{11} \Big)\hat{\mathbf{s}} \equiv \mathcal{S}_{11}\,\hat{\mathbf{s}}, \tag{6.4e}$$

$$(\bar{\mathbf{s}})_m^{n} = \Big([\mathcal{O}, \mathbf{Q}, \mathcal{O}, \mathcal{O}] + h\mathbf{Q}\mathcal{K}_{12} \Big)\hat{\mathbf{s}} \equiv \mathcal{S}_{12}\,\hat{\mathbf{s}}, \tag{6.4f}$$

$$(\bar{\mathbf{s}})_m^{n-1} = \Big([\mathcal{O}, \mathcal{O}, \mathbf{Q}, \mathcal{O}] + h\mathbf{Q}\mathcal{K}_{13} \Big)\hat{\mathbf{s}} \equiv \mathcal{S}_{13}\,\hat{\mathbf{s}}, \tag{6.4g}$$

$$(\bar{\mathbf{s}})_m^{n-2} = \Big([\mathcal{O}, \mathcal{O}, \mathcal{O}, \mathbf{Q}] + h\mathbf{Q}\mathcal{K}_{14} \Big)\hat{\mathbf{s}} \equiv \mathcal{S}_{14}\,\hat{\mathbf{s}}, \tag{6.4h}$$

where $\mathbf{Q}$ is defined in (2.21). Next, we find the analogue of the MoC-ME stability

matrix:

$$(\ddot{\mathbf{s}})_m^{n+1} = \left[\frac{1}{2}\Big([\mathbf{Q},\mathcal{O},\mathcal{O},\mathcal{O}] + \mathcal{S}_{11} + h\mathcal{K}_{11}\mathcal{M}_{11}\Big)\right]\hat{\mathbf{s}} \equiv \mathcal{S}_{21}\,\hat{\mathbf{s}}, \tag{6.4i}$$

$$(\ddot{\mathbf{s}})_m^{n} = \left[\frac{1}{2}\Big([\mathcal{O},\mathbf{Q},\mathcal{O},\mathcal{O}] + \mathcal{S}_{12} + h\mathcal{K}_{12}\mathcal{M}_{12}\Big)\right]\hat{\mathbf{s}} \equiv \mathcal{S}_{22}\,\hat{\mathbf{s}}, \tag{6.4j}$$

$$(\ddot{\mathbf{s}})_m^{n-1} = \left[\frac{1}{2}\Big([\mathcal{O},\mathcal{O},\mathbf{Q},\mathcal{O}] + \mathcal{S}_{13} + h\mathcal{K}_{13}\mathcal{M}_{13}\Big)\right]\hat{\mathbf{s}} \equiv \mathcal{S}_{23}\,\hat{\mathbf{s}}, \tag{6.4k}$$

$$(\ddot{\mathbf{s}})_m^{n-2} = \left[\frac{1}{2}\Big([\mathcal{O},\mathcal{O},\mathcal{O},\mathbf{Q}] + \mathcal{S}_{14} + h\mathcal{K}_{14}\mathcal{M}_{14}\Big)\right]\hat{\mathbf{s}} \equiv \mathcal{S}_{24}\,\hat{\mathbf{s}}, \tag{6.4l}$$

where, in a similar fashion to the MoC-pRK3,

$$\mathcal{M}_{11} = \begin{bmatrix} \mathcal{S}_{11} \\ \mathcal{O}_{12\times 16} \end{bmatrix},\ \mathcal{M}_{12} = \begin{bmatrix} \mathcal{O}_{4\times 16} \\ \mathcal{S}_{12} \\ \mathcal{O}_{8\times 16} \end{bmatrix},\ \mathcal{M}_{13} = \begin{bmatrix} \mathcal{O}_{8\times 16} \\ \mathcal{S}_{13} \\ \mathcal{O}_{4\times 16} \end{bmatrix},\ \mathcal{M}_{14} = \begin{bmatrix} \mathcal{O}_{12\times 16} \\ \mathcal{S}_{14} \end{bmatrix}. \tag{6.5}$$

Applying (6.2{g,h}) to (2.11) gives us an analogue of (5.4{g,h}):

$$\left(\boldsymbol{\ell}_2^{\pm}\right)_{m\mp 1}^{n} = \Big([\mathbf{P}_{\text{diag}},\mathcal{O},\mathcal{O},\mathcal{O}]\,\mathcal{M}_{11} + [\mathbf{P}_{\text{offdiag}},\mathcal{O},\mathcal{O},\mathcal{O}]\,\mathcal{M}_{21}\Big)\hat{\mathbf{s}} \equiv \mathcal{L}_{21}\,\hat{\mathbf{s}}, \tag{6.4m}$$

$$\left(\boldsymbol{\ell}_2^{\pm}\right)_{m\mp 1}^{n-1} = \Big([\mathcal{O},\mathbf{P}_{\text{diag}},\mathcal{O},\mathcal{O}]\,\mathcal{M}_{12} + [\mathcal{O},\mathbf{P}_{\text{offdiag}},\mathcal{O},\mathcal{O}]\,\mathcal{M}_{22}\Big)\hat{\mathbf{s}} \equiv \mathcal{L}_{22}\,\hat{\mathbf{s}}, \tag{6.4n}$$

$$\left(\boldsymbol{\ell}_2^{\pm}\right)_{m\mp 1}^{n-2} = \Big([\mathcal{O},\mathcal{O},\mathbf{P}_{\text{diag}},\mathcal{O}]\,\mathcal{M}_{13} + [\mathcal{O},\mathcal{O},\mathbf{P}_{\text{offdiag}},\mathcal{O}]\,\mathcal{M}_{23}\Big)\hat{\mathbf{s}} \equiv \mathcal{L}_{23}\,\hat{\mathbf{s}}, \tag{6.4o}$$

$$\left(\boldsymbol{\ell}_2^{\pm}\right)_{m\mp 1}^{n-3} = \Big([\mathcal{O},\mathcal{O},\mathcal{O},\mathbf{P}_{\text{diag}}]\,\mathcal{M}_{14} + [\mathcal{O},\mathcal{O},\mathcal{O},\mathbf{P}_{\text{offdiag}}]\,\mathcal{M}_{24}\Big)\hat{\mathbf{s}} \equiv \mathcal{L}_{24}\,\hat{\mathbf{s}}, \tag{6.4p}$$

where

$$\mathcal{M}_{21} = \begin{bmatrix} \mathcal{S}_{21} \\ \mathcal{O}_{12\times 16} \end{bmatrix},\ \mathcal{M}_{22} = \begin{bmatrix} \mathcal{O}_{4\times 16} \\ \mathcal{S}_{22} \\ \mathcal{O}_{8\times 16} \end{bmatrix},\ \mathcal{M}_{23} = \begin{bmatrix} \mathcal{O}_{8\times 16} \\ \mathcal{S}_{23} \\ \mathcal{O}_{4\times 16} \end{bmatrix},\ \mathcal{M}_{24} = \begin{bmatrix} \mathcal{O}_{12\times 16} \\ \mathcal{S}_{24} \end{bmatrix}. \tag{6.6}$$

Applying (6.2{i,j}) to (2.11) gives us an analogue of the MoC-pRK3 matrix (5.4i):

$$(\tilde{\mathbf{s}})_m^{n+1} = \Big([\mathbf{Q}, \mathcal{O}, \mathcal{O}, \mathcal{O}] + \frac{h}{12} \left(13\mathbf{Q}\mathcal{K}_{11} + 5\mathcal{L}_{21} - \mathbf{Q}^2\mathcal{K}_{12} - 5\mathbf{Q}\mathcal{L}_{22} \right) \Big) \hat{\mathbf{s}} \equiv \mathcal{S}_{31}\, \hat{\mathbf{s}}, \tag{6.4q}$$

$$(\tilde{\mathbf{s}})_m^{n} = \Big([\mathcal{O}, \mathbf{Q}, \mathcal{O}, \mathcal{O}] + \frac{h}{12} \left(13\mathbf{Q}\mathcal{K}_{12} + 5\mathcal{L}_{22} - \mathbf{Q}^2\mathcal{K}_{13} - 5\mathbf{Q}\mathcal{L}_{23} \right) \Big) \hat{\mathbf{s}} \equiv \mathcal{S}_{32}\, \hat{\mathbf{s}}, \tag{6.4r}$$

$$(\tilde{\mathbf{s}})_m^{n-1} = \Big([\mathcal{O}, \mathcal{O}, \mathbf{Q}, \mathcal{O}] + \frac{h}{12} \left(13\mathbf{Q}\mathcal{K}_{13} + 5\mathcal{L}_{23} - \mathbf{Q}^2\mathcal{K}_{14} - 5\mathbf{Q}\mathcal{L}_{24} \right) \Big) \hat{\mathbf{s}} \equiv \mathcal{S}_{33}\, \hat{\mathbf{s}}. \tag{6.4s}$$

Equation (6.4s) is the reason we must include $(\mathbf{s})_m^{n-2}$ and $(\mathbf{s})_m^{n-3}$ in this stability analysis on, respectively, the left- and right-hand sides of (6.3). Indeed, the calculation of $(\tilde{\mathbf{s}})_m^{n-1}$ is required in the MoC-pRK4, which in turn requires the aforementioned values by way of $\mathcal{K}_{14}$ and $\mathcal{L}_{24}$ having a nonzero last 4×4 block. Note also that we do *not* need to compute $(\tilde{\mathbf{s}})_m^{n-2}$ (see Eqs. (6.2{k,l,m,n}), and hence do not need to include the solution at levels lower than $(n-3)$ in (6.3).

Continuing with the analysis, (6.2{k,l}) gives us:

$$\left(\mathbf{k}_2^{\pm}\right)_{m\mp1}^{n} = \Big(\left[\mathbf{P}_{\text{diag}}, \mathcal{O}, \mathcal{O}, \mathcal{O}\right] \mathcal{M}_{11} + \left[\mathbf{P}_{\text{offdiag}}, \mathcal{O}, \mathcal{O}, \mathcal{O}\right] \mathcal{M}_{31} \Big) \hat{\mathbf{s}} \equiv \mathcal{K}_{21}\, \hat{\mathbf{s}}, \tag{6.4t}$$

$$\left(\mathbf{k}_2^{\pm}\right)_{m\mp1}^{n-1} = \Big(\left[\mathcal{O}, \mathbf{P}_{\text{diag}}, \mathcal{O}, \mathcal{O}\right] \mathcal{M}_{12} + \left[\mathcal{O}, \mathbf{P}_{\text{offdiag}}, \mathcal{O}, \mathcal{O}\right] \mathcal{M}_{32} \Big) \hat{\mathbf{s}} \equiv \mathcal{K}_{22}\, \hat{\mathbf{s}}, \tag{6.4u}$$

$$\left(\mathbf{k}_2^{\pm}\right)_{m\mp1}^{n-2} = \Big(\left[\mathcal{O}, \mathcal{O}, \mathbf{P}_{\text{diag}}, \mathcal{O}\right] \mathcal{M}_{13} + \left[\mathcal{O}, \mathcal{O}, \mathbf{P}_{\text{offdiag}}, \mathcal{O}\right] \mathcal{M}_{33} \Big) \hat{\mathbf{s}} \equiv \mathcal{K}_{23}\, \hat{\mathbf{s}}, \tag{6.4v}$$

where

$$\mathcal{M}_{31} = \begin{bmatrix} \mathcal{S}_{31} \\ \mathcal{O}_{12\times16} \end{bmatrix}, \; \mathcal{M}_{32} = \begin{bmatrix} \mathcal{O}_{4\times16} \\ \mathcal{S}_{32} \\ \mathcal{O}_{8\times16} \end{bmatrix}, \; \mathcal{M}_{33} = \begin{bmatrix} \mathcal{O}_{8\times16} \\ \mathcal{S}_{33} \\ \mathcal{O}_{4\times16} \end{bmatrix}. \tag{6.7}$$

Equations (6.2{m,n}) applied to (2.11) yield:

$$\begin{aligned} (\mathbf{s})_m^{n+1} &= \Bigg(\left[\mathbf{Q}, \mathcal{O}, \mathcal{O}, \mathcal{O}\right] + h \Bigg(\frac{37}{24}\mathbf{Q}\mathcal{K}_{11} + \frac{3}{8}\mathcal{K}_{21} - \frac{7}{12}\mathbf{Q}^2\mathcal{K}_{12} - \frac{3}{4}\mathbf{Q}\mathcal{K}_{22} + \\ &\qquad\qquad \frac{1}{24}\mathbf{Q}^3\mathcal{K}_{13} + \frac{3}{8}\mathbf{Q}^2\mathcal{K}_{23} \Bigg) \Bigg) \hat{\mathbf{s}} \\ &\equiv \mathcal{N}_1\, \hat{\mathbf{s}}, \end{aligned} \tag{6.4w}$$

where the powers on $\mathbf{Q}$ are as such for reasons analogous to those explained in Chapter

5 after Eq. (5.4i). We can also write:

$$(\mathbf{s})_m^n = [\mathbf{I}, \mathcal{O}, \mathcal{O}, \mathcal{O}]\, \hat{\mathbf{s}} \equiv \mathcal{N}_2\, \hat{\mathbf{s}}, \tag{6.4x}$$

$$(\mathbf{s})_m^{n-1} = [\mathcal{O}, \mathbf{I}, \mathcal{O}, \mathcal{O}]\, \hat{\mathbf{s}} \equiv \mathcal{N}_3\, \hat{\mathbf{s}}, \tag{6.4y}$$

$$(\mathbf{s})_m^{n-2} = [\mathcal{O}, \mathcal{O}, \mathbf{I}, \mathcal{O}]\, \hat{\mathbf{s}} \equiv \mathcal{N}_4\, \hat{\mathbf{s}}, \tag{6.4z}$$

where $\mathbf{I}$ is the 4×4 identity matrix. Thus, the stability matrix $\mathbf{\Phi}$ in (6.3) is given by:

$$\mathbf{\Phi}(z) = \begin{bmatrix} \mathcal{N}_1 \\ \mathcal{N}_2 \\ \mathcal{N}_3 \\ \mathcal{N}_4 \end{bmatrix}. \tag{6.8}$$

In Fig. 6.2, we show the von Neumann stability plot corresponding to $\mathbf{\Phi}(\mathbf{z})$.

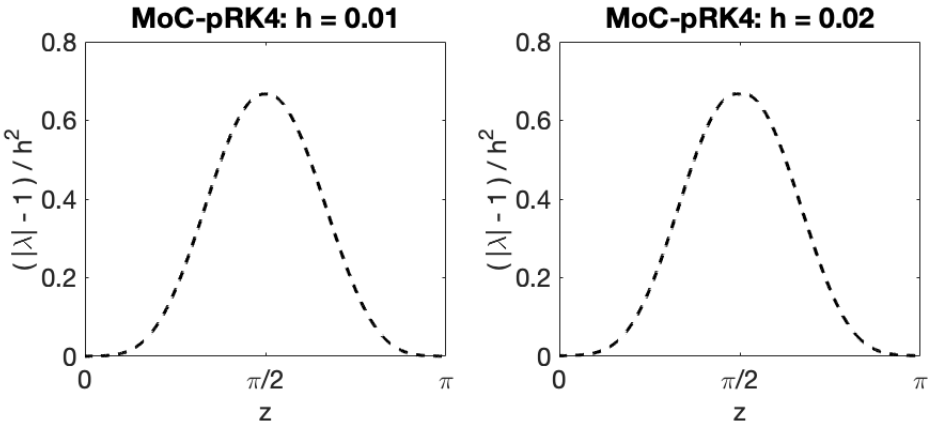

Figure 6.2: Largest eigenvalue of the stability matrix for the MoC-pRK4 for $h = 0.01$ and $h = 0.02$. Like the MoC-ME and MoC-pRK3, the largest instability occurs for harmonics near the middle of the spectrum. The growth rate of this instability is $O(h)$ because the maximum value of each plot is the same for both values of h (see Appendix B). This growth rate is roughly double than that for the MoC-pRK3. However, it is still small enough that the method is suitable for reasonably long calculations.

6.3 Numerical verification of the von Neumann analysis

Following the analysis of Appendix B, the growth constant in the MoC-pRK4 is $c \approx 0.67$. Therefore, if we solve system (1.50) with the MoC-pRK4 up to $t = 500$ with $h = 0.01$ using periodic b.c., we should expect the maximum error in the log-10 scale to be approximately -6.3. This is shown to be the case in Fig. 6.3.

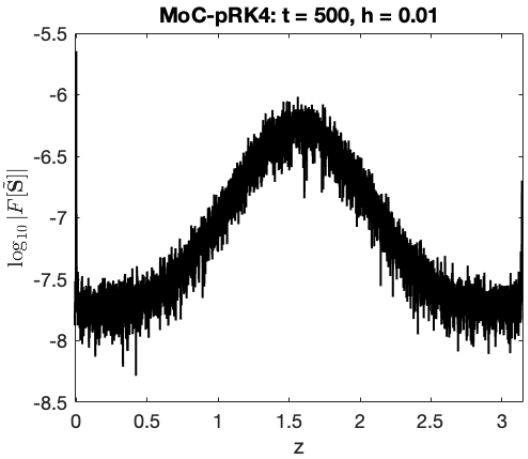

Figure 6.3: Logarithm of the Fourier spectrum of the error in the solution obtained by solving (1.50) with the MoC-pRK4 up to $t = 500$ with $h = 0.01$ using periodic b.c. The logarithm of the maximum error is close to -6.3, as predicted.

6.4 Implementation of the MoC-pRK4 scheme with periodic boundary conditions

Recall the meaning of periodic b.c. discussed in Chapter 5:

$$(Y^{\pm})^n_{-1} \equiv (Y^{\pm})^n_M, \quad (Y^{\pm})^n_{M+1} \equiv (Y^{\pm})^n_0, \tag{5.8}$$

where $M+1$ is the number of spatial grid points and $m = 0, 1, \ldots, M$. In an MoC-(p)RK scheme, the same convention applies to the values of the stage derivatives immediately outside of the boundary.

In the MoC-pRK4, we need values at three nodes outside of the left and right boundaries. This is illustrated in Fig. 6.4 for the case of computing $(Y^+)^{n+1}_{m=0,1}$. From

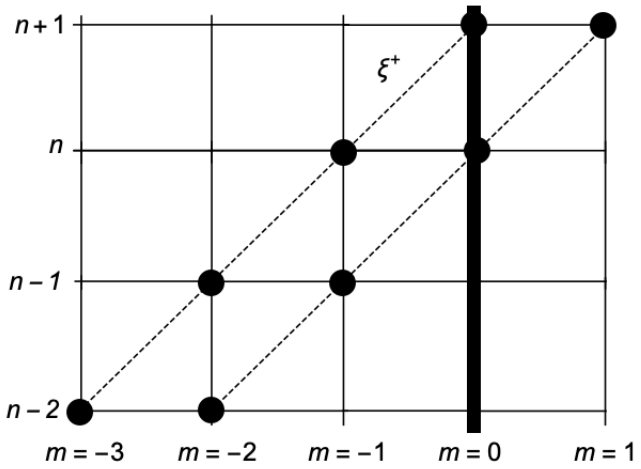

Figure 6.4: Stencil for computing $(Y^+)^{n+1}_0$ *and* $(Y^+)^{n+1}_1$ *along* ξ^+.

the stencil, we see that the computation of $(Y^+)^{n+1}_0$ requires the values at 3 nodes to the left of the boundary. Indeed, according to (6.2m),

$$(Y^+)^{n+1}_0 = (Y^+)^n_{-1} + h\left(\frac{37}{24}(k_1^+)^n_{-1} + \frac{3}{8}(k_2^+)^n_{-1} - \frac{7}{12}(k_1^+)^{n-1}_{-2} - \frac{3}{4}(k_2^+)^{n-2}_{-2} + \frac{1}{24}(k_1^+)^{n-2}_{-3} + \frac{3}{8}(k_2^+)^{n-2}_{-3}\right). \tag{6.9}$$

Values at the nodes $(m, n) = (-1, n)$ can be found as prescribed by (5.8). As we

described in Chapter 5, values at nodes $(-2, n-1)$ can be found by:

$$(Y^{\pm})_{-2}^{n-1} \equiv (Y^{\pm})_{M-1}^{n-1}. \tag{5.10a}$$

By the same logic, periodic b.c. are maintained at the nodes $(-3, n-2)$ by setting:

$$(Y^{\pm})_{-3}^{n-2} \equiv (Y^{\pm})_{M-2}^{n-1}. \tag{6.10a}$$

Similarly, for values at nodes outside the right boundary, one can set:

$$(Y^{\pm})_{M+2}^{n-1} \equiv (Y^{\pm})_{1}^{n-1}, \tag{5.10b}$$

$$(Y^{\pm})_{M+3}^{n-2} \equiv (Y^{\pm})_{2}^{n-2}. \tag{6.10b}$$

When computing $(Y^{+})_{1}^{n+1}$, the procedure for the outside-boundary values is the same, but with the upper index of the nodes shifted. This has no effect on how the periodic b.c. are applied. Our code which implements the MoC-pRK4 with periodic b.c. will be given in Appendix F.

6.5 Implementing nonreflecting b.c. in the MoC-pRK4

Recall that when nonreflecting b.c. are used, the boundary values are computed at every time step by a known function, meaning

$$(Y^{+})_{0}^{n} = F^{+}(n), \quad (Y^{-})_{M}^{n} = F^{-}(n). \tag{5.13}$$

In Chapter 5, we saw that when nonreflecting b.c. are used in the MoC-ME and MoC-pRK3 schemes, the instability that arises when periodic b.c. are used is eliminated. In this Chapter, we will show that this is a trait shared by the MoC-pRK4 as well. Before presenting the numerical results which show this, we will explain how one implements nonreflecting b.c. in the MoC-pRK4 scheme.

To compute any level of the solution by the MoC-pRK4, one needs the solution at the current level *and* the solution at the two previous levels. Thus, we cannot begin using the MoC-pRK4 until 3 levels of the solution are available. The solution at $n = 1$ is supplied by the initial condition. The next two levels must be computed with local accuracy $O(h^4)$. Our options for this include MoC-cRK3 and MoC-pRK3. The latter method, however, must be started by the MoC-ME. In theory, one could take the first step by MoC-cRK3 and the second step by MoC-pRK3. For simplicity, we choose to take both steps my MoC-cRK3. Although this method is unstable for certain systems (see Sec. 3.3), it will not hurt us to use it for these two steps only.

Once the first three levels are available, we can begin using the MoC-pRK4. Figure 6.5 shows the advancement of the solution from level $n = 3$ to $n = 4$ using the MoC-pRK4.

The value $(Y^+)_0^4$ is supplied by the boundary condition. The computation of $(Y^+)_1^4$ requires the yet unknown values of the stage derivatives, and hence of $Y^{\pm}$, at the nodes $(m, n) = (-2, 1)$ and $(-1, 2)$. Similarly, $(Y^+)_2^4$ requires the yet unknown values at $(-1, 1)$. Since the function evaluations of the values at these nodes will be multiplied by h in the MoC-pRK4 algorithm, we must compute them with local accuracy $O(h^4)$. We do not have the solution available at the appropriate points to compute these points with a third-order method such as the MoC-cRK3 or MoC-

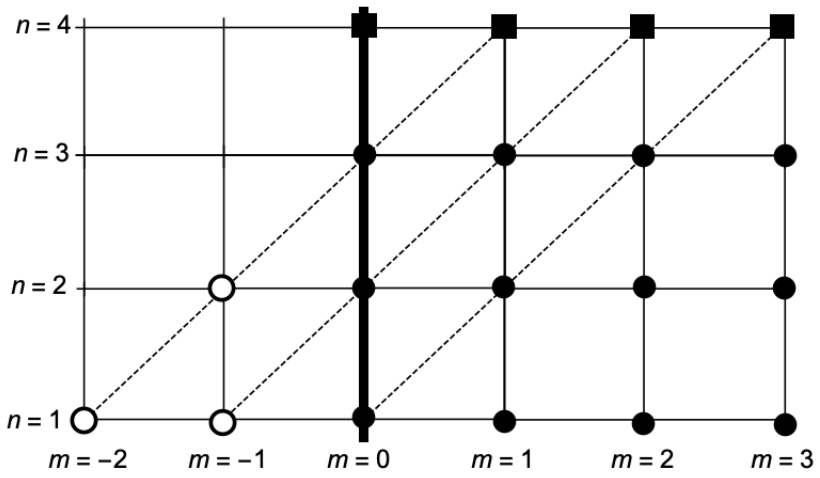

Figure 6.5: Stencil showing the advancement of the Y^+ solution along ξ^+ from $n = 3$ to $n = 4$ using the MoC-pRK4. Black squares represent the nodes we are advancing to, filled-in circles represent the nodes where the solution is available, and empty circles represent nodes which we need the values at in order to advance to $n = 4$ but do not have. In the current stencil, the solution inside the integration domain at $n = 1, 2, 3$ was obtained from the initial condition and the MoC-cRK3.

pRK3. Therefore, **at these levels and these levels only**, we extrapolate the points with the fourth-order formula derived in Appendix E:

$$(Y^{\pm})^1_{-1} = 4(Y^{\pm})^1_0 - 6(Y^{\pm})^1_1 + 4(Y^{\pm})^1_2 - (Y^{\pm})^1_3, \tag{6.11a}$$

$$(Y^{\pm})^1_{-2} = 4(Y^{\pm})^1_{-1} - 6(Y^{\pm})^1_0 + 4(Y^{\pm})^1_1 - (Y^{\pm})^1_2. \tag{6.11b}$$

The extrapolation of $(Y^+)^2_{-1}$ is similar. Since we will need the value $(Y^+)^2_{-2}$ when advancing to $n = 5$, we extrapolate it now as well. Again, we perform this extrapolation **only** at these two initial time levels because it may not be reliable for non-smooth solutions. Moreover, it is unknown how such an extrapolation, if done for all n, can affect the stability of the scheme.

With these values in hand we can now compute the MoC-pRK4 solutions $(Y^+)^4_{1,2}$.

The information required to find the solutions $(Y^{\pm})^4_m,\ m = 3, \ldots, M-3$ is fully available in the integration domain. Once we come to $m = M-2$, we perform an analogous extrapolation near the right boundary.

Now let us explain how to advance the MoC-pRK4 solution from $n = 4$ to $n = 5$. This includes *the trickiest part of the treatment of non-periodic b.c.*

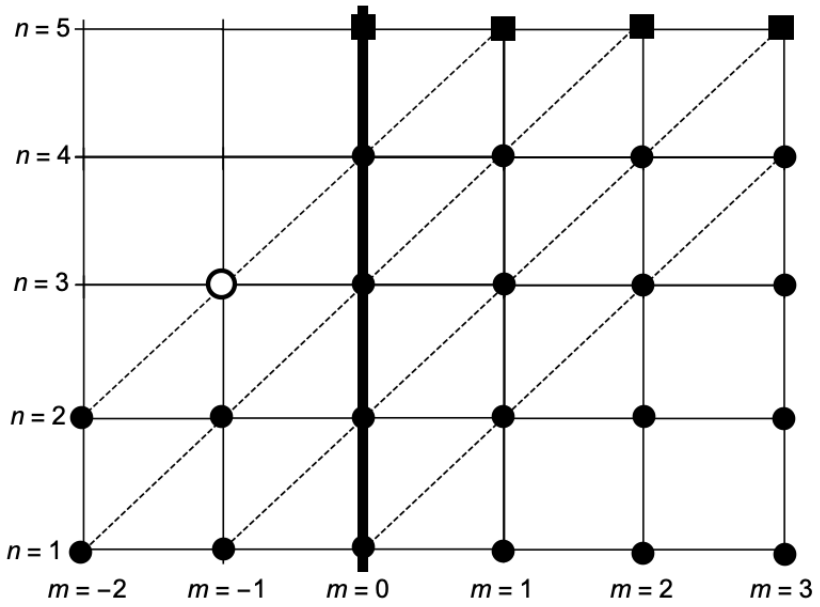

Figure 6.6: Stencil showing the advancement of the Y^+ solution from $n = 4$ to $n = 5$ using the MoC-pRK4. The only node (near the left boundary) where we need values but do not have them is $(-1, 3)$.

All values $(Y^-)^5_m$ near the left boundary and $(Y^+)^5_{m\geq 2}$ are computed by the in-bulk algorithm (6.1). $(Y^+)^5_0$ is given by the b.c. The difficulty arises with the computation of $(Y^+)^5_1$. To find $(Y^+)^5_1$ with local accuracy $O(h^5)$, we need to compute the stage derivatives, and hence $Y^{\pm}$, at nodes $(-2, 2)$ and $(-1, 3)$ with local accuracy $O(h^4)$. $(Y^{\pm})^2_{-2}$ have already been computed above, and so the aforementioned difficulty occurs in the computation of $(Y^{\pm})^3_{-1}$. This node is shown by the open circle in Fig. 6.6.

Since, as we have noted, we intend to avoid using interpolation at levels $n \geq 3$, we must then compute the values $(Y^{\pm})^3_{-1}$ by a third-order scheme. Between the two available such schemes, MoC-cRK3 and MoC-pRK3, we choose the latter because of the instability that can potentially be introduced by the former scheme (see Sec. 5.9)[1]. We cannot find $(Y^{+})^3_{-1}$ using the regular MoC-pRK3 stencil, because it would require nodes with $m = -2, -3$. However, we can use the trick of "rotating" the stencil, as we did earlier to compute virtual nodes in the MoC-pRK3 with nonreflecting b.c. in Sec. 5.7. Then, by "rotating" the stencil in Fig. 6.6, we will obtain a "new" stencil, shown in Fig. 6.7.

Thus, our current goal is to compute $(Y^{\pm})^3_{-1}$ by a "rotated" version of the MoC-pRK3, shown in the right-pane of Fig. 6.7. According to Fig. 5.1, for this computation we need the solution at nodes $(1,1), (0,2), (0,4)$ (the filled-in circles in Fig. 6.7) and at node $(1,5)$ (the filled-in square). We have the solution at the first three nodes from previous time levels and the b.c. However, $(Y^{+})^5_1$ is precisely the solution we are trying to compute! Thus, to summarize the conundrum: to compute $(Y^{+})^5_1$ with MoC-pRK4, we need to first compute $(Y^{\pm})^3_{-1}$ by MoC-pRK3, which in turn requires the solution $(Y^{\pm})^5_1$. We seem to have run into a viscous circle.

The resolution to this conundrum comes from the observation that the computation of $(Y^{\pm})^3_{-1}$ by the MoC-pRK3 requires $(Y^{\pm})^5_1$ at a lower accuracy – only $O(h^3)$ – than the $O(h^5)$ accuracy which we need to achieve in the MoC-pRK4 computation of $(Y^{\pm})^5_1$. Thus, to compute $(Y^{\pm})^3_0$ by the rotated MoC-pRK3, *it will suffice to com-*

[1]Note parenthetically that if, regardless of the stability considerations, we had chosen to compute $(Y^{\pm})^3_{-1}$ by the MoC-cRK3, we would still have had to use this rotated stencil, because the MoC-cRK3 requires a solution to be available on the *left* of node $(-1,3)$. No such solution is available in the regular stencil (see Fig. 6.6), but it would become given by $(Y^{\pm})^2_{-1}$ in the rotated stencil in Fig. 6.7. This, as well as an analogous consideration for $(Y^{\pm})^3_{-2}$, shows that using the MoC-cRK3 to compute the solution at virtual nodes has no advantages over computing it with the MoC-pRK3.

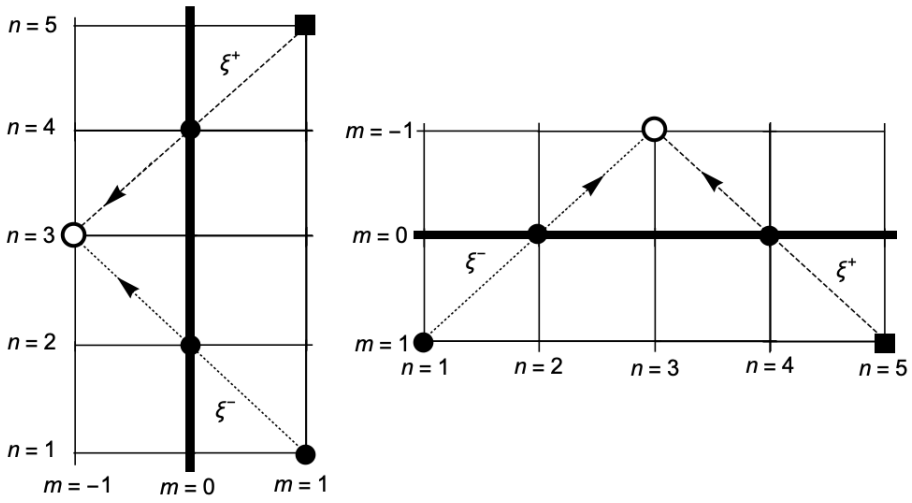

Figure 6.7: Left pane: second and third "columns" of the stencil in Fig. 6.6; right pane: the same substencil, rotated by −90 degrees. The characteristics ξ^+ (dashed) and ξ^- (dotted), along with the direction in which they propagate, are indicated in the stencil. The rotated stencil seems appropriate to compute the desired values at the node $(m = -1, n = 3)$ with the desired accuracy by the MoC-pRK3, with one caveat: the value at the node $(m = 1, n = 5)$ ***has yet to be computed*** *by the MoC-pRK4. The resolution to this will be explained in the text.*

pute $(Y^{\pm})_1^5$ *by the MoC-ME*, which we can do readily using the information already available.

Thus, the computation of $(Y^{\pm})_{-1}^3$ proceeds as follows. The second stage derivatives in the rotated MoC-pRK3 are given by (5.1{g,h}):

$$(\ell_2^+)_0^4 = f^+\left(\left(\overline{Y}^+\right)_{-1}^3, \left(\ddot{Y}^-\right)_{-1}^3\right), \tag{6.12a}$$

$$(\ell_2^-)_0^2 = f^-\left(\left(\ddot{Y}^+\right)_{-1}^3, \left(\overline{Y}^-\right)_{-1}^3\right), \tag{6.12b}$$

$$(\ell_2^+)_1^5 = f^+\left(\left(\overline{Y}^+\right)_{0}^4, \left(\ddot{Y}^-\right)_{0}^4\right), \tag{6.12c}$$

$$(\ell_2^-)_1^1 = f^-\left(\left(\ddot{Y}^+\right)_{0}^2, \left(\overline{Y}^-\right)_{0}^2\right), \tag{6.12d}$$

where the $\overline{Y}^{\pm}$ values are computed by the rotated MoC-SE (5.16{a,b}) and the $\ddot{Y}^{\pm}$

values have been computed with at least third-order local accuracy. Namely, $(\ddot{Y}^{\pm})^3_{-1}$ are computed by the rotated MoC-ME (5.16{c,d}), while $(\ddot{Y}^{-})^4_0$ has already been found by the MoC-pRK4 at $n = 4$, and $(\ddot{Y}^{+})^2_0$ is given by the b.c. The rotated MoC-pRK3 solution is then:

$$(Y^+)^3_{-1} = (Y^+)^4_0 - \frac{h}{12}\left(13(k_1^+)^4_0 + 5(\ell_2^+)^4_0 - (k_1^+)^5_1 - 5(\ell_2^+)^5_1\right), \tag{6.12e}$$

$$(Y^-)^3_{-1} = (Y^-)^2_0 + \frac{h}{12}\left(13(k_1^-)^2_0 + 5(\ell_2^-)^2_0 - (k_1^-)^1_1 - 5(\ell_2^-)^1_1\right), \tag{6.12f}$$

where the $k_1^{\pm}$ values are computed as usual by (6.2{a,b}). Note that we have multiplied h by negative 1 in the Y^+ solution to reflect the fact that the ξ^+ characteristic propagates in the rotated MoC-pRK3 stencil (Fig. 6.7) in the direction opposite to that in the bulk. This is similar to the situation in the rotated MoC-ME stencil in Fig. 5.9.

Similarly, near the right boundary, we need the value $(Y^{\pm})^3_{M+1}$ with local accuracy $O(h^4)$. In this case, the rotated MoC-pRK3 equations are:

$$(\ell_2^+)^2_M = f^+\left(\left(\overline{Y}^+\right)^3_{M+1}, \left(\ddot{Y}^-\right)^3_{M+1}\right), \tag{6.13a}$$

$$(\ell_2^-)^4_M = f^-\left(\left(\ddot{Y}^+\right)^3_{M+1}, \left(\overline{Y}^-\right)^3_{M+1}\right), \tag{6.13b}$$

$$(\ell_2^+)^1_{M-1} = f^+\left(\left(\overline{Y}^+\right)^2_{M}, \left(\ddot{Y}^-\right)^2_{M}\right), \tag{6.13c}$$

$$(\ell_2^-)^5_{M-1} = f^-\left(\left(\ddot{Y}^+\right)^4_{M}, \left(\overline{Y}^-\right)^4_{M}\right), \tag{6.13d}$$

$$(Y^+)^3_{M+1} = (Y^+)^2_M + \frac{h}{12}\left(13(k_1^+)^2_M + 5(\ell_2^+)^2_M - (k_1^+)^1_{M-1} - 5(\ell_2^+)^1_{M-1}\right), \tag{6.13e}$$

$$(Y^-)^3_{M+1} = (Y^-)^4_M - \frac{h}{12}\left(13(k_1^-)^4_M + 5(\ell_2^-)^4_M - (k_1^-)^5_{M-1} - 5(\ell_2^-)^5_{M-1}\right), \tag{6.13f}$$

where the $\overline{Y}^{\pm}$ values are computed by the rotated MoC-SE (5.17{a,b}), the $\ddot{Y}^{\pm}$ are

computed with local accuracy of at least $O(h^3)$, the $k_1^{\pm}$ are computed as usual by (6.2{a,b}), and the function evaluations on the r.h.s. of (6.13) are multiplied by $-h$ to reflect the fact that ξ^- propagates in the rotated MoC-pRK3 stencil in the direction opposite to that in the bulk.

With the points outside of the boundary computed with the desired accuracy, we can obtain the entire solution within the integration domain at $n = 5$. Observe that to compute $(Y^{\pm})^3_{-1,M+1}$, the highest level at which we needed the MoC-pRK4 solution was $n = 4$. Therefore, once the solution at level n is obtained, the next step is to compute the solution at $(Y^{\pm})^{n-1}_{-1,M+1}$, as this will be needed to advance to level $n + 1$.

We now explain how to find the solution at $n = 6$. The calculation at the remaining steps will mirror the one presented here. The stencil for this calculation is shown in Fig. 6.8.

We see that the only node where we need a solution which we do not yet have is $(m, n) = (-2, 3)$. We only need these values with local accuracy $O(h^4)$, so we can use the rotated MoC-pRK3, as we have the appropriate points to use this method. The remaining steps will follow a similar pattern. Once the solution is found at level n, first compute the values at $(Y^{\pm})^{n-1}_{-1,M+1}$, by the rotated MoC-pRK3, then compute the values $(Y^{\pm})^{n+1}_{m=0,\ldots,M}$ by the MoC-pRK4. Next, compute $(Y^{\pm})^{n}_{-1,M+1}$, and finally compute the values at $(Y^{\pm})^{n-2}_{-2,M+2}$ by the same procedure as used to compute $(Y^{\pm})^{n-1}_{-1,M+1}$.

The pseudocode for the MoC-pRK4 with nonreflecting b.c. is presented in Algorithm 2. In Appendix F, we will include our commented Matlab implementation of the method.

Algorithm 2: MoC-pRK4 algorithm with nonreflecting boundary conditions

while $n < nmax$ **do**
  if $n < 4$ **then**
    if $n = 1$ **then**
      $(Y^{\pm})^{1}_{[0,M]}$ given by initial condition;
      Extrapolate $(Y^{\pm})^{1}_{-2,-1,M+1,M+2}$ by (6.11);
    else if $n = 2$ **then**
      Compute $(Y^{\pm})^{2}_{[0,M]}$ by MoC-cRK3;
      Extrapolate $(Y^{\pm})^{2}_{-2,-1,M+1,M+2}$ by (6.11);
    else
      Compute $(Y^{\pm})^{3}_{[0,M]}$ by MoC-cRK3;
    end
  else
    Compute $(\ddot{Y}^{\pm})^{n}_{[0,M]}$ by MoC-pRK3.
      As a byproduct, we obtain: $(k_1^{\pm})^{n-1}_{[0,M]}$ and the MoC-SE solution $(\overline{Y}^{\pm})^{n}_{[0,M]}$;
    if $n > 4$ **then**
      Compute $(Y^{\pm})^{n-2}_{-1,M+1}$ by rotated MoC-pRK3;
      if $n > 5$ **then**
        Compute $(Y^{\pm})^{n-3}_{-2,M+2}$ by rotated MoC-pRK3;
      end
    end
    if $n = 4$ **then**
      Compute $(k^{+}_{1,2})^{1}_{[-2,M-3]}$ and $(k^{-}_{1,2})^{1}_{[3,M+2]}$ by (6.2{a,b,k,l});
      Compute $(k^{+}_{1,2})^{2}_{[-1,M-2]}$ and $(k^{-}_{1,2})^{2}_{[2,M+1]}$ by (6.2{a,b,k,l});
    else
      Set $(k_1^{+})^{n-3}_{[-2,M-3]} = \left[f^{+}((Y^{+})^{n-3}_{-2}, (Y^{-})^{n-3}_{-2}), (k_1^{+})^{n-3}_{[-1,M-3]}\right]$;
      Set $(k_1^{-})^{n-3}_{[3,M+2]} = \left[(k_1^{-})^{n-3}_{[3,M+1]}, f^{-}((Y^{+})^{n-3}_{M+2}, (Y^{-})^{n-3}_{M+2})\right]$;
      Compute $(\overline{Y}^{+})^{n-2}_{-1}$ and $(\overline{Y}^{-})^{n-2}_{M+1}$ by MoC-SE using $(Y^{\pm})^{n-3}_{-2,M+2}$;
      Set $(k_2^{+})^{n-3}_{[-2,M-3]} = \left[f^{+}((\overline{Y}^{+})^{n-2}_{-1}, (Y^{-})^{n-2}_{-1}), (k_2^{+})^{n-3}_{[-1,M-3]}\right]$;
      Set $(k_2^{-})^{n-3}_{[3,M+2]} = \left[(k_2^{-})^{n-3}_{[3,M+1]}, f^{-}((Y^{+})^{n-2}_{M+1}, (\overline{Y}^{-})^{n-2}_{M+1})\right]$;
      Compute $(k^{+}_{1,2})^{n-2}_{[-1,M-2]}$ and $(k^{-}_{1,2})^{n-2}_{[2,M+1]}$ as in Lines 24-28 using $(Y^{\pm})^{n-2}_{-1,M+1}$;
    end
    Set $(k_2^{+})^{n-1}_{[0,M-1]} = f^{+}\left((\overline{Y}^{+})^{n}_{[1,M]}, (\ddot{Y}^{-})^{n}_{[1,M]}\right)$;
    Set $(k_2^{-})^{n-1}_{[1,M]} = f^{-}\left((\ddot{Y}^{+})^{n}_{[0,M-1]}, (\overline{Y}^{-})^{n}_{[0,M-1]}\right)$;
    Compute $(Y^{\pm})^{n}_{[0,M]}$ by MoC-pRK4.
      Note: $(k_1^{\pm})^{n-1}$ were found when computing $(\ddot{Y}^{\pm})^{n}$ by MoC-pRK3 ;
  end
end

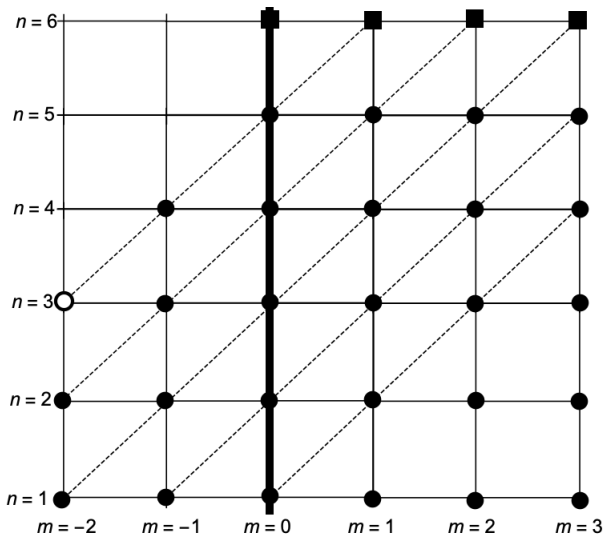

Figure 6.8: Stencil to advance the Y^+ solution from $n = 5$ to $n = 6$ using the MoC-pRK4. The only node where we need values but do not have them is $(m, n) = (-2, 3)$. All of the other available values were computed in previous steps, and the value at $(-1, 4)$ was computed by the rotated MoC-pRK3 after the solution at $n = 5$ was found.

6.6 Numerical results of the MoC-pRK4 with nonreflecting b.c.

In Fig. 6.9, we show the results of solving system (1.50) with the MoC-pRK4 using nonreflecting b.c. We see that nonreflecing b.c. iron out the instability present in the scheme with periodic b.c., similar to the behavior witnessed in the MoC-ME and MoC-pRK3. Thus, we have a stable, fourth-order method of characteristics.

In Chapter 7, we present the numerical results verifying that the MoC-pRK3 and MoC-pRK4 are third- and fourth-order methods, respectively. We also verify that they preserve conserved quantities with order 3 and order 5, respectively, as predicted in Chapter 4.

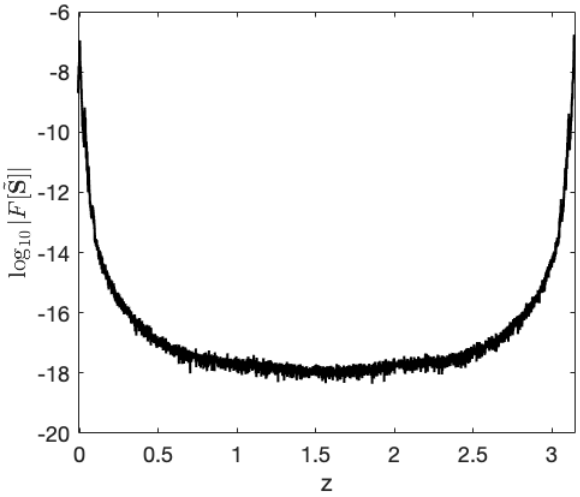

Figure 6.9: Logarithm of the Fourier spectrum of the error obtained by solving (1.50) with the MoC-pRK4 using nonreflecting b.c. up to $t = 500$ with $h = 0.01$. One can see that the error decays here, whereas in the case of periodic b.c. the error grows. Unlike the MoC-ME and MoC-pRK3 with nonreflecting b.c., there is not a noticeable "beak" forming in the middle of the spectrum.

7 Numerical results using the MoC-pRK3 and MoC-pRK4

In this chapter, we present some numerical results which verify that the MoC-pRK3 and MoC-pRK4 are third- and fourth-order accurate schemes, respectively. We also verify that these schemes preserve conserved quantities to the degree that their ODE counterparts do. That is, the preservation of conserved quantities should be third-order in the MoC-pRK3 and fifth-order in the MoC-pRK4.

The model problem that we use to obtain our results in this chapter is the massive Gross–Neveu model, first mentioned in Sec. 1.6:

$$S_t^+ + S_x^+ = i(|S^-|^2 S^+ + (S^-)^2 (S^+)^*) - iS^-, \tag{1.51a}$$

$$S_t^- - S_x^- = i(|S^+|^2 S^- + (S^+)^2 (S^-)^*) - iS^+. \tag{1.51b}$$

The conserved quantities admitted by system (1.51) are Hamiltonian, H, and charge, Q, given by:

$$H = \int \Big[- i((S^+)^* S_x^+ - (S^-)^* S_x^-) - \frac{1}{2}(S^+(S^-)^* + (S^+)^* S^-)^2 + (S^+(S^-)^* + (S^+)^* S^-) \Big] \, dx, \tag{1.52}$$

$$Q = \int \left(|S^+|^2 + |S^-|^2 \right) \, dx. \tag{1.53}$$

We will present results which solve system (1.51) using periodic boundary conditions, then nonreflecting boundary conditions. We will examine the error obtained in each

computation as well as the change in H and Q.

7.1 Results using periodic b.c.

When solving system (1.51) with periodic b.c., we impose the initial condition in the form of a standing (i.e., non-moving) soliton:

$$S^+ = \frac{u_0 + v_0}{\sqrt{2}}, \quad S^- = \frac{u_0 - v_0}{\sqrt{2}}, \tag{7.1a}$$

where

$$u_0(x) = \frac{\sqrt{2(1-\omega)}\cosh(\beta x)}{\cosh^2(\beta x) - \mu^2 \sinh^2(\beta x)}, \quad v_0(x) = i\mu \tanh(x) u_0(x), \tag{7.1b}$$

and

$$\beta = \sqrt{1-\omega^2}, \quad \mu = \sqrt{(1-\omega)/(1+\omega)}. \tag{7.1c}$$

The argument x in (7.1b) refers to a point in the spatial grid, and the frequency ω is constant and is set by the user. u_0 and iv_0 are plotted in Fig. 7.1 for different values of ω.

In the results that follow, we set the length of the spatial grid to be $L = 128$ with center at the origin, i.e. $x \in [-L/2, L/2]$. The fineness of the grid is determined by M, from which $h = L/M$.

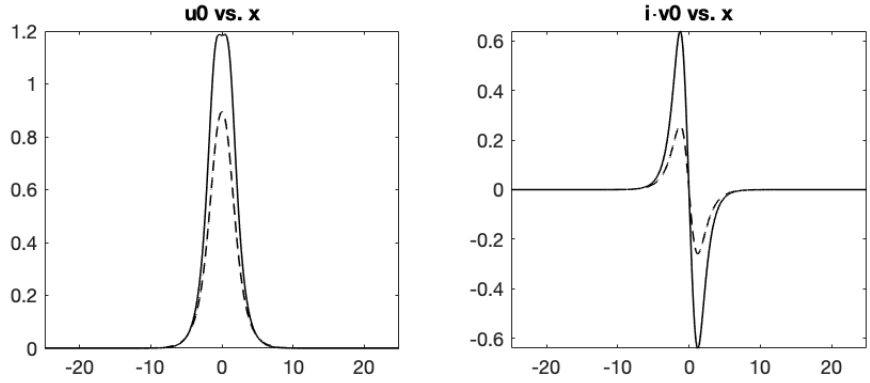

Figure 7.1: u_0 (left pane) and iv_0 (right pane) for $\omega = 0.3$ (solid) and $\omega = 0.6$ (dashed).

Table 7.1: The log-10 error obtained from solving system (1.51) with the MoC-pRK3 up to $t = 200$ using periodic b.c. with different values of M and ω.

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	-4.3	-5.2	-6.1	-7.0
0.6	-5.1	-6.0	-6.9	-7.7

7.1.1 Scaling of the error and conserved quantities (long-term)

In Tables 7.1 and 7.2, we show the numerical results which verify the order of accuracy of the MoC-pRK3 and MoC-pRK4, respectively.

In Tables 7.3 and 7.4, we show the numerical results which verify that the Hamiltonian is preserved by a factor of 3 in with the MoC-pRK3 and a factor of 5 with the MoC-pRK4. In Tables 7.5 and 7.6, we verify the same result for the charge. These simulations were run until $t = 200$ to show that the Hamiltonian and charge are preserved with the desired scaling after long-time simulations. In a later subsection, we will run the simulations until $t = 5$ for many values of h to demonstrate the proper scaling of the error.

Let us briefly explain how these tables are interpreted. By increasing M by a factor of 2, we are halving h, since $h = L/M$. Thus, when $h_1/h_2 = 2$, a third-order method, for example, should have $\text{error}_1/\text{error}_2 = 2^3$. In the log-10 scale, which is the scale our tables are in, this means we should have $\log_{10}\text{error}_1 - \log_{10}\text{error}_2 = 3\log_2 \approx 0.9$. For fourth-order scaling, this number is $4\log_{10} 2 \approx 1.2$, and for fifth-order scaling, $5\log_{10} 2 \approx 1.5$.

Table 7.2: Same description as Table 7.1 but with the MoC-pRK4.

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	−5.5	−6.7	−7.9	−9.1
0.6	−6.0	−7.3	−8.5	−9.7

Table 7.3: The log-10 change in the Hamiltonian, H, that arises from solving system (1.51) with the MoC-pRK3 up to $t = 200$ using periodic b.c. with different values of M and ω.

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	−2.4	−3.3	−4.2	−5.1
0.6	−2.5	−3.4	−4.3	−5.2

Table 7.4: Same description as Table 7.3 but with the MoC-pRK4

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	−4.6	−6.1	−7.6	−9.1
0.6	−5.1	−6.6	−8.1	−9.7

7.1.2 Comparison of MoC-pRK and MoC-NpRK methods

Our results show that the MoC-NpRK methods may have an advantage over the MoC-pRK methods when it comes to preserving the conserved quantities. As discussed in

Table 7.5: Change in the charge, Q, that arises from solving system (1.51) with the MoC-pRK3 up to $t = 200$ using periodic b.c. with different values of M and ω.

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	-2.1	-3.0	-3.9	-4.8
0.6	-2.3	-3.2	-4.1	-5.0

Table 7.6: Same description as Table 7.5 but with the MoC-pRK4

ω \ M	2^{11}	2^{12}	2^{13}	2^{14}
0.3	-4.3	-5.8	-7.3	-8.8
0.6	-5.0	-6.5	-8.0	-9.5

Chapter 4, the NpRK methods each have at least one free parameter. In Appendix D, we discuss how the selection of these free parameters can affect the conservation laws admitted by a problem. Figure 7.2 shows the change in the Hamiltonian and the charge obtained by using the MoC-NpRK3 for different values of the free parameter a_{20}, and Fig. 7.3 shows the same for the MoC-NpRK4 and the free parameter c_{22}.

We can therefore draw the conclusion that for certain values of the free parameter in MoC-NpRK methods, the conserved quantities may be preserved better.

7.1.3 Scaling of the error (short-term)

In this subsection, we verify that the error in the third- and fourth-order MoC schemes scales properly in the short-term. The motivation for doing this is twofold: First, since we use many small values of h, long simulations are not time efficient. Second, when we consider a moving pulse in the next subsection, the computation requires us to track the center and frequency of the pulse. This becomes rather complicated and can lead to a growth in the error in the long-term which is *not* a byproduct of the

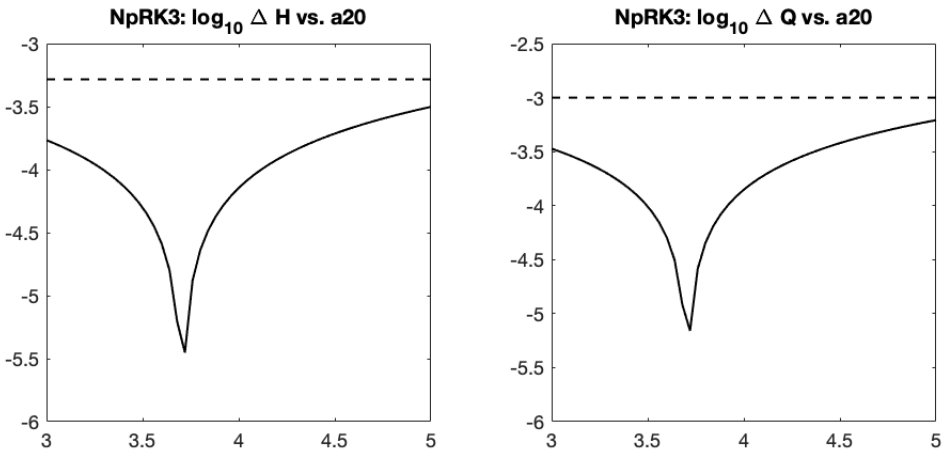

Figure 7.2: Results from solving system (1.51) with the MoC-NpRK3 up to $t = 200$ using $M = 2^{12}$, $\omega = 0.3$, and different values of a_{20}. Left pane: log-10 of the change in H vs. a_{20}. Right pane: log-10 of the change in Q vs. a_{20}. In each pane, the dotted line shows the value of ΔH or ΔQ from the MoC-pRK3 (see Tables 7.3 and 7.5). The optimal value of a_{20} in each plot is $a_{20} \approx 3.72 \approx e + 1$. We see that for this optimal value, H and Q are preserved by at least two orders of magnitude better in the MoC-NpRK3 than they are in the MoC-pRK3.

numerical method.

To examine the error in the short-term, we compute the logarithm of the error obtained solving system (1.51) with various third- and fourth-order MoC schemes for 11 values of M spaced evenly between $M = 2^{11}$ and $M = 2^{16}$ (i.e. $h \approx 0.06$ and $h \approx 0.002$) up to $t = 5$. Results for third-order schemes are shown in Fig. 7.4, and results for fourth-order schemes are shown in Fig. 7.5.

Our results also show that the MoC-pRK and MoC-NpRK methods produce very similar numerical error, regardless of the free parameters in the MoC-NpRK methods. Since we will not consider H and Q when we use nonreflecting b.c., we will omit results concering the MoC-NpRK methods using nonreflecting b.c.

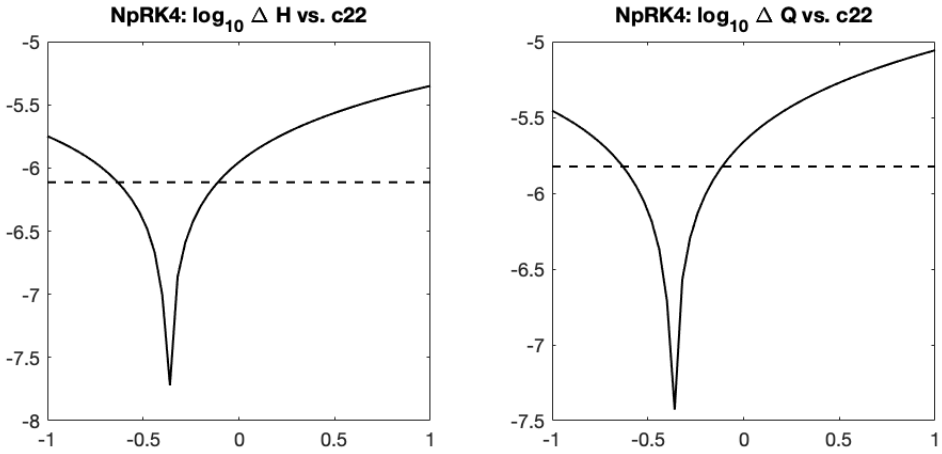

Figure 7.3: Results from solving system (1.51) with the MoC-NpRK4 up to $t = 200$ using $M = 2^{12}$, $\omega = 0.3$, and different values of c_{22}. Left pane: log-10 of the change in H vs. c_{22}. Right pane: log-10 of the change in Q vs. c_{22}. In each pane, the dotted line shows the value of ΔH or ΔQ from the MoC-pRK4 (see Tables 7.4 and 7.6). The optimal value of c_{22} in each plot is $c_{22} \approx -0.36$. We see that for this optimal value, H and Q are preserved by at least an order of magnitude better in the MoC-NpRK4 than they are in the MoC-pRK4.

7.2 Results using nonreflecting b.c.

We now present the numerical results obtained from using the MoC-pRK methods to solve (1.51) with nonreflecting b.c. To test these methods with nonreflecting b.c., we solve (1.51) in the case of a moving soliton. As described in [13], the equations of a soliton moving with velocity $V \in (-1, 1)$ are given by:

$$\begin{pmatrix} u_{\text{mov}}(x,t) \\ v_{\text{mov}}(x,t) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{\Gamma+1} & \sqrt{\Gamma-1} \\ \sqrt{\Gamma-1} & \sqrt{\Gamma+1} \end{pmatrix} \begin{pmatrix} u(x_{\text{mov}}, t_{\text{mov}}) \\ v(x_{\text{mov}}, t_{\text{mov}}) \end{pmatrix} \tag{7.2a}$$

where

$$\Gamma = 1/\sqrt{1-V^2}, \quad x_{\text{mov}} = \Gamma(x - Vt), \quad t_{\text{mov}} = \Gamma(t - Vx). \tag{7.2b}$$

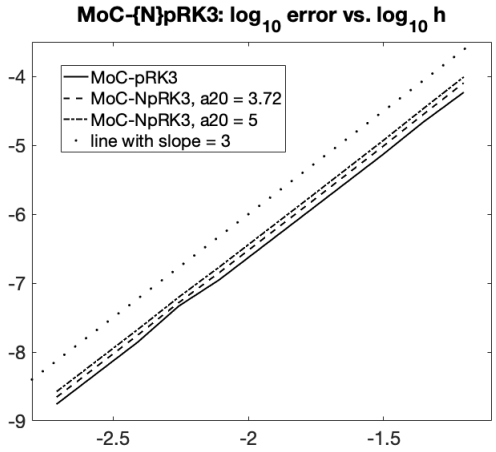

Figure 7.4: Log-10 of the error vs. h obtained by solving system (1.51) with the MoC-pRK3 (solid), MoC-NpRK3 with $a_{20} = 3.72$ (dashed), and the MoC-NpRK3 with $a_{20} = 5$ (dot-dashed) up to $t = 5$ with $\omega = 0.3$ and periodic b.c. The dotted line is a line of slope 3 and is included for reference to show that each of the error vs. h curves have slope 3. This shows that the error in these method scales as h^3. We show the results for two MoC-NpKk3 methods to illustrate that the free parameter a_{20} has very little effect on the error, as opposed to the effect it has on the Hamiltonian and charge.

The initial shape of the moving soliton is shown in Fig. 7.6. Numerical results verifying the proper scaling of the MoC-pRK3 with nonreflecting b.c. are shown in Table 7.7 and Fig. 7.6.

Table 7.7: The log-10 error obtained from solving system (1.51) with the MoC-pRK3 and MoC-pRK4 up to $t = 5$ using nonreflecting b.c. with different values of M and $\omega = 0.3$. For reasons explained in Sec. 7.1, this shows that the error scales as h^3 in the MoC-pRK3 and as h^4 in the MoC-pRK4.

M	2^{11}	2^{12}	2^{13}	2^{14}
MoC-pRK3	-3.8	-4.7	-5.6	-6.5
MoC-pRK4	-4.7	-6.0	-7.1	-8.4

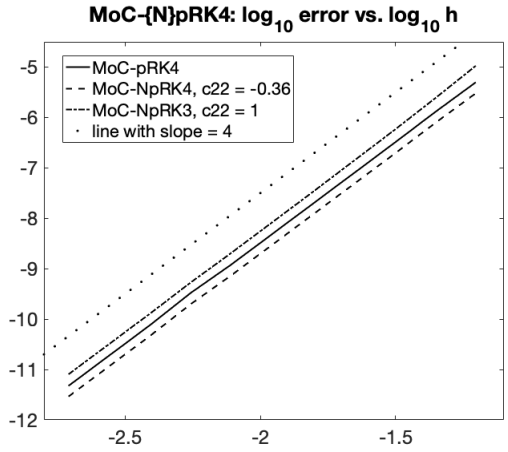

Figure 7.5: Log-10 of the error vs. h obtained by solving system (1.51) with the MoC-pRK4 (solid), MoC-NpRK4 with $c_{22} = -0.36$ (dashed), and the MoC-NpRK4 with $c_{22} = 1$ (dot-dashed) up to $t = 5$ with $\omega = 0.3$ and periodic b.c. The dotted line is a line of slope 4 and is included for reference to show that each of the error vs. h curves have slope 4. This shows that the error in these method scales as h^4. We show the results for two MoC-NpRK4 methods to illustrate that the free parameter c_{22} has very little effect on the error, as opposed to the effect it has on the Hamiltonian and charge.

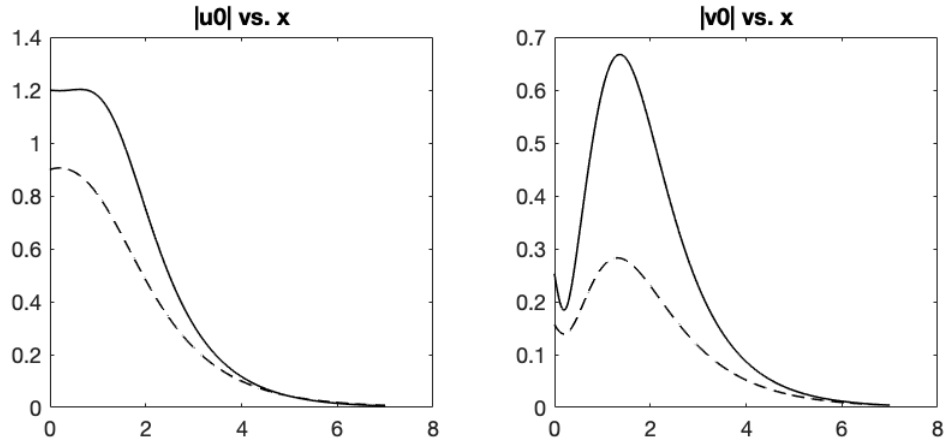

Figure 7.6: Initial shape of each component of the soliton (7.1) now moving according to (7.2). The solid curves show the soliton with $\omega = 0.3$, and the dashed curves show the soliton with $\omega = 0.6$.

7.3 Concluding remarks

In Chapter 1 of this thesis, it was stated that the objective of this thesis was to develop explicit MoC schemes which are of an order greater than two. After showing

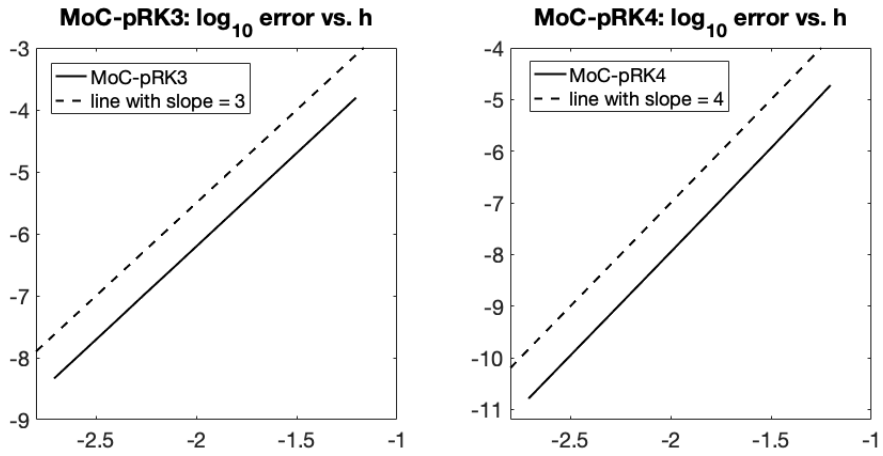

Figure 7.7: Log-10 of the error vs. h obtained by solving system (1.51) with the MoC-pRK3 (left pane) and MoC-pRK4 (right pane) up to $t = 5$ with $\omega = 0.3$ and nonreflecting b.c. The dashed line in each pane is included for reference to show that the error in each method has the desired scaling.

that MoC schemes using the third- and fourth-order RK methods are unstable in Chapter 3, we studied the pRK methods in Chapter 4, then developed MoC schemes using the pRK3 and pRK4 in Chapters 5 and 6. It was shown in these chapters that the MoC-pRK3 and MoC-pRK4 can have only a mild instability when periodic b.c. are used, and no instability at all in the case of nonreflecting b.c. In this chapter, we verified numerically that the MoC-pRK3 and MoC-pRK4 are third- and fourth-order methods. Therefore, the main objective of this thesis has been completed.

8 Conclusion

We will now summarize the main results of this thesis.

In Chapter 1, we explained that only first- and second-order accurate explicit MoC schemes for the solution of hyperbolic systems of PDEs with straight-line and crossing characteristics have been reported. This fact motivated the work undertaken in the thesis: to develop higher-order and numerically stable explict MoC schemes.

Before attempting to develop higher-order methods, in Chapter 2 we gave a detailed exposition of how one constructs an MoC scheme using explicit RK methods. We also described the procedure of the von Neumann stability analysis for these schemes, which showed that the MoC-SE, MoC-ME, and MoC-MP all have a growth rate in the error that is $O(h)$.

Our first approach in achieving the goal of developing higher-order methods was in Chapter 3. We gave many examples of how one can construct a third-order MoC scheme using third-order RK methods. However, the von Neumann stability analysis predicted, and the numerical simulations verified, that the MoC schemes using third-order RK methods can be strongly unstable, with an $O(1)$ growth rate in the error. We showed that the MoC using the fourth-order RK method has the same unpleasant stability properties of the aforementioned third-order schemes. These third- and fourth-order schemes all required the approximation of the solution at nodes which were not on the stencil. Accepting the hypothesis that the off-stencil points lead to a strong instability, we chose to seek higher-order ODE methods whose MoC counterparts would not need to approximate off-stencil values.

This brought us to the pRK methods, explored in Chapter 4. There, we compared

the traditional pRK methods and Nakashima pRK methods for ODEs. The main benefit of the pRK methods is that when used in the MoC for hyperbolic PDEs, the schemes do not require any off-stencil values. In Chapter 5, we described how one constructs a third-order MoC scheme using third-order pRK methods. We showed that the MoC-pRK3 results in only a mild instability with the growth rate in the error of $O(h)$. Although the MoC-pRK3 and already known MoC-ME both have the $O(h)$ instability, the growth constant in the MoC-pRK3 is three times smaller than that of the MoC-ME. In the first part of Chapter 6, we constructed the MoC-pRK4 and showed that it has the mild $O(h)$ instability as well.

In the latter parts of Chapters 5 and 6, we explained that the instability seen in the MoC-pRK3 and MoC-pRK4 using periodic boundary conditions vanishes when nonreflecting b.c. are used instead. Because of the multistep nature of the pRK methods, the implementation of nonreflecting b.c. becomes quite complicated. We therefore provided a detailed account of how to overcome the complications which arise when nonreflecting b.c. are used, and we proposed an algorithmic method for how they can be implemented.

Finally, in Chapter 7, we presented numerical results using the MoC-pRK3 and MoC-pRK4. These results verified that the error obtained in the MoC-pRK3 scales as h^3, and the error obtained in the MoC-pRK4 scales as h^4. These results were consistent using both periodic and nonreflecting b.c.

Other than the order of the error, another important advantage that some of the third- and fourth-order MoC-pRK methods provide over the second-order MoC-ME concerns the preservation of conserved quantities, such as the total mass or the Hamiltonian. In the MoC-ME, these quantities drift (i.e. increase or decrease

linearly) in time with a rate $O(h^3)$. In the third-order MoC-pRK, the drift rate scales similarly, i.e. as $O(h^3)$. However, in the fourth-order MoC-pRK, the drift rate drops substantially, to $O(h^5)$. Moreover, the constant in front of $O(h^n)$, $n = 3$ or 5, in the drift rate can be increased by one to two orders of magnitude by choosing an appropriate Nakashima "flavor" of the pRK instead of the traditional pRK. This improvement of the preservation of conserved quantities by Nakashima pRK methods over traditional ones was reported and verified for both ODEs (Chapter 4) and hyperbolic PDEs (Chapter 7).

As we have mentioned, in this thesis we only considered explicit MoC schemes, as opposed to implicit schemes. The most popular "implicit" (see below regarding the quotes) scheme appears to be a fourth-order scheme described in [3]. While the ODE solver for this scheme can be formulated as an implicit RK method, in practice it is implemented as a predictor-corrector method. The correction step can be applied an arbitrary number k of times, and in the limit $k \to \infty$ (but, in practice, already for $k = 3$) the method approaches its implicit "ancestor". Hence the quotes in "implicit". A natural question then arises: do implicit (or "implicit") methods resolve the instability seen in explicit methods? In work which we are currently undertaking, the answer appears to be "yes" for the implicit MoC-ME, and "no" for the implicit fourth-order MoC using periodic boundary conditions.

Even if nonreflecting b.c. are found to be able to vanquish the instability in the implicit fourth-order MoC, one may wonder if the MoC-pRK4 is still a better choice. The main advantage of the MoC-pRK4 appears to be computational efficiency, since the "implicit" method requires iterations. The MoC-pRK4 requires 4 function evaluations per characteristic per step, while the "implicit" method requires $3 + 2k$

function evaluations per characteristic per step, where k is the number of iterations. For the commonly used value $k = 3$, the MoC-pRK4 is expected to run about twice as fast as the MoC based on the "implicit" RK method. A more thorough investigation of the implicit MoC schemes is a work in progress.

Bibliography

[1] T. I. Lakoba and Z. Deng. Stability analysis of the numerical Method of characteristics applied to energy-preserving systems. Part I: Periodic boundary conditions. *J. Comp. Appl. Math*, (to be published), 2019.

[2] J. D. Hoffman. *Numerical Methods for Engineers and Scientists*, chapter 10, pages 458–466. McGraw-Hill, 1992.

[3] C. M. de Sterke, K. R. Jackson, and B. D. Robert. Nonlinear coupled-mode equations on a finite interval: a numerical procedure. *J. Opt. Soc. Am. B*, 8(2):403–412, Feb 1991.

[4] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.

[5] S. Pitois, G. Millot, and S. Wabnitz. Nonlinear polarization dynamics of counterpropagating waves in an isotropic optical fiber: theory and experiments. *J. Opt. Soc. Am. B*, 18(4):432–443, Apr 2001.

[6] T. I. Lakoba. Numerical study of solitary wave stability in cubic nonlinear dirac equations in 1d. *Physics Letters A*, 382(5):300 – 308, 2018.

[7] M. K. Jain. *Numerical Solution of Differential Equations*, chapter 2, pages 32–36. Halsted Press, 1984.

[8] G. D. Byrne and R. J. Lambert. Pseudo-Runge–Kutta methods involving two points. *J. Assoc. Comput. Mach.*, 13:114–123, 1966.

[9] M. Nakashima. On pseudo-Runge–Kutta methods with 2 and 3 stages. *Publ. RIMS, Kyoto Univ.*, 18:895–909, 1982.

[10] L. T. Hwee. A third order Nakashima pseudo-Runge–Kutta method. *Sunway Academic Journal*, 10:36–45, 2014.

[11] H. Shintani. On pseudo-Runge–Kutta methods of the third kind. *Hiroshima Math. J.*, 11:247–254, 1981.

[12] T. I. Lakoba and Z. Deng. Stability analysis of the numerical method of characteristics applied to energy-preserving systems. Part II: Nonreflecting boundary conditions. *J. Comp. Appl. Math*, (to be published), 2019.

[13] T. I. Lakoba. Study of (in)stability of the Fourier split-step method for the massive Gross–Neveu model. In preparation, 2019.

[14] T. D. Sauer. *Numerical Analysis*, chapter 3, pages 140–143. Pearson, 2006.

A Derivation of the Lax–Wendroff Method

In this Appendix, we derive the Lax–Wendroff method which is used used to solve the one-dimensional hyperbolic convection equation,

$$u_t + cu_x = 0. \tag{A.1}$$

Before we do so, we will need approximations of the first and second spatial derivatives.

A.1 Second-order centered-difference approximations of the first and second spatial derivatives

Suppose $y(x,t) \equiv y$ is discretized over a two-dimensional grid, where y_m^n denotes the solution at grid point (m,n). Here we derive expressions for $y_x|_m^n$ and $y_{xx}|_m^n$. One can derive expressions for the time derivatives in a similar manner.

Begin by expanding y_{m+1}^n and y_{m-1}^n in a Taylor series in the x-variable around y_m^n:

$$y_{m+1}^n = y_m^n + y_x|_m^n \Delta x + \frac{1}{2} y_{xx}|_m^n \Delta x^2 + O(\Delta x^3), \tag{A.2a}$$

$$y_{m-1}^n = y_m^n - y_x|_m^n \Delta x + \frac{1}{2} y_{xx}|_m^n \Delta x^2 + O(\Delta x^3). \tag{A.2b}$$

Subtracting (A.2b) from (A.2a), and solving for $y_x|_m^n$, yields:

$$y_x|_m^n = \frac{y_{m+1}^n - y_{m-1}^n}{2\Delta x} + O(\Delta x^2), \tag{A.3}$$

the second-order approximation of the first derivative. By expanding (A.2a) and (A.2b) to be fourth-order accurate, then adding the results, one arrives at:

$$y_{xx}|_m^n = \frac{y_{m+1}^n - 2y_m^n + y_{m-1}^n}{\Delta x^2} + O(\Delta x^2), \tag{A.4}$$

the second-order approximation of the second derivative.

A.2 Lax–Wendroff Method

We can now derive the Lax–Wendroff Method. We begin by expanding (A.1) in a Taylor series in time about the grid point (m, n):

$$u_m^{n+1} = u_m^n + u_t|_m^n \Delta t + \frac{1}{2} u_{tt}|_m^n \Delta t^2 + O(\Delta t^3). \tag{A.5}$$

Differentiating (A.1) with respect to time, we obtain:

$$u_{tt} = (-cu_x)_t = -c(u_t)_x = -c(-cu_x)_x = c^2 u_{xx}. \tag{A.6}$$

Substituting (A.1) and (A.6) into (A.5) yields:

$$u_m^{n+1} = u_m^n - cu_x|_m^n \Delta t + \frac{1}{2} c^2 u_{xx}|_m^n \Delta t^2 + O(\Delta t^3). \tag{A.7}$$

Substituting the second-order centered-difference approximations for $u_x|_m^n$ and $u_{xx}|_m^n$ ((A.3) and (A.4)) gives us:

$$u_m^{n+1} = u_m^n - c\left(\frac{u_{m+1}^n - u_{m-1}^n}{2\Delta x} + O(\Delta x^2)\right)\Delta t + \frac{1}{2}c^2\left(\frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{\Delta x^2} + O(\Delta x^2)\right)\Delta t^2 + O(\Delta t^3). \tag{A.8}$$

Simplifying and dropping the truncation error term yields:

$$u_m^{n+1} = u_m^n - \frac{\gamma}{2}(u_{m+1}^n - u_{m-1}^n) + \frac{\gamma^2}{2}(u_{m+1}^n - 2u_m^n + u_{m-1}^n), \tag{A.9a}$$

where in (A.9a)

$$\gamma = \frac{c\Delta t}{\Delta x}. \tag{A.9b}$$

Equations (A.9) are the Lax–Wendroff Method. Note that (A.8) implies that its discretization error is $O(\Delta x^2 \Delta t + \Delta t^3)$.

B Interpretation of results obtained in the von Neumann stability analysis and numerical verification

B.1 Growth rate of the error in the numerical solution predicted by the von Neumann analysis

Recall in the von Neumann analysis (Sec. 2.3), one obtains a stability matrix, $\mathbf{\Phi}(z)$, where $z = kh$. One can then recover information about the stability of the numerical method by plotting the maximum eigenvalue λ of $\mathbf{\Phi}(z)$ for $z \in [0, \pi)$, since the magnitude of initially small perturbations in the numerical solution at step n will be proportional to $|\lambda|^n$. When this is done for the MoC-SE, MoC-ME, and MoC-MP, one obtains:

Examining the y-axis of each plot, one can conclude that the maximum (in modulus) eigenvalue satisfies:

$$|\lambda| = 1 + O(h^2) \tag{B.1}$$

for each method. This follows from the fact that the magnitude of the largest eigenvalue is the same for a given method no matter what h is. In Fig. B.1 we show results only for two values of h to illustrate this point. Equation (B.1) can be written

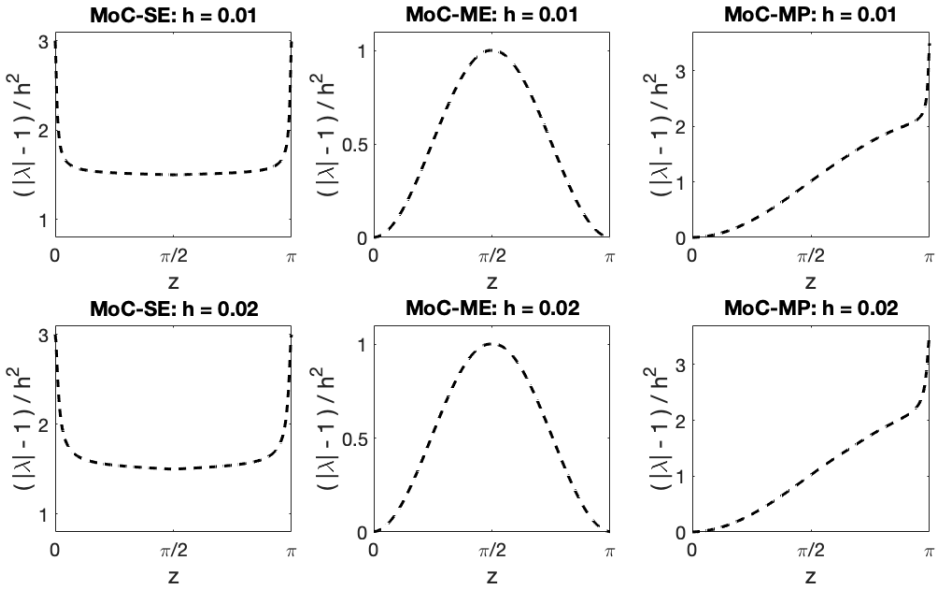

Figure B.1: von Neumann stability plots for the MoC-SE, MoC-ME, and MoC-MP. The reason why we show essentially identical plots for two different values of h is explained in the text.

equivalently as:

$$|\lambda| = 1 + ch^2, \tag{B.2}$$

where c depends on z (i.e., the wavenumber), but on on h. Therefore, the size of the perturbations at step n is:

$$|\lambda|^n = |\lambda|^{t/h} = (1 + ch^2)^{t/h} \approx e^{(ch)t}, \tag{B.3}$$

meaning that the growth rate in the error of the solution is $ch = O(h)$.

B.2 Numerical verification of the von Neumann analysis

Here we explain how the numerical simulation we perform verifies the results of the von Neumann analysis. To begin the simulation, we add to the initial condition a small white noise, whose Fourier spectrum is uniform (see Fig. B.2).

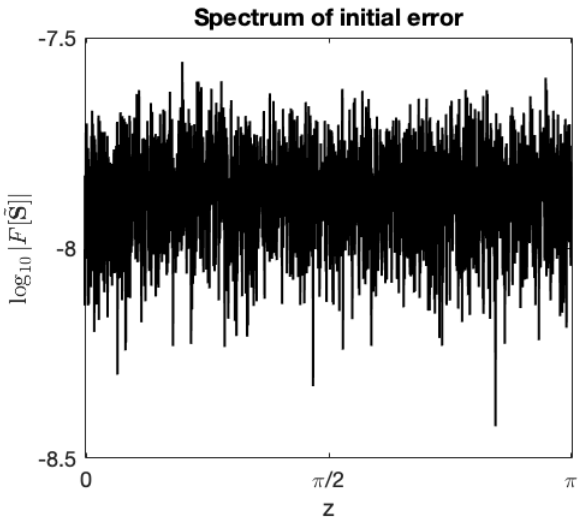

Figure B.2: Fourier spectrum of initial white noise

This noise acts as the small initial perturbation to the solution. Let W_n represent the magnitude of the noise in the spectrum at time level n for some wavenumber k. Then, according to (B.3),

$$W_n = W_0 e^{cht}, \tag{B.4}$$

where the value c may depend on k. Taking the logarithm of (B.4) yields:

$$\log_{10}(W_n) = cht + \log_{10} W_0, \tag{B.5}$$

which by (B.2) means:

$$\begin{aligned} \log_{10}(W_n) &= \left(\frac{|\lambda| - 1}{h^2}\right) ht + \log_{10} W_0 \\ &\equiv \left(\frac{|\lambda| - 1}{h^2}\right) \text{const}_1 + \text{const}_2, \end{aligned} \tag{B.6}$$

where $\text{const}_{1,2}$ do not depend on k. The only dependence on k on the r.h.s. of B.6 comes from $|\lambda|$. Therefore, the shape of the spectrum on the log scale is the same as that which we see in the von Neumann stability plot.

B.3 Prediction and interpretation of numerical results

Given the numerical data used to create Fig. B.1, one can solve (B.2) for c. For the MoC-SE and MoC-ME (with $h = 0.02$ or 0.01), one finds:

$$c_{\text{SE}} = 3, \quad c_{\text{ME}} = 1. \tag{B.7}$$

Thus, by way of the analysis in B.2, one can predict the size of the numerical error at a given time using either of these methods. By (B.5) and (B.7), one will have:

$$\text{err}_{\text{SE}} = \frac{3ht}{\ln 10} + \log_{10} W_0, \quad \text{err}_{\text{ME}} = \frac{ht}{\ln 10} + \log_{10} W_0, \tag{B.8}$$

where err_{SE} denotes the maximum error in the Fourier spectrum (in log-10 scale) resulting from the MoC-SE, and err_{ME} has a similar meaning.

For example, suppose we integrate to $t = 500$ with $h = 0.01$. In the simulations conducted in this thesis for the integration of (1.50), $W_0 = 10^{-7.8}$ (see Fig. B.2). Therefore, by (B.8), we will have:

$$\text{err}_{\text{SE}} \approx -1.3, \quad \text{err}_{\text{ME}} \approx -5.6.$$

This is quite close to what one sees in Fig. 2.4. of Sec. 2.3.3.

C Derivation of the fourth-order local accurate Taylor interpolation used in the MoC-cRK4

In this Appendix, we derive the fourth-order local accurate Taylor approximation of the points $(Y^{\pm})^{n+1/2}_{m\mp1/2}$ used in the MoC-cRK4.

To perform this interpolation, one must have the solutions $(Y^{\pm})^{n}_{m}$ and $(Y^{\pm})^{n+1}_{m\pm1}$ with at least local accuracy $O(h^4)$. The former will be available from the previous step, and the latter can be computed by e.g. MoC-cRK3.

We will provide the details of the derivation for $(Y^{+})^{n+1/2}_{m+1/2}$. The details for the negative component will similarly follow. One begins by expanding the desired term in a fourth-order Taylor series around the point $(Y^{+})^{n}_{m}$:

$$(Y^{+})^{n+1/2}_{m+1/2} = (Y^{+})^{n}_{m} + \frac{h}{2}\left[(Y^{+})^{n}_{m}\right]' + \frac{h^2}{8}\left[(Y^{+})^{n}_{m}\right]'' + \frac{h^3}{48}\left[(Y^{+})^{n}_{m}\right]''' + O(h^4). \quad \text{(C.1)}$$

To find the unknown terms $[(Y^{+})^{n}_{m}]''$ and $[(Y^{+})^{n}_{m}]'''$ in (C.1) with the desired accuracy $O(h^4)$, we expand the known terms $(Y^{+})^{n+1}_{m+1}$ and $[(Y^{+})^{n+1}_{m+1}]' \equiv f^{+}((Y^{+})^{n+1}_{m+1}, (Y^{-})^{n+1}_{m+1})$ in Taylor series, and then solve a linear system of equations:

$$(Y^{+})^{n+1}_{m+1} = (Y^{+})^{n}_{m} + h\left[(Y^{+})^{n}_{m}\right]' + \frac{h^2}{2}\left[(Y^{+})^{n}_{m}\right]'' + \frac{h^3}{6}\left[(Y^{+})^{n}_{m}\right]''' + O(h^4), \quad \text{(C.2)}$$

$$h\left[(Y^{+})^{n+1}_{m+1}\right]' = h\left[(Y^{+})^{n}_{m}\right]' + h^2\left[(Y^{+})^{n}_{m}\right]'' + \frac{h^3}{2}\left[(Y^{+})^{n}_{m}\right]''' + O(h^4), \quad \text{(C.3)}$$

while ignoring the $O(h^4)$ terms. Equations (C.2) and (C.3) have the solutions

$$\frac{h^2}{8}\left[(Y^+)_m^n\right]'' = \frac{3}{4}(Y^+)_{m+1}^{n+1} - \frac{3}{4}(Y^+)_m^n - \frac{h}{2}\left[(Y^+)_m^n\right]' - \frac{h}{4}\left[(Y^+)_{m+1}^{n+1}\right]', \tag{C.4}$$

$$\frac{h^3}{48}\left[(Y^+)_m^n\right]''' = \frac{1}{4}(Y^+)_m^n - \frac{1}{4}(Y^+)_{m+1}^{n+1} + \frac{h}{8}\left[(Y^+)_m^n\right]' + \frac{h}{8}\left[(Y^+)_{m+1}^{n+1}\right]'. \tag{C.5}$$

Substituting (C.4) and (C.5) into (C.1) yields:

$$(Y^+)_{m+1/2}^{n+1/2} = \frac{1}{2}(Y^+)_m^n + \frac{1}{2}(Y^+)_{m+1}^{n+1} + \frac{h}{8}f^+\left((Y^+)_m^n, (Y^-)_m^n\right) - \frac{h}{8}f^+\left((Y^+)_{m+1}^{n+1}, (Y^-)_{m+1}^{n+1}\right). \tag{C.6}$$

In a similar fashion, one obtains:

$$(Y^-)_{m-1/2}^{n+1/2} = \frac{1}{2}(Y^-)_m^n + \frac{1}{2}(Y^-)_{m-1}^{n+1} + \frac{h}{8}f^-\left((Y^+)_m^n, (Y^-)_m^n\right) - \frac{h}{8}f^-\left((Y^+)_{m-1}^{n+1}, (Y^-)_{m-1}^{n+1}\right). \tag{C.7}$$

D Selection of the free parameter(s) in the NpRK methods

Recall that in the NpRK3 and NpRK4 schemes for ODEs, the user has the choice of at least one free parameter. It was shown in Chapter 4 that the free parameter has an effect on the stability region of the method, which in turn effects how well the method preserves conserved quantities. In this Appendix, we explore this fact in greater detail.

D.1 Free parameter in the NpRK3: ODE example

In the NpRK3 scheme (4.8), the parameter a_{20} is selected by the user, which then determines the rest of the coefficients by Eqs. (4.9). In Sec. 4.5.3, it was shown that the value of a_{20} which yields the stability region with highest degree of tangency to the imaginary axis (for small λh) is approximately 1.4.

This result might lead one to believe that if one integrated the Kepler two-body problem (4.30), for example, with the NpRK3, the Hamiltonian and angular momentum would be conserved the best if $a_{20} = 1.4$. Indeed, if one sets the eccentricity of the orbit, ϵ, to be any value which yields a non-stiff problem (say, $\epsilon \lesssim 0.5$) one will find this to be the case, as shown in Fig. D.1. However, for higher eccentricity values, leading to a stiff problem, the error of H and A can have a minimum at different values of a_{20}.

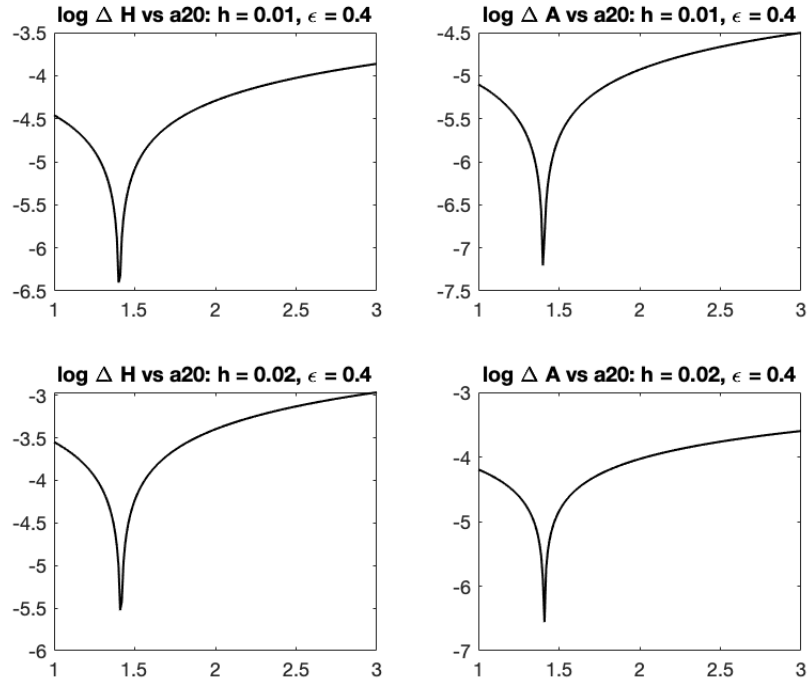

Figure D.1: Change in the Hamiltonian and angular momentum vs. a_{20} obtained when integrating the Kepler two-body problem, $\epsilon = 0.4$, $t = 50$, with the NpRK3. We give the results for $h = 0.01$ and $h = 0.02$ to show that the effect a_{20} has on these quantities appears to be independent of the stepsize (for reasonably small h). These results agree with the result of Sec. 4.5.3, that the optimal value of a_{20} occurs somewhere near 1.4.

D.2 Free parameter in the NpRK3: PDE example

In this example, we solve the system (1.51) using the MoC-NpRK3 with different values of a_{20}. We use periodic boundary conditions and the initial condition defined in Chapter 7. This system admits the conserved quantities Hamilton, H, and charge, Q, defined in (1.52) and (1.53), respectively. Thus, our goal is to see how a_{20} affects the change in each of these quantities.

When solving this system, the user must set the frequency, ω. Figure D.3 shows the results for different values of ω. The results lead us to believe that the optimal value of a_{20} in the MoC-NpRK3 may be completely unrelated to the optimal value for the ODE scheme. It also appears that in the PDE case, the optimal a_{20} depends on the parameters of the problem at hand.

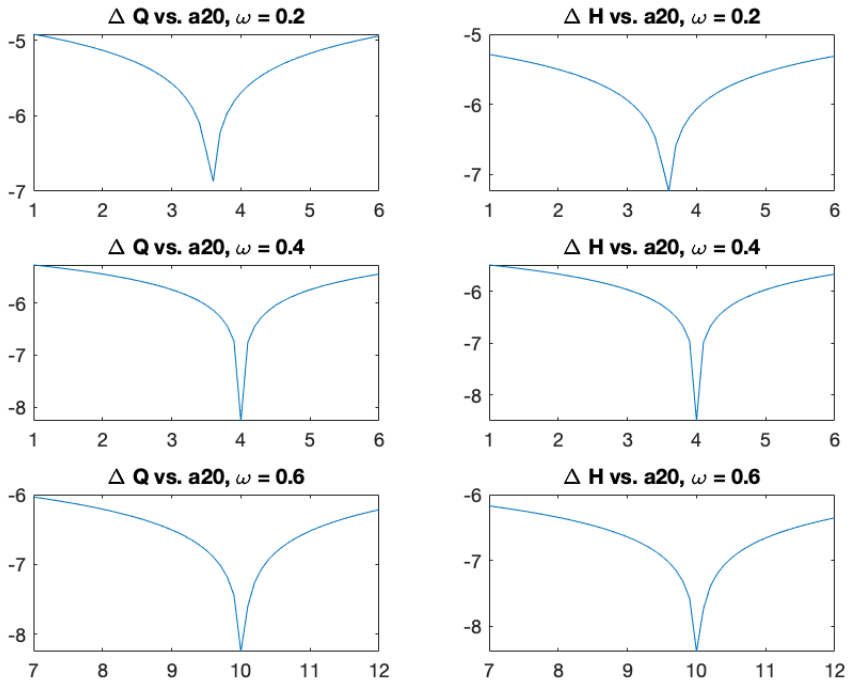

Figure D.2: Change in charge and the Hamiltonian vs. a_{20} obtained when solving system (1.51) with different values of ω and $h = 0.02$. (We obtain the same results independent of h.) The optimal value of a_{20} changes with ω, and is never close to the value of 1.4 we found for ODEs.

D.3 Free parameters in the NpRK4: ODE examples

In the NpRK4 scheme (4.10), one obtains the free parameter Λ or c_{22}. In this section, we solve the Kepler problem and investigate how each free parameter affects the conserved quantities. In Fig. D.3, we show the results when c_{22} is free, and in Fig. D.4 we show the results when Λ is free.

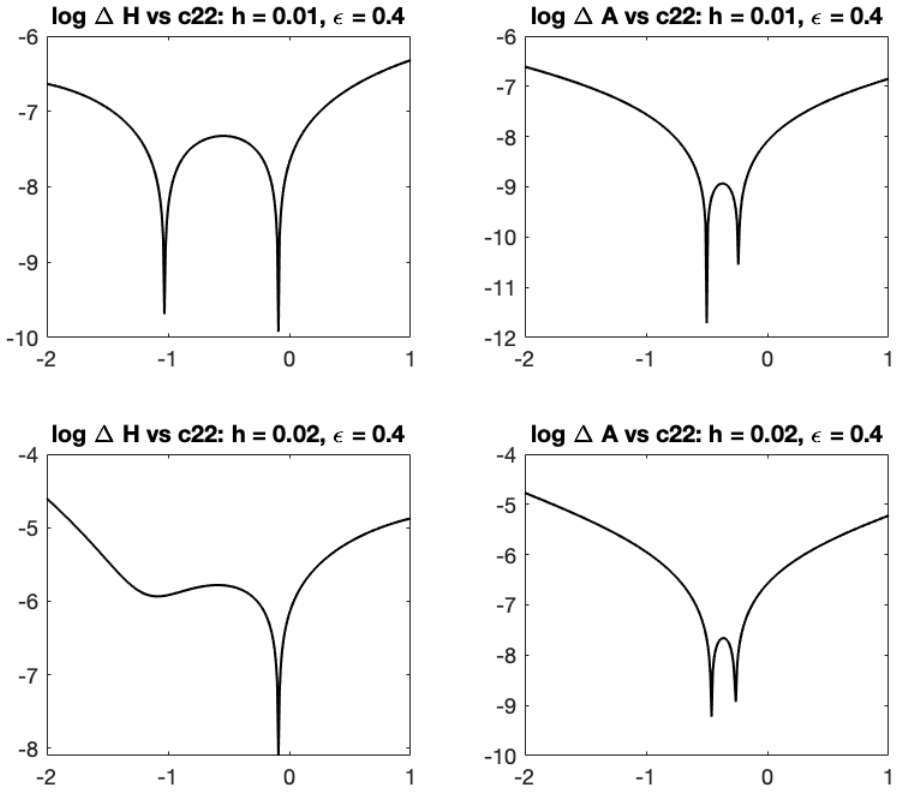

Figure D.3: Similar to Fig. D.1, but now with the NpRK4 with c_{22} free. For this method, there are two large dips in the error in each plot (except ΔH for $h = 0.02$). The ΔH plots have a consistent optimal value of $c_{22} \approx -0.1$, and for ΔA, $c_{22} \approx -0.2$ or $c_{22} \approx -0.5$. Thus, $c_{22} = -0.15$ seems like a fair compromise. This is the value we use in Chapter 4.

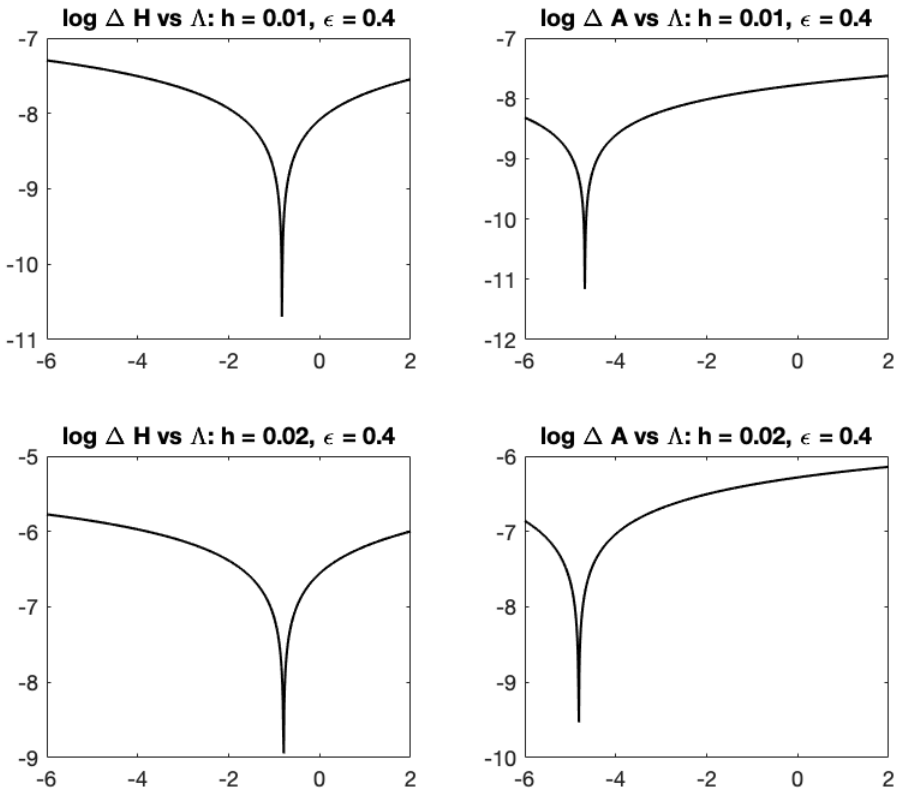

Figure D.4: Similar to Fig. D.3, but now with the NpRK4 with Λ free. There appears to be an optimal Λ for ΔH and a very different optimal Λ for ΔA. The optimal Λ for ΔA is close to -5, which corresponds to the previously analyzed case of c_{22} being free. Therefore, rather than seek a trade-off between the values of Λ, we may want to use the NpRK4 method with c_{22} free.

D.4 Free parameters in the NpRK4: PDE example

We will now examine how the MoC-NpRK4 conserves the Hamiltonian and charge when solving system (1.51) with different values of c_{22} (Fig. D.5) and Λ (Fig. D.6).

From these results, we will draw two main conclusions. The first conclusion is

that c_{22} and Λ being optimal in the ODE case does not imply they are optimal in the PDE case. The second conclusion is that, in the case of solving (1.51), we are more likely to preserve the Hamiltonian and charge as much as possible when we select c_{22} as the free parameter in the MoC-NpRK4.

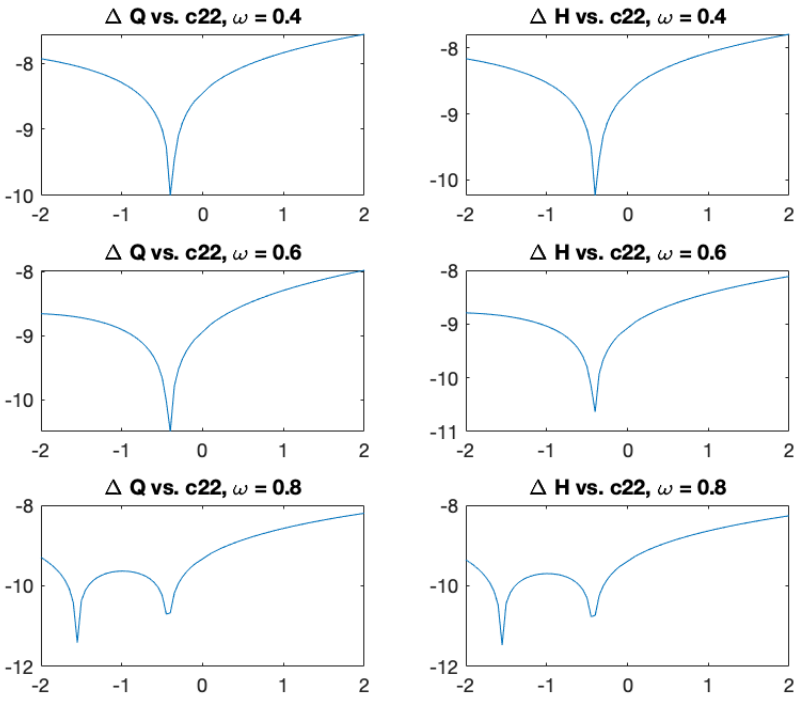

Figure D.5: Similar to Fig. D.2, but now with the MoC-NpRK4 with c_{22} free. These plots are somewhat similar to the plots of ΔA vs. c_{22} in Fig. D.2, with an optimal value of $c_{22} \approx -0.4$. For higher values of ω, we see two large dips in the error, also similar to Fig. D.2. We do not, however, see any indication that the optimal value for the MoC-NpRK4 is $c_{22} = -0.15$.

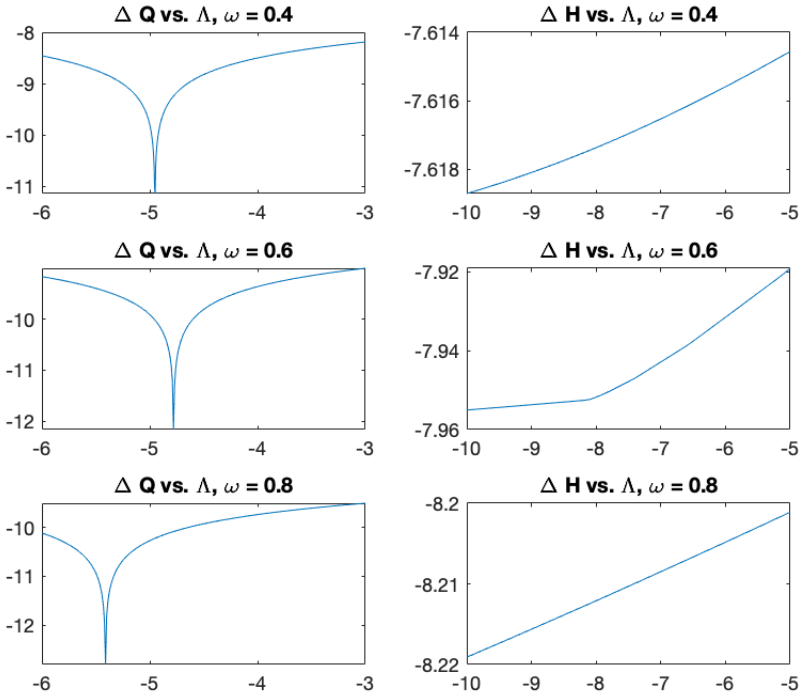

Figure D.6: Similar to Fig. D.5, but now with Λ free. In the ΔQ vs. Λ plots, we see an optimal value close to -5. This is similar to ΔA in the ODE version of this case (Fig. D.4). The optimal value of Λ for ΔH is nowhere near the value for ΔQ, though. Again, this is similar to the ODE case. From this, we conclude that selecting c_{22} as the free parameter in the MoC-NpRK4 will allow us to better optimize both of the conserved quantities.

D.5 Concluding remarks

Based on the experiments conducted in this Appendix, it seems that the optimal value of the free parameter to select in an NpRK method for ODEs can be nearly predicted by the stability plot. This does not apply to MoC-NpRK methods used for solving PDEs, though. In that case, one must investigate for themselves what value the free parameter needs to be in order to optimize the conserved quantities.

E Extrapolation of nodes outside the integration domain for use in MoC-pRK methods

When implementing nonreflecting boundary conditions in the MoC-pRK schemes, one must extrapolate points that lie outside the boundary at a few initial time levels. In this Appendix, we derive the extrapolation formulae used in our analysis.

E.1 Third-order extrapolation of a single point

In the MoC-pRK3, we require the values of $Y^{\pm}$ at the nodes $(m, n) = (-1, 1)$ and $(M + 1, 1)$ with third-order accuracy (see Sec. 5.7). Here we will explain how one obtains these values at $(-1, 1)$. The explanation for the node $(M+1, 1)$ is symmetric.

Figure E.1 shows the stencil for our argument. The time level $n = 1$, as well as the superscript "$\pm$", is irrelevant in this analysis, so we will refer to the desired value simply as Y_{-1}.

Figure E.1: Stencil for the extrapolation of Y_{-1} given the points $Y_{0,1,2}$.

We use the available values $(x, y) \equiv (0, Y_0), (h, Y_1), (2h, Y_2)$ to extrapolate the value $(-h, Y_{-1})$. Using the Lagrange form, (see, e.g. [14]), we find that the degree-3

polynomial interpolating the available points is:

$$\mathcal{L}_3(x) = Y_0\frac{(x-h)(x-2h)}{(0-h)(0-2h)} + Y_1\frac{(x-0)(x-2h)}{(h-0)(h-2h)} + Y_2\frac{(x-0)(x-h)}{(2h-0)(2h-h)}. \tag{E.1}$$

Evaluating $\mathcal{L}_3(x)$ at $-h$ gives us Y_{-1}.

$$Y_{-1} = \mathcal{L}_3(h) = 3Y_0 - 3Y_1 + Y_2. \tag{E.2}$$

Expanding the l.h.s. of (E.2) in a Taylor series yields:

$$Y_{-1} = Y_0 - hY_0' + \frac{h^2}{2}Y_0'' + O(h^3). \tag{E.3}$$

Similarly, we expand the r.h.s. of (E.2) in a Taylor series to obtain:

$$\begin{aligned} 3Y_0 - 3Y_1 + Y_2 &= 3Y_0 - 3\left(Y_0 + hY_0' + \frac{h^2}{2}Y_0''\right) + Y_0 + 2hY_0' + 2h^2Y_0'' + O(h^3) \\ &= Y_0 - hY_0' + \frac{h^2}{2}Y_0'' + O(h^3). \end{aligned} \tag{E.4}$$

The l.h.s. and r.h.s. of (E.2) agree up to terms of order 3, so (E.2) is a valid third-order formula to obtain the desired point. At the right boundary, one will have:

$$Y_{M+1} = 3Y_M - 3Y_{M-1} + Y_{M-2}. \tag{E.5}$$

E.2 Fourth-order extrapolation of a single point

The fourth-order extrapolation needed in the MOC-pRK4 is similar. To find Y_{-1}, one finds the degree-4 polynomial, $\mathcal{L}_4(x)$, interpolating the points $(0, Y_0), (h, Y_1), (2h, Y_2)$, and $(3h, Y_3)$. Then, one finds:

$$Y_{-1} = \mathcal{L}_4(-h) = 4Y_0 - 6Y_1 + 4Y_2 - Y_3. \tag{E.6}$$

It can be verified that (E.6) is fourth-order accurate by expanding the l.h.s. and r.h.s. of (E.6) in a fourth-order Taylor series. Similarly, at the right boundary, one has:

$$Y_{M+1} = 4Y_M - 6Y_{M-1} + 4Y_{M-2} - Y_{M-3}. \tag{E.7}$$

F MATLAB code for the MoC-pRK3 and MoC-pRK4 algorithms

In this Appendix, we include our commented MATLAB code implementing the MoC-pRK3 and MoC-pRK4 algorithms. The codes presented here are for systems with two components, S^+ and S^-, with the characteristics ξ^+ and ξ^-.

F.1 MoC-pRK3 algorithm

```
% MoC-pRK3 algorithm ------------------------------------------------------
    %   Conventions:
    %   s1 denotes the solution along the positive characteristic
    %   s2 denotes the solution along the negative characteristic
    %   Corresponding functions are f1 and f2
% n = 1 is given by the initial condition ---------------------------------
s1_1 = Initial Condition; s2_1 = Initial Condition;
% Extrapolate the "virtual" boundadry values by (5.15)
v1_left = 3*s1(1)-3*s1(2)+s1(3); v2_left = 3*s2(1)-3*s2(2)+s2(3);
v1_right = 3*s1(M)-3*s1(M-1)+s1(M-2); v2_right = 3*s2(M)-3*s2(M-1)+s2(M-2);
% Store boundary values for later use. These will be length-3 vectors,
% and will be denoted with a capital S.
S1_left(1) = s1_1(1); S1_right(1) = s1_1(M);
S2_left(1) = s2_1(1); S2_right(1) = s2_1(M);
% n = 2 -------------------------------------------------------------------
% Compute the MoC-SE solution, sb, to find the MoC-ME solution.
% Here, the _2 notations means these values are at n = 2.
s1b_2(2:M)   = s1(1:M-1) + dt*f1(s1(1:M-1),s2(1:M-1));
s2b_2(1:M-1) = s2(2:M)   + dt*f2(s1(2:M)  ,s2(2:M));
s1b_2(1) = Left b.c.; s2b_2(M) = Right b.c.;
% Compute the MoC-ME solution, sd.
s1d_2(2:M) =   (1/2)*(s1(1:M-1)+s1b_2(2:M)  +dt*f1(s1b_2(2:M)  ,s2b_2(2:M)));
s2d_2(1:M-1) = (1/2)*(s2(2:M)  +s2b_2(1:M-1)+dt*f2(s1b_2(1:M-1),s2b_2(1:M-1)));
s1d_2(1) = Left b.c.; s2d_2(M) = Right b.c.;
% Update the solution for n = 3.
s1 = s1d_2; s2 = s2d_2;
% Store the boundary values
S1_left(2) = s1(1); S1_right(2) = s1(M);
S2_left(2) = s2(1); S2_right(2) = s2(M);
% Main loop: for n >= 3 ---------------------------------------------------
for n = 3 : nmax
```

```
    % Compute the MoC-SE and MoC-ME solutions
    s1b(2:M)   = s1(1:M-1) + dt*f1(s1(1:M-1),s2(1:M-1));
    s2b(1:M-1) = s2(2:M)   + dt*f2(s1(2:M)  ,s2(2:M));
    s1b(1) = Left b.c.; s2b(M) = Right b.c.;
    s1d(2:M)   = (1/2)*(s1(1:M-1)+ s1b(2:M)+dt*f1(s1b(2:M)  ,s2b(2:M)));
    s2d(1:M-1) = (1/2)*(s2(2:M)+ s2b(1:M-1)+dt*f2(s1b(1:M-1),s2b(1:M-1)));
    s1d(1) = Left b.c.; s2d(M) = Right b.c.;
    if n == 3
        % Compute k1 and k2 at n = 1.
        % Notation: ki_j = stage derivative i, component j.
        k1_1_nmin2 = [f1(v1_left,v2_left),f1(s1_1(1:M-2),s2_1(1:M-2))];
        k1_2_nmin2 = [f2(s1_1(3:M),s2_1(3:M)),f1(v1_right,v2_right)];
        s1b_left = v1_left + dt*f1(v1_left,v2_left);
        s2d_left = s2d_2(1); s1d_right = s1d_2(M);
        s2b_right = v2_right + dt*f2(v1_right,v2_right);
        k2_1_nmin2 = [f1(s1b_left,s2d_left),f1(s1b_2(2:M-1),s2d_2(2:M-1))];
        k2_2_nmin2 = [f2(s1d_2(2:M-1),s2b_2(2:M-1)),f2(s1d_right,s2b_right)];
    else
        % Compute the virtual nodes by rotated MoC-ME
        s1b_left = S1_left(3) - dt*f1(S1_left(3),S2_left(3));
        s2b_left = S2_left(1) + dt*f2(S1_left(1),S2_left(1));
        % Rotated ME for the right boundary (5.16{c,d})
        v1_left = (1/2)*(S1_left(3) + s1b_left - dt*f1(s1b_left,s2b_left));
        v2_left = (1/2)*(S2_left(1) + s2b_left + dt*f2(s1b_left,s2b_left));
        % Rotated SE for the right boundary (5.17{a,b})
        s1b_right = S1_right(1) + dt*f1(S1_right(1),S2_right(1));
        s2b_right = S2_right(3) - dt*f2(S1_right(3),S2_right(3));
        % Rotated ME for the right bounday (5.17{c,d})
        v1_right = (1/2)*(S1_right(1)+s1b_right+dt*f1(s1b_right,s2b_right));
        v2_right = (1/2)*(S2_right(3)+s2b_right-dt*f2(s1b_right,s2b_right));
        % Compute k1 and k2 at level (n-2)
        k1_1_left = f1(v1_left,v2_left); k1_2_right = f2(v1_right,v2_right);
        k1_1_nmin2 = [k1_1_left,k1_1_nmin1(1:end-1)];
        k1_2_nmin2 = [k1_2_nmin1(2:end),k1_2_right];
        s1b_left = v1_left + dt*k1_1_left;
        s2b_right = v2_right + dt*k1_2_right;
        k2_1_nmin2 = [f1(s1b_left,S2_left(3)),k2_1_nmin1(1:end-1)];
        k2_2_nmin2 = [k2_2_nmin1(2:end),f2(S1_right(3),s2b_right)];
        % Update the boundary values
        S1_left(1) = S1_left(2); S1_right(1) = S1_right(2);
        S1_left(2) = S1_left(3); S1_right(2) = S1_right(3);
        S2_left(1) = S2_left(2); S2_right(1) = S2_right(2);
        S2_left(2) = S2_left(3); S2_right(2) = S2_right(3);
    end
    % Compute k1 and k2 at level (n-1)
    k1_1_nmin1 = f1(s1(1:M-1),s2(1:M-1)); k1_2_nmin1 = f2(s1(2:M),s2(2:M));
    k2_1_nmin1 = f1(s1b(2:M),s2d(2:M)); k2_2_nmin1 = f2(s1d(1:M-1),s2b(1:M-1));
    % Compute the MoC-pRK3 solution
```

```
    s1n(2:M) = s1(1:M-1)+(dt/12)*(13*k1_1_nmin1+5*k2_1_nmin1-...
                                        k1_1_nmin2-5*k2_1_nmin2);
    s2n(1:M-1) = s2(2:M)+(dt/12)*(13*k1_2_nmin1+5*k2_2_nmin1-...
                                        k1_2_nmin2-5*k2_2_nmin2);
    s1n(1) = Left b.c.; s2n(M) = Right b.c.;
    % Update the solution and store boundary values
    s1 = s1n; s2 = s2n;
    S1_left(3) = s1(1); S1_right(3) = s1(M);
    S2_left(3) = s2(1); S2_right(3) = s2(M);
end
```

F.2 MoC-pRK4 algorithm

```
% MoC-pRK4 algorithm with nonreflecting b.c. -----------------------------
% Conventions:
%   s1 denotes the solution along the positive characteristic
%   s2 denotes the solution along the negative characteristic
%   Corresponding functions are f1 and f2
%   To start, s1 and s2 have length M. Once we extrapolate the near-boundary
%   values, all solution and stage derivative vectors will have length M+2.
% Auxiliary Functions are given at the end of the main code.

% n = 1 ------------------------------------------------------------------
% a) Solution at this level given by initial condition
% b) Extrapolate to m = 0, M+1. (See Functions below)
[v1_left,v2_left] = extrapolateL(s1(1:4),s2(1:4));
[v1_right,v2_right] = extrapolateR(s1(M-3:M),s2(M-3:M));
V1_left(1) = v1_left; V2_left(1) = v2_left;
V1_right(1) = v1_right; V2_right(1) = v2_right;
% Update the solution vector
s1 = [V1_left(1),s1,V1_right(1)]; s2 = [V2_left(1),s2,V2_right(1)];
% c) Extrapolate to m = -1, M+2
% Notation: "lleft" and "rright" refer to m = -1 and m = M+2
% in this context. They have a different meaning later on.
[V1_lleft,V2_lleft] = extrapolateL(s1(1:4),s2(1:4));
[V1_rright,V2_rright] = extrapolateR(s1(M-1:M+2),s2(M-1:M+2));
% Update the solution vector
s1 = [V1_lleft,s1,V1_rright]; s2 = [V2_lleft,s2,V2_rright];
% d) Update the boundary values
% Notation: "lleft" and "rright" refer to m = 2 and m = M-1 in
% this context.
S1_left(1) = s1(3); S1_lleft(1) = s1(4);
S2_left(1) = s2(3); S2_lleft(1) = s2(4);
S1_right(1) = s1(M+2); S1_rright(1) = s1(M+1);
S2_right(1) = s2(M+2); S2_rright(1) = s2(M+1);

```

```
% n = 2 ------------------------------------------------------------------
% Set the nonreflecting b.c.
% s1left = Left_Boundary_Condition(n); s2right = Right_Boundary_Condition(n);
Bc = [s1left,s2right];
% a) Compute the MoC-cRK3 solution (see Function below)
[s1n,s2n] = cRK3(M,dt,s1(3:M+2),s2(3:M+2),Bc,F);
% b) Extrapolate to m = 0, M+1
[v1_left,v2_left] = extrapolateL(s1n(1:4),s2n(1:4));
[v1_right,v2R] = extrapolateR(s1n(M-3:M),s2n(M-3:M));
V1_left(2) = v1_left; V2_left(2) = v2_left;
V1_right(2) = v1_right; V2_right(2) = v2R;
s1n = [V1_left(2),s1n,V1_right(2)]; s2n = [V2_left(2),s2n,V2_right(2)];
% c) Extrapolate to m = -1, M+2
[V1_lleft,V2_lleft] = extrapolateL(s1n(1:4),s2n(1:4));
[V1_rright,V2_rright] = extrapolateR(s1n(M-1:M+2),s2n(M-1:M+2));
s1n = [0,s1n,0]; s2n = [0,s2n,0];
% d) Compute k_{1,2} at n = 1 by (6.2{a,b,k,l})
k1_1_nmin2 = f1(s1(1:M-1),s2(1:M-1)); k1_2_nmin2 = f2(s1(6:M+4),s2(6:M+4));
% Requires the MoC-SE solution; see Functions below.
s1b = SE_1(dt,s1(1:M-1),k1_1_nmin2); s2b = SE_2(dt,s2(6:M+4),k1_2_nmin2);
k2_1_nmin2 = f1(s1b,s2n(2:M)); k2_2_nmin2 = f2(s1n(5:M+3),s2b);
% e) Reassign the solution
s1 = s1n; s2 = s2n;
% f) Store the boundary values
S1_left(2) = s1(3); S1_lleft(2) = s1(4);
S2_left(2) = s2(3); S2_lleft(2) = s2(4);
S1_right(2) = s1(M+2); S1_rright(2) = s1(M+1);
S2_right(2) = s2(M+2); S2_rright(2) = s2(M+1);
% Notation: _p refers to values needed in the MoC-pRK3 later on.
S1_left_p(1) = s1(3); S1_right_p(1) = s1(M+2);
S2_left_p(1) = s2(3); S2_right_p(1) = s2(M+2);

% n = 3 ------------------------------------------------------------------
% Set the nonreflecting b.c.
% s1left = Left_Boundary_Condition(n); s2right = Right_Boundary_Condition(n)
Bc = [s1left,s2right];
% a) MoC-cRK3 solution
[s1n,s2n] = cRK3(M,dt,s1(3:M+2),s2(3:M+2),Bc,F);
s1n = [0,0,s1n,0,0]; s2n = [0,0,s2n,0,0];
% b) k{1,2} at n = 2
k1_1_nmin1 = f1(s1(2:M),s2(2:M)); k1_2_nmin1 = f2(s1(5:M+3),s2(5:M+3));
s1b = SE_1(dt,s1(2:M),k1_1_nmin1); s2b = SE_2(dt,s2(5:M+3),k1_2_nmin1);
k2_1_nmin1 = f1(s1b,s2n(3:M+1)); k2_2_nmin1 = f2(s1n(4:M+2),s2b);
% c) Reassign the solution
s1 = s1n; s2 = s2n;
% f) Assign boundary values
S1_left(3) = s1(3); S1_lleft(3) = s1(4);
S2_left(3) = s2(3); S2_lleft(3) = s2(4);
```

```
S1_right(3) = s1(M+2); S1_rright(3) = s1(M+1);
S2_right(3) = s2(M+2); S2_rright(3) = s2(M+1);
S1_left_p(2) = s1(3); S1_right_p(2) = s1(M+2);
S2_left_p(2) = s2(3); S2_right_p(2) = s2(M+2);

% The main loop begins here:
for n = 4 : nmax
    % Set the nonreflecting b.c.
    % s1left = Left_Boundary_Condition(n); s2right = Right_Boundary_Condition(n);
    Bc = [s1left,s2right];
   % a) Compute K1 at level n and MoC-SE soln at n+1
    k1_1_nmin0 = f1(s1(3:M+1),s2(3:M+1)); k1_2_nmin0 = f2(s1(4:M+2),s2(4:M+2));
    s1b = SE_1(dt,s1(3:M+1),k1_1_nmin0); s2b = SE_2(dt,s2(4:M+2),k1_2_nmin0);
    % b) Compute the MoC-pRK3 solution (see previous code in this Appendix)
    % Compute k1
    s1_bulk = s1(3:M+2); s2_bulk = s2(3:M+2);
    k1_1_nmin0_p = [0,f1(s1_bulk,s2_bulk),0]; k1_2_nmin0_p = [0,f2(s1_bulk,s2_bulk)
        ,0];
    % Compute the MoC-SE and MoC-ME solutions
    s1b_p(2:M)   = s1_bulk(1:M-1) + dt*f1(s1_bulk(1:M-1),s2_bulk(1:M-1));
    s2b_p(1:M-1) = s2_bulk(2:M)   + dt*f2(s1_bulk(2:M)  ,s2_bulk(2:M));
    s1b_p(1) = s1left; s2b_p(M) = s2right;
    s1d_p(2:M)   = (1/2)*(s1_bulk(1:M-1) + s1b_p(2:M)   + dt*f1(s1b_p(2:M)  ,s2b_p(2:
        M)));
    s2d_p(1:M-1) = (1/2)*(s2_bulk(2:M)   + s2b_p(1:M-1) + dt*f2(s1b_p(1:M-1),s2b_p(1:
        M-1)));
    s1d_p(1) = s1left; s2d_p(M) = s2right;
    % Compute k2 at the *current* level, in the bulk only
    k2_1_nmin0_p(1:M-1) = f1(s1b_p(2:M),s2d_p(2:M)); k2_2_nmin0_p(2:M)   = f2(s1d_p
        (1:M-1),s2b_p(1:M-1));
    k2_1_nmin0_p = [0,k2_1_nmin0_p(1:M-1),0,0]; k2_2_nmin0_p = [0,0,k2_2_nmin0_p(2:M)
        ,0];
   % Compute new solution by pRK3
   % If n = 4, we use the already computed k1_nmin1.
   % It gets computed *specifically* for this loop later on.

   if n == 4
    s1n_p(2:M) = s1_bulk(1:M-1)+(dt/12)*(13*k1_1_nmin0_p(2:M)  +5*k2_1_nmin0_p(2:M)-
        ...
                                        1*k1_1_nmin1-5*k2_1_nmin1);
    s2n_p(1:M-1) = s2_bulk(2:M)+(dt/12)*(13*k1_2_nmin0_p(3:M+1)+5*k2_2_nmin0_p(3:M+1)
        -...
                                        1*k1_2_nmin1-5*k2_2_nmin1);
   else
    s1n_p(2:M) = s1_bulk(1:M-1)+(dt/12)*(13*k1_1_nmin0_p(2:M)  +5*k2_1_nmin0_p(2:M)-
        ...
                                        1*k1_1_nmin1_p(1:M-1)-5*k2_1_nmin1_p(1:M-1));
```

```
    s2n_p(1:M-1) = s2_bulk(2:M)+(dt/12)*(13*k1_2_nmin0_p(3:M+1)+5*k2_2_nmin0_p(3:M+1)
        -...
                                        1*k1_2_nmin1_p(4:M+2)-5*k2_2_nmin1_p(4:M+2));
end
    s1n_p(1) = s1left; s2n_p(M) = s2right;
    % Update the solution and store boundary values
    s1_p = [0,s1n_p,0]; s2_p = [0,s2n_p,0];
    S1_left_p(3) = s1_p(2); S1_right_p(3) = s1_p(M+1);
    S2_left_p(3) = s2_p(2); S2_right_p(3) = s2_p(M+1);
    % Compute near-boundary values by "rotated" ME
    % Rotated SE at the left boundary (5.16{a,b})
    s1b_left_p = S1_left_p(3) - dt*f1(S1_left_p(3),S2_left_p(3));
    s2b_left_p = S2_left_p(1) + dt*f2(S1_left_p(1),S2_left_p(1));
    % Rotated ME at the right boundary (5.16{c,d})
    v1_left_p = (1/2)*(S1_left_p(3) + s1b_left_p - dt*f1(s1b_left_p,s2b_left_p));
    v2_left_p = (1/2)*(S2_left_p(1) + s2b_left_p + dt*f2(s1b_left_p,s2b_left_p));
    % Rotated SE at the right boundary (5.17{a,b})
    s1b_right_p = S1_right_p(1) + dt*f1(S1_right_p(1),S2_right_p(1));
    s2b_right_p = S2_right_p(3) - dt*f2(S1_right_p(3),S2_right_p(3));
    % Rotated ME at the right bounday (5.17{c,d})
    v1_right_p = (1/2)*(S1_right_p(1) + s1b_right_p + dt*f1(s1b_right_p,s2b_right_p))
        ;
    v2_right_p = (1/2)*(S2_right_p(3) + s2b_right_p - dt*f2(s1b_right_p,s2b_right_p))
        ;
    % Compute and store the "virtual" k1 values at the previous level
    k1_1_nmin1_p = [f1(v1_left_p,v2_left_p),k1_1_nmin0_p(2:M+1),f1(v1_right_p,
        v2_right_p)];
    k1_2_nmin1_p = [f2(v1_left_p,v2_left_p),k1_2_nmin0_p(2:M+1),f2(v1_right_p,
        v2_right_p)];
    % Compute and store the "virtual" k2 values at the previous level
    s1b_left_p = v1_left_p + dt*f1(v1_left_p,v2_left_p);
    s2d_left_p = S2_left_p(3); s1d_right_p = S1_right_p(3);
    s2b_right_p = v2_right_p + dt*f2(v1_right_p,v2_right_p);
    k2_1_nmin1_p = [f1(s1b_left_p,s2d_left_p),k2_1_nmin0_p(2:end)];
    k2_2_nmin1_p = [k2_2_nmin0_p(1:end-1),f2(s1d_right_p,s2b_right_p)];
    % Update boundary values for use in the next step
    S1_left_p(1) = S1_left_p(2); S1_right_p(1) = S1_right_p(2);
    S1_left_p(2) = S1_left_p(3); S1_right_p(2) = S1_right_p(3);
    S2_left_p(1) = S2_left_p(2); S2_right_p(1) = S2_right_p(2);
    S2_left_p(2) = S2_left_p(3); S2_right_p(2) = S2_right_p(3);
    % s1dd refers to the MoC-pRK3 solution
    s1dd = [0,s1_p,0]; s2dd = [0,s2_p,0];
    % c) Compute virtual boundary values and K's
    if n == 4
        % Do nothing for this n
    else
        % Compute m = {0,M+1}, n = n-2 by "rotated" pRK3 (function at end)
```

```
        v1Ltemp = pRK3_loc1(dt,F,s1dd(4),s2dd(4),S1_left(end),S2_left(end),S1_left(
            end-2),S2_left(end-2),'L');
        v2Ltemp = pRK3_loc2(dt,F,S1_left(end),S2_left(end),S1_left(end-2),S2_left(end
            -2),S1_lleft(end-3),S2_lleft(end-3),'L');
        v1Rtemp = pRK3_loc1(dt,F,S1_right(end),S2_right(end),S1_right(end-2),S2_right
            (end-2),S1_rright(end-3),S2_rright(end-3),'R');
        v2Rtemp = pRK3_loc2(dt,F,s1dd(M+1),s2dd(M+1),S1_right(end),S2_right(end),
            S1_right(end-2),S2_right(end-2),'R');
        if n == 5
            % Reassign boundary values
            V1_left(3) = v1Ltemp; V2_left(3) = v2Ltemp; V1_right(3) = v1Rtemp;
                V2_right(3) = v2Rtemp;
        else
            % Reassign boundary values
            V1_left(1) = V1_left(2); V2_left(1) = V2_left(2);
            V1_right(1) = V1_right(2); V2_right(1) = V2_right(2);
            V1_left(2) = V1_left(3); V2_left(2) = V2_left(3);
            V1_right(2) = V1_right(3); V2_right(2) = V2_right(3);
            V1_left(3) = v1Ltemp; V2_left(3) = v2Ltemp;
            V1_right(3) = v1Rtemp; V2_right(3) = v2Rtemp;
            % Compute m = {-1,M+2}, n = n-2
            V1_lleft = pRK3_loc1(dt,F,S1_left(end),S2_left(end),V1_left(3),V2_left(3)
                ,V1_left(1),V2_left(1),'L');
            V2_lleft = pRK3_loc2(dt,F,V1_left(3),V2_left(3),V1_left(1),V2_left(1),
                S1_left(end-4),S2_left(end-4),'L');
            V1_rright = pRK3_loc1(dt,F,V1_right(3),V2_right(3),V1_right(1),V2_right
                (1),S1_right(end-4),S2_right(end-4),'R');
            V2_rright = pRK3_loc2(dt,F,S1_right(end),S2_right(end),V1_right(3),
                V2_right(3),V1_right(1),V2_right(1),'R');
        end
        % K1_nmin2
        k1_1_nmin2 = [f1(V1_lleft,V2_lleft),k1_1_nmin1(1:M-2)];
        k1_2_nmin2 = [k1_2_nmin1(2:M-1),f2(V1_rright,V2_rright)];
        % K2_nmin2
        s1bL_nmin2 = SE_loc1(dt,F,V1_lleft,V2_lleft);
        s2bR_nmin2 = SE_loc2(dt,F,V1_rright,V2_rright);
        k2_1_nmin2 = [f1(s1bL_nmin2,V2_left(3)),k2_1_nmin1(1:M-2)];
        k2_2_nmin2 = [k2_2_nmin1(2:M-1),f2(V1_right(3),s2bR_nmin2)];
        % K1_nmin1
        k1_1_nmin1 = [f1(V1_left(3),V2_left(3)),k1_1_nmin0(1:M-2)];
        k1_2_nmin1 = [k1_2_nmin0(2:M-1),f2(V1_right(3),V2_right(3))];
        % K2_nmin1
        s1bL_nmin1 = SE_loc1(dt,F,V1_left(3),V2_left(3));
        s2bR_nmin1 = SE_loc2(dt,F,V1_right(3),V2_right(3));
        k2_1_nmin1 = [f1(s1bL_nmin1,s2(3)),k2_1_nmin0(1:M-2)];
        k2_2_nmin1 = [k2_2_nmin0(2:M-1),f2(s1(M+2),s2bR_nmin1)];
    end
    % d) Compute K2_nmin0
```

```
    k2_1_nmin0 = f1(s1b,s2dd(4:M+2));
    k2_2_nmin0 = f2(s1dd(3:M+1),s2b);
    % e) Compute the MoC-pRK4 solution (function below)
    [s1n,s2n] = pRK4fun(M,dt,Bc,s1(3:M+2),s2(3:M+2),k1_1_nmin0,k1_2_nmin0,k1_1_nmin1,
        k1_2_nmin1,k1_1_nmin2,k1_2_nmin2,...
                                                    k2_1_nmin0,k2_2_nmin0,k2_1_nmin1,
                                                        k2_2_nmin1,k2_1_nmin2,
                                                        k2_2_nmin2);
    s1 = [0,0,s1n,0,0]; s2 = [0,0,s2n,0,0];
    % f) Update the boundary values
    if n == 4 || n == 5
        lengthSLetc = length(S1_left);
        S1_left(lengthSLetc + 1) = s1(3); S1_lleft(lengthSLetc + 1) = s1(4);
        S2_left(lengthSLetc + 1) = s2(3); S2_lleft(lengthSLetc + 1) = s2(4);
        S1_right(lengthSLetc + 1) = s1(M+2); S1_rright(lengthSLetc + 1) = s1(M+1);
        S2_right(lengthSLetc + 1) = s2(M+2); S2_rright(lengthSLetc + 1) = s2(M+1);
    else
        for jSL = 1 : 4
            S1_left(jSL) = S1_left(jSL + 1);
            S2_left(jSL) = S2_left(jSL + 1);
            S1_right(jSL) = S1_right(jSL + 1);
            S2_right(jSL) = S2_right(jSL + 1);
            S1_lleft(jSL) = S1_lleft(jSL + 1);
            S2_lleft(jSL) = S2_lleft(jSL + 1);
            S1_rright(jSL) = S1_rright(jSL + 1);
            S2_rright(jSL) = S2_rright(jSL + 1);
        end
        S1_left(5) = s1(3); S1_right(5) = s1(M+2);
        S2_left(5) = s2(3); S2_right(5) = s2(M+2);
        S1_lleft(5) = s1(4); S1_rright(5) = s1(M+1);
        S2_lleft(5) = s2(4); S2_rright(5) = s2(M+1);
    end
end

% Function to extrapolate the LEFT boundary value
% at n = 1 and n = 2. (Equation E.6)
function [v1L,v2L] = extrapolateL(s1,s2)
    v1L = 4*s1(1) - 6*s1(2) + 4*s1(3) - s1(4);
    v2L = 4*s2(1) - 6*s2(2) + 4*s2(3) - s2(4);
end

% Function to extrapolate the RIGHT boundary value
% at n = 1 and n = 2 (Equation E.7)
function [v1R,v2R] = extrapolateR(s1,s2)
    v1R = 4*s1(4) - 6*s1(3) + 4*s1(2) - s1(1);
    v2R = 4*s2(4) - 6*s2(3) + 4*s2(2) - s2(1);
end

```

```
% Function to compute the MoC-cRK3 solution
function [s1n,s2n] = cRK3(M,dt,s1,s2,Bc,F)
    bc1 = Bc(1); bc2 = Bc(2); f1 = F{1}; f2 = F{2};
    % K1
    k1_1(1:M) = f1(s1(1:M),s2(1:M));
    k2_1(1:M) = f2(s1(1:M),s2(1:M));
    % s-bar (Simple euler)
    s1b(2:M) = s1(1:M-1) + dt*k1_1(1:M-1);
    s2b(1:M-1) = s2(2:M) + dt*k2_1(2:M);
    s1b(1) = bc1; s2b(M) = bc2;
    % s-dot (Modified Euler)
    s1d(2:M) = (1/2)*(s1(1:M-1) + s1b(2:M) + dt*f1(s1b(2:M),s2b(2:M)));
    s2d(1:M-1) = (1/2)*(s2(2:M) + s2b(1:M-1) + dt*f2(s1b(1:M-1),s2b(1:M-1)));
    s1d(1) = bc1; s2d(M) = bc2;
    % Auxiliary value used to find K2
    s1c(1:M-1) = s1(1:M-1) + (dt/2)*k1_1(1:M-1);
    s2c(2:M) = s2(2:M) + (dt/2)*k2_1(2:M);
    s1c(M) = s1(M) + (dt/2)*k1_1(M);
    s2c(1) = s2(1) + (dt/2)*k2_1(1);
    % s-tilde (the half-step)
    s1t(2:M) = (3/4)*s1(1:M-1) + (1/4)*s1d(2:M) + (dt/4)*k1_1(1:M-1);
    s2t(1:M-1) = (3/4)*s2(2:M) + (1/4)*s2d(1:M-1) + (dt/4)*k2_1(2:M);
    s1t(1) = (3/4)*s1(M) + (1/4)*s1d(1) + (dt/4)*k1_1(M);
    s2t(M) = (3/4)*s2(1) + (1/4)*s2d(M) + (dt/4)*k2_1(1);
    % k2
    k1_2(1:M) = f1(s1c(1:M),s2t(1:M));
    k2_2(1:M) = f2(s1t(1:M),s2c(1:M));
    % Auxiliary value used in K3
    s1h(1 : M-1) = s1(1 : M-1) - dt*k1_1(1:M-1) + 2*dt*k1_2(1:M-1);
    s2h(2 : M) = s2(2 : M) - dt*k2_1(2:M) + 2*dt*k2_2(2:M);
    s1h(M) = s1(M) - dt*k1_1(M) + 2*dt*k1_2(M);
    s2h(1) = s2(1) - dt*k2_1(1) + 2*dt*k2_2(1);
    % k3
    k1_3(1:M-1) = f1(s1h(1:M-1),s2d(2:M));
    k2_3(2:M) = f2(s1d(1:M-1),s2h(2:M));
    k1_3(M) = f1(s1h(M),s2d(1));
    k2_3(1) = f2(s1d(M),s2h(1));
    % Solution at next time step
    s1n(2 : M) = s1(1 : M-1) + dt*(1/6)*k1_1(1:M-1) + dt*(2/3)*k1_2(1:M-1) + dt*(1/6)
        *k1_3(1:M-1);
    s2n(1 : M-1) = s2(2 : M) + dt*(1/6)*k2_1(2:M) + dt*(2/3)*k2_2(2:M) + dt*(1/6)*
        k2_3(2:M);
    s1n(1) = bc1;
    s2n(M) = bc2;
end

% Function to compute SE for the 1st component
function s1b = SE_1(dt,s1,k1_1)
```

```
    s1b = s1 + dt*k1_1;
end

% Function to compute SE for the 2nd component
function s2b = SE_2(dt,s2,k2_1)
    s2b = s2 + dt*k2_1;
end

% Function to compute the rotated pRK3 soln for the FIRST component
function v1 = pRK3_loc1(dt,F,s1_p1,s2_p1,s1_p2,s2_p2,s1_p3,s2_p3,boundary)
    f1 = F{1}; f2 = F{2};
    if boundary == 'L'
        % 1) K1(+) at p1 and p2 ----------------------------------------
        k1_1_p1 = f1(s1_p1,s2_p1); k1_1_p2 = f1(s1_p2,s2_p2);
        % 2) K2(+) at p1 -----------------------------------------------
        % SE(+) at p2
        s1b_p2 = s1_p1 - dt*k1_1_p1;
        % ME(-) at p2 we already have, so no need to compute
        k1_2_p1 = f1(s1b_p2,s2_p2);
        % 3) K2(+) at p2 -----------------------------------------------
        s1b_v = s1_p2 - dt*k1_1_p2;
        s2b_v = s2_p3 + dt*f2(s1_p3,s2_p3);
        % ME(-) at v
        s2d_v = (1/2)*(s2_p3 + s2b_v + dt*f2(s1b_v,s2b_v));
        k1_2_p2 = f1(s1b_v,s2d_v);
        % 4 ) pRK3 soln at v -------------------------------------------
        v1 = s1_p2 - dt*((13/12)*k1_1_p2 + (5/12)*k1_2_p2 - (1/12)*k1_1_p1 - (5/12)*
            k1_2_p1);
    else
        % 1) K1(+) at p3 and p2 ----------------------------------------
        k1_1_p3 = f1(s1_p3,s2_p3); k1_1_p2 = f1(s1_p2,s2_p2);
        % 2) K2(+) at p3 -----------------------------------------------
        s1b_p2 = s1_p3 + dt*k1_1_p3;
        % ME(-) at p2 we already have
        % Compute k2 at p3
        k1_2_p3 = f1(s1b_p2,s2_p2);
        % 3) K2(+) at p2 -----------------------------------------------
        s1b_v = s1_p2 + dt*k1_1_p2;
        s2b_v = s2_p1 - dt*f2(s1_p1,s2_p1);
        % ME(-) at v
        s2d_v = (1/2)*(s2_p1 + s2b_v - dt*f2(s1b_v,s2b_v));
        % Compute K2(+) at p2
        k1_2_p2 = f1(s1b_v,s2d_v);
        % pRK3 solution ------------------------------------------
        v1 = s1_p2 + dt*((13/12)*k1_1_p2 + (5/12)*k1_2_p2 - (1/12)*k1_1_p3 - (5/12)*
            k1_2_p3);
    end
end
```

```
339
340 % Function to compute the rotated pRK3 soln for the SECOND component
341 function v2 = pRK3_loc2(dt,F,s1_p1,s2_p1,s1_p2,s2_p2,s1_p3,s2_p3,boundary)
342     f1 = F{1}; f2 = F{2};
343     if boundary == 'L'
344         % 1) K1(-) at p2 and p3
345         k2_1_p2 = f2(s1_p2,s2_p2); k2_1_p3 = f2(s1_p3,s2_p3);
346         % 2) K2(-) at p2
347         s1b_v = s1_p1 - dt*f1(s1_p1,s2_p1);
348         s2b_v = s2_p2 + dt*k2_1_p2;
349         s1d_v = (1/2)*(s1_p1 + s1b_v - dt*f1(s1b_v,s2b_v));
350         k2_2_p2 = f2(s1d_v,s2b_v);
351         % 3) K2(-) at p3
352         s2b_p2 = s2_p3 + dt*k2_1_p3;
353         k2_2_p3 = f2(s1_p2,s2b_p2);
354         % 4) pRK3 soln at v
355         v2 = s2_p2 + dt*((13/12)*k2_1_p2 + (5/12)*k2_2_p2 - (1/12)*k2_1_p3 - (5/12)*
                k2_2_p3);
356     else
357         % 1) K1(-) at p1 and p2
358         k2_1_p2 = f2(s1_p2,s2_p2); k2_1_p1 = f2(s1_p1,s2_p1);
359         % 2) K2(-) at p1
360         s2b_p2 = s2_p1 - dt*k2_1_p1;
361         k2_2_p1 = f2(s1_p2,s2b_p2);
362         % 3) K2(-) at p2
363         s1b_v = s1_p3 + dt*f1(s1_p3,s2_p3);
364         s2b_v = s2_p2 - dt*k2_1_p2;
365         s1d_v = (1/2)*(s1_p3 + s1b_v + dt*f1(s1b_v,s2b_v));
366         k2_2_p2 = f2(s1d_v,s2b_v);
367         % 4) pRK3 soln
368         v2 = s2_p2 - dt*((13/12)*k2_1_p2 + (5/12)*k2_2_p2 - (1/12)*k2_1_p1 - (5/12)*
                k2_2_p1);
369     end
370 end
371
372 function [s1n,s2n] = pRK4fun(M,dt,Bc,s1,s2,k1_1_nmin0,k2_1_nmin0,k1_1_nmin1,
        k2_1_nmin1,k1_1_nmin2,k2_1_nmin2,k1_2_nmin0,k2_2_nmin0,k1_2_nmin1,k2_2_nmin1,
        k1_2_nmin2,k2_2_nmin2)
373     bc1 = Bc(1); bc2 = Bc(2);
374     s1n(2:M) = s1(1:M-1) + dt*(37/24)*k1_1_nmin0 + dt*(9/24)*k1_2_nmin0...
375                          - dt*(7/12)* k1_1_nmin1 - dt*(9/24)*k1_2_nmin1...
376                          + dt*(1/24)* k1_1_nmin2 + dt*(9/24)*k1_2_nmin2;
377     s2n(1:M-1) = s2(2:M) + dt*(37/24)*k2_1_nmin0 + dt*(9/24)*k2_2_nmin0...
378                          - dt*(7/12)* k2_1_nmin1 - dt*(9/24)*k2_2_nmin1...
379                          + dt*(1/24)* k2_1_nmin2 + dt*(9/24)*k2_2_nmin2;
380     s1n(1) = bc1; s2n(M) = bc2;
381 end
```