MULTI-RATE RUNGE-KUTTA-CHEBYSHEV TIME STEPPING FOR PARABOLIC EQUATIONS ON ADAPTIVELY REFINED MESHES

by

Talin Mirzakhanian

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mathematics

Boise State University

August 2017

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Talin Mirzakhanian

Thesis Title: Multi-rate Runge-Kutta-Chebyshev Time Stepping for Parabolic Equations on Adaptively Refined Meshes

Date of Final Oral Examination: 27 April 2017

The following individuals read and discussed the thesis submitted by student Talin Mirzakhanian, and they evaluated her presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Donna Calhoun, Ph.D.	Chair, Supervisory Committee
Grady Wright, Ph.D.	Member, Supervisory Committee
Jodi L. Mead, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Donna Calhoun, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. Donna Calhoun, for her continuing support, patience, and encouragement. Without her, this thesis would not have been possible. I also would like to thank my committee members, Dr. Jodi Mead and Dr. Grady Wright for their services. My gratitude extends to my family and friends for continuously supporting me throughout the program in various ways.

Abstract

In this thesis, we develop an explicit multi-rate time stepping method for solving parabolic equations on a one dimensional adaptively refined mesh. Parabolic equations are characterized by their stiffness and as a result are usually solved using implicit time stepping schemes [16]. However, implicit schemes have the disadvantage that they can be expensive in higher dimensions or complicated to implement on adaptive or otherwise non-uniform meshes. Moreover, for coupled systems of parabolic equations, it can be difficult to achieve the expected order of accuracy without using sophisticated operator splitting techniques. For these reasons, we seek to exploit the properties of stabilized explicit methods for parabolic equations. In particular, we use the Runge-Kutta-Chebyshev (RKC) methods, a family of explicit Runge-Kutta methods, with numerical stability regions that extend far into the left half plane [12, 15, 21, 22, 26, 27, 28].

A central goal of this thesis is to use a second order RKC scheme to numerically solve parabolic equations on a one dimensional adaptively refined finite volume mesh. To make our implementation efficient, we design a time stepping algorithm in which time step sizes are chosen to respect the local mesh widths. This time stepping process requires communication between the RKC stages on different refinement levels. By linearly interpolating in time between the stage values, we obtain the ghost cell values for the finite volume scheme on each level. To our knowledge, this approach to adaptively refining in time, commonly referred to as a "multi-rate time stepping" strategy, combined with RKC time stepping method has not been previously implemented.

We develop our multi-rate algorithm on a one dimensional statically refined mesh using the second order finite volume scheme to numerically solve the heat or diffusion equation on each grid stored in a hierarchy of meshes. Using the "method of manufactured solutions", we demonstrate that our method is second order accurate, and for our test problem, the multi-rate scheme requires only about 20% of the computational work required by the uniformly refined mesh at the same resolution.

The algorithm we develop manages the time stepping between the refinement levels only, and so extends directly to higher dimensional problems. Future work in this direction includes applying the new multi-rate RKC time stepping scheme to biological pattern formations or crystal growth in the 2D ForestClaw code [7] on parallel machines.

Contents

Abstract v

List of Tables ix

List of Figures xi

1 Introduction: The heat equation 1

2 Numerical schemes for the heat equation 4

3 The Runge-Kutta-Chebyshev (RKC) schemes 12

3.1 Stability functions 13

3.1.1 First order RKC schemes 15

3.1.2 Second order RKC schemes 19

3.2 Damped stability functions 25

3.2.1 First order damped RKC schemes 26

3.2.2 Second order damped RKC schemes 29

3.3 RKC time stepping formulas 30

3.4 Efficiency of the RKC schemes 35

3.5 Numerical results of the RKC schemes 38

4 Multi-rate RKC time stepping on adaptively refined meshes 42

4.1 Spatial refinement using tree data structures 44

4.2 Temporal refinement using multi-rate time stepping 45

4.3 Heat equation with AMR and multi-rate time stepping 49

4.4 Interpolation schemes on the spatial grids 57

5 Conclusion 61

Bibliography 63

A Butcher Tableau of RKC 67

B The order of Accuracy of RKC 71

C Eigenvalues of the MOL discretization matrix of the heat equation 76

List of Tables

3.1 Comparing one time step solved with a first order RKC scheme and one time step solved using a stability function. Parameters: s is the number of stages, and $P_s(z)$ is the stability function of first order RKC schemes. 35

3.2 Comparing one time step solved with a second order RKC scheme and one time step solved using a stability function. Parameters: s is the number of stages, and $B_s(z)$ is the stability function of second order RKC schemes. 36

3.3 Errors from the problem given in equation (3.28) using the first order RKC schemes. The last column shows the convergence rate of the scheme. 39

3.4 Errors from the problem given in equation (3.28) using the second order RKC schemes. The last column shows the convergence rate of the scheme. 39

4.1 Comparing uniform versus adaptively refined grids, as well as comparing global versus multi-rate time stepping. The uniform mesh has a fixed number of levels (minlevel=maxlevel=6) while the adaptive mesh varies in refinement levels (minlevel=3, maxlevel=6). 54

4.2 Errors produced using three different interpolation schemes to fill in fine grid ghost cell values. The “mixed” scheme refers to the scheme that uses pointwise values from both the coarse and fine grid. The last row shows the errors computed using the exact solution to fill in the fine grids ghost cell values. 60

A.1 The Butcher tableau for the RKC Scheme 68

List of Figures

1.1 An example of a solution $U(x,t)$ of the heat equation in equation (1.1) with a heat source term $S(x,t)$ that produces sharp fronts in the solution. At the regions with a sharp gradient, we will want to locally refine the mesh to improve computational efficiency. 3

2.1 Stability regions of four basic time-stepping schemes: Forward Euler, Backward Euler, Trapezoidal, and fourth order Runge-Kutta schemes in the complex plane. The red shaded region is the stability region, and $z = k\lambda$ where λ is the eigenvalue and k is the time step size. 10

3.1 The stability functions $P_s(z)$ for $2 \le s \le 5$ 17

3.2 The stability region of $P_3(z)$. The stability region is the red shaded region. 18

3.3 The stability region of $P_6(z)$. 18

3.4 The stability functions $B_s(x)$ for $2 \le s \le 5$. 22

3.5 The stability region of $B_5(z)$. 23

3.6 The stability region of $B_6(z)$. 23

3.7 The stability functions $B_5(z)$ and $B_6(z)$ and their intersections with the lines $y = \pm 1$. The stability interval of an odd degree stability function is slightly larger than an even degree stability function. The odd intersection point (red circle) is the x^* point referred to in the text. 24

3.8 The difference between the x^* and β_R for an odd degree stability function for second order RKC schemes. 25

3.9 The stability region of $P_3(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping. 28

3.10 The stability region of $P_6(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping. 28

3.11 The stability region of $B_5(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping. 31

3.12 The stability region of $B_6(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping. 31

3.13 The plot of error versus the spatial grid size h: as h decreases, the error decreases as well. The Forward Euler and second order RKC schemes are second order, and the first order RKC scheme is first order. The numbers in the parentheses show the numerical convergence rates. 40

3.14 The plot of computational work (in time) versus the number of grid cells N: as N increases, the computational work increases as well. The amount of work required for both RKC schemes is lower and increasing at a rate slower than the Forward Euler scheme. The numbers in the parentheses show the numerical convergence rates. 41

3.15 The plot of work (in time) versus the error: the computational cost increases as the error decreases. For better accuracy, more computational work is needed. The convergence rates in parentheses show that the second order RKC schemes are the most efficient, while the first order RKC schemes are the least efficient. 41

4.1 An exmple of a two dimensional adaptively refined mesh, where each rectangle represents a grid. The resolution of the grids increases as we get closer to the interface (the blue swirl). 43

4.2 A binary tree data structure. The $0-1$ bit index uniquely locates each node in the tree structure. 45

4.3 Multi-rate time stepping on two levels of refinement: a coarse grid of level 0, which takes one full time step, and a fine grid of level 1, which takes two half time steps. The spatial resolution on level 1 is twice as much as the spatial resolution on level 0. 46

4.4 Obtaining ghost cell values on each stage by linearly interpolating in time using the adjacent grids' stages while using multi-rate time stepping on two levels of refinement. 47

4.5 Performance gains using multi-rate time stepping on four levels of refinement: the number of stages computed by multi-rate time stepping is 97 as opposed to global time stepping with 160 stages. 49

4.6 One dimensional refinement pattern for a regularized square wave of radius 0.25 centered at $x = 0.5$. The bottom plot is the composite grid of the problem with refinement 51

4.7 The binary tree structure of the grid in Figure 4.6 with the composite grid that shows the refinement pattern. 52

4.8 The linear relationship of the computational time and the number of evaluations of the right hand side function. This shows that the computational time does not involve overhead. 53

4.9 The plot of error versus the effective grid size h. As we decrease the grid size h, the error decreases as well. Multi-rate time stepping produces comparable results compared to global time stepping. 55

4.10 The plot of computational work (number of evaluations of the RHS function) versus the effective grid resolution N. As we increase the effective N, the computational work increases. The multi-rate scheme takes far less computational work. 56

4.11 The plot of computational work (number of evaluations of the RHS) versus the error. The computational cost increases as the error decreases. The multi-rate approach with AMR has the smallest error. . . 56

4.12 The plot of computational work (number of evaluations of the RHS) versus the effective N. In the uniformly refined approach, we increase the levels from $3-3$ to $6-6$ (on the uniform grid, the minimum and maximum levels are equal). On the AMR grids, we increase the maximum level each time from $3-3$ to $3-6$. The multi-rate approach has the smallest rate of increase. 57

4.13 Pointwise interpolation using coarse grids. The average value of the two cell values on the fine grid is computed to fill in the ghost cell value on the coarse grid. Then, this value is used with two other cell values on the coarse grid to interpolate quadratically and obtain a ghost cell value on the fine grid. 58

4.14 Pointwise interpolation using two cell values from the fine grid and one cell value from the coarse grid. The arrows show the cell values used for the pointwise interpolation. 59

4.15 Conservative interpolation using an integral to compute the cell averages. The arrows show the cells on which the integrals are imposed. . . . 60

Chapter 1

INTRODUCTION: THE HEAT EQUATION

A standard example of a parabolic partial differential equation (PDE) is the time dependent diffusion of heat, vorticity, momentum, or tracer species. This model is often a component in a larger system of equations for modeling fluid flow in engineering and science applications. For example, in incompressible flow problems, the viscous terms are modeled using a parabolic equation. In other problems, parabolic equations are used to model the diffusion of heat, energy, chemical species, and so on. For this thesis, we will use the terms "parabolic equation", "heat equation", or "diffusion equation" interchangeably.

In one dimension, a simple example of heat diffusion is the change in a temperature field over time in a one dimensional rod. The rod has a length l and has a sufficiently small diameter so that the heat is distributed equally over the cross section at time t. The surface of the rod is insulated, therefore, there is no heat loss through the surface [10, 17]. The temperature distribution of the rod can be modeled by the solution of the initial boundary value problem,

$$U_t(x,t) = \kappa U_{xx}(x,t) + S(x,t), \qquad x \in (0,l), \quad t \in (0,T), \tag{1.1}$$

where κ is the diffusion coefficient (for simplicity, we assume that $\kappa = 1$), $S(x,t)$ is the heat source term, T is the final time, and $l > 0$ is the length of the rod. The initial condition is a function of x given by

$$U(x,0) = f(x),$$

and the Dirichlet boundary conditions are prescribed as

$$U(0,t) = g(0,t),$$
$$U(l,t) = g(l,t),$$

where the temperature at both ends may be a function of time.

As an example, in Figure 1.1, we show the change in the solution of the heat flow over time with a heat source term. In this example, we see that the heat is leaving the domain, so the total amount of heat in the domain is decreasing over time. The black curve shows the initial condition, and the curves below the black curve show the solution at different times as they reach a uniform temperature of zero. The sharp fronts are produced by the heat source term $S(x,t)$ in equation (1.1). The heat source term causes the solution to maintain its sharp fronts while diffusing over time. Therefore, the solution does not behave like the solution of a typical diffusion problem in which the solution diffuses fast in areas with high curvature. This specific example will be discussed in more detail in Chapter 4.

On a uniform grid, in order to effectively approximate the numerical solution in Figure 1.1 with a desired level of accuracy, we are restricted to discretizing the grid using the smallest grid size required on the sharp fronts. While this is effective for accuracy reasons, we are potentially wasting computational effort in areas where the solution is smooth. In such areas, a lower number of grid cells would be potentially

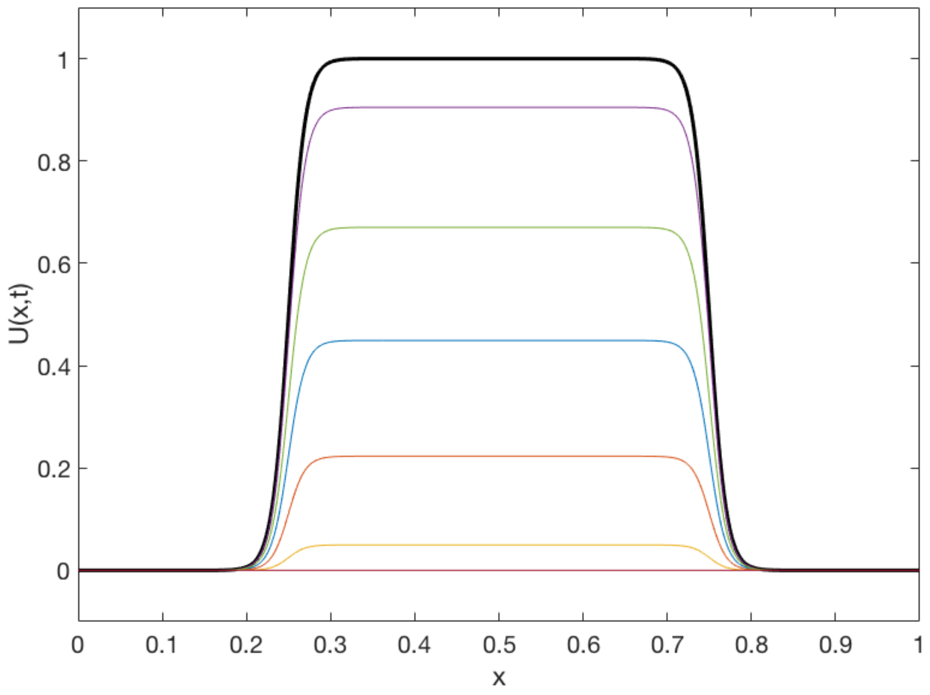


Figure 1.1: An example of a solution $U(x,t)$ of the heat equation in equation (1.1) with a heat source term $S(x,t)$ that produces sharp fronts in the solution. At the regions with a sharp gradient, we will want to locally refine the mesh to improve computational efficiency.

adequate to achieve a desired level of accuracy. The use of adaptive mesh refinement (AMR) allows us to split the spatial domain into a number of grids with different resolutions which could produce a comparable level of accuracy with a significantly less computational effort. In addition, it is advantageous to adaptively refine in time as well and take time step sizes that are chosen to respect the local spatial grid sizes. This time stepping strategy is often referred to as a “multi-rate time stepping” strategy. Using adaptive mesh refinement with multi-rate time stepping can lead to significant savings and computational work compared to the work needed to achieve a comparable solution on a uniformly refined mesh.

Chapter 2

NUMERICAL SCHEMES FOR THE HEAT EQUATION

One approach to numerically solving the heat equation is to use a semi-discretization or a Method-of-Lines (MOL) approach. In this approach, we first discretize in space to get a system of ordinary differential equations, and then apply standard time stepping methods. There are several methods that can be used for the spatial discretization such as the finite difference, finite element, and finite volume methods. Here, we use the finite volume method. We break the spatial domain into N grid cells and approximate the total integral of the unknown solution $U(x,t)$ over each grid cell. The finite volume scheme solves for the cell average of $U(x,t)$ over a finite volume cell. This average can be written as

$$U_i(t) \approx \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U(x,t)\,dx,$$

where the limits of integration are: $x_{i-\frac{1}{2}} = x_i - \frac{h}{2}$, and $x_{i+\frac{1}{2}} = x_i + \frac{h}{2}$. We have N cells: $i = 1,\ldots,N$, and $h = \dfrac{1}{N}$ is the spatial scale (assuming a domain of length 1), and $x_i = \left(i - \frac{1}{2}\right)h$ are the cell center values, and $x_{i-\frac{1}{2}} = (i-1)\,h$ are the cell edges. Using an integral form of the differential equation (1.1) first presented in Chapter 1, we get that the time rate of change of the average is given by

$$\frac{1}{h}\frac{d}{dt}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U(x,t)\,dx = \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} U_{xx}(x,t)\,dx + \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} S(x,t)\,dx.$$

Computing the integrals and using the fundamental theorem of calculus, we have,

$$\begin{aligned}\frac{d}{dt}U_i(t) &= \frac{1}{h}\left[U_x\left(x_{i+\frac{1}{2}},t\right) - U_x\left(x_{i-\frac{1}{2}},t\right)\right] + \hat{S}(x_i,t),\\ \implies U_i'(t) &= \frac{1}{h}\left[U_x\left(x_{i+\frac{1}{2}},t\right) - U_x\left(x_{i-\frac{1}{2}},t\right)\right] + \hat{S}_i(t),\end{aligned}$$

where the prime is differentiation with respect to t. We can replace the right hand side term with its difference formula:

$$\begin{aligned}U_i'(t) &= \frac{1}{h}\left[\frac{U(x_{i+1},t) - U(x_i,t)}{h} - \frac{U(x_i,t) - U(x_{i-1},t)}{h}\right] + \hat{S}_i(t),\\ &= \frac{1}{h^2}\left[U(x_{i+1},t) - 2U(x_i,t) + U(x_{i-1},t)\right] + \hat{S}_i(t).\end{aligned} \tag{2.1}$$

Equation (2.1) can be turned into a system of ordinary differential equations (ODEs) in time:

$$U'(t) = AU(t) + b(t) + \hat{S}(t), \tag{2.2}$$

where $U(t)$ is the vector of solution values, $b(t)$ accounts for the inhomogeneous boundary conditions, and A is a tridiagonal matrix with the three stencils in equation (2.1):

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}, \tag{2.3}$$

Before we consider applying a numerical time stepping scheme, we first investigate the eigenvalues of A, as these will be used to determine any time step restrictions on our numerical scheme. The eigenvalues of A can be computed as

$$\lambda_p = \frac{2}{h^2}\left(\cos\left(\frac{p\pi}{N+1}\right) - 1\right) \quad \text{for} \quad p = 1, 2, \ldots, N. \tag{2.4}$$

A derivation of the eigenvalues in equation (2.4) can be found in Appendix C.

A matrix is stable when all of its eigenvalues have a norm strictly less than one. The largest eigenvalue in magnitude dominates the behavior of the matrix and controls the stability of the system [16]. The value of the largest eigenvalue λ_N occurs when $p = N$. Since N is a large number,

$$\frac{N}{N+1} \approx 1 \implies \cos\left(\frac{N\pi}{N+1}\right) \approx \cos(\pi),$$

thus, for the largest eigenvalue we have,

$$\lambda_N \approx \frac{2}{h^2}\left(\cos(\pi) - 1\right) = -\frac{4}{h^2}. \tag{2.5}$$

The eigenvalues of the matrix A lie on the negative real axis and their magnitudes increase with N.

We now use a time stepping scheme to discretize the problem in time. We take time steps:

$$t_n = nk, \quad k = t_{n+1} - t_n, \quad t \in [0, T],$$

where t_n is the n-th time step, k is the fixed time step size, and T is the final time. As an example, we discuss the explicit Forward Euler time stepping scheme. In order to compute the solution at time t_{n+1}, the Forward Euler scheme can be written as,

$$U^{n+1} = U^n + kF\left(U^n\right), \tag{2.6}$$

where k is the time step, $U^n \approx U\left(t_n\right)$ is the numerical approximation of the exact solution $U\left(t_n\right)$ on the grid cells at time t_n $\big($the variable x_i in $U\left(t_n\right)$ is dropped because this equation is for all x values$\big)$, and the function $F\left(U^n\right)$ is the right hand side function, which is equal to:

$$F\left(U^n\right) = AU^n + b\left(t\right) + S^n.$$

Without loss of generality, we can assume that the heat source term is equal to zero, and that we have homogeneous Dirichlet boundary conditions; for a more detailed approach, please see [16]. We replace the right hand side function $F\left(U^n\right) = AU^n$ in equation (2.6),

$$\begin{aligned} U^{n+1} &= U^n + k\left(AU^n\right), \\ &= \left(I + kA\right)U^n, \\ \Longrightarrow U^{n+1} &= BU^n, \end{aligned}$$

where I is an identity matrix, and B is a new matrix equal to $I + kA$. The solutions U_i^{n+1} will remain bounded in time if the eigenvalues of B are less than 1. In order to determine the eigenvalues of the matrix B, we take advantage of the fact that we know the eigenvalues of A. Let λ be an eigenvalue of A, then $1 + k\lambda$ is an eigenvalue of B. For every eigenvalue in B, we must have:

$$
\begin{aligned}
&|1 + k\lambda| \leq 1, \\
\implies &-1 \leq 1 + k\lambda \leq 1, \\
\implies &-2 \leq k\lambda \leq 0. \qquad (2.7)
\end{aligned}
$$

We define the stability region, $\mathcal{S} \subset \mathbb{C}$ as

$$\mathcal{S}_{FE} = \{k\lambda : |1 + k\lambda| \leq 1, \lambda \in \mathbb{C}\}.$$

In Figure 2.1a, the stability region of the Forward Euler scheme is shown (the red shaded region) in the complex plane. It is a disk of radius 1 centered at point $(-1, 0)$ in the complex plane.

The product $z = k\lambda$ must stay inside the stability region to have a stable scheme. The largest most negative eigenvalue from our discretization of the heat equation is

$$\lambda_N \approx -\frac{4}{h^2}.$$

Replacing λ with λ_N in the inequality in equation (2.7), we have

$$
\begin{aligned}
&-2 \leq k\lambda_N \leq 0, \\
&-2 \leq -\frac{4}{h^2}k \leq 0, \\
\implies &k \leq \frac{h^2}{2}.
\end{aligned}
$$

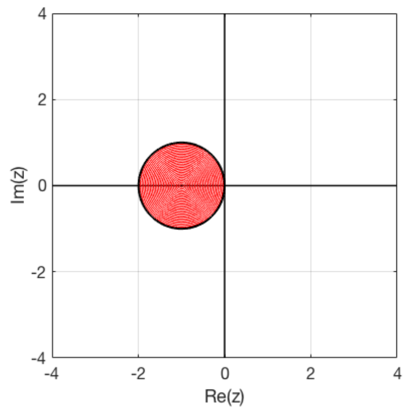
If the spatial scale h is small (where, eventually, we assume that $h \ll 1$), we must take a small time step size k to maintain stability. This is a severe restriction on the time step. This shows the "stiffness" of the heat equation, which is the general idea that the size of the time step is severely restricted due to the numerical scheme and the nature of the physical problem. For the stiff heat or diffusion problem, we must choose small time steps for stability rather than for accuracy reasons [16]. In a time stepping scheme, it is preferred to have a time step size controlled by accuracy rather than stability considerations. For the Forward Euler scheme, the temporal scale is proportional to the square of the spatial scale,

$$k \sim h^2.$$

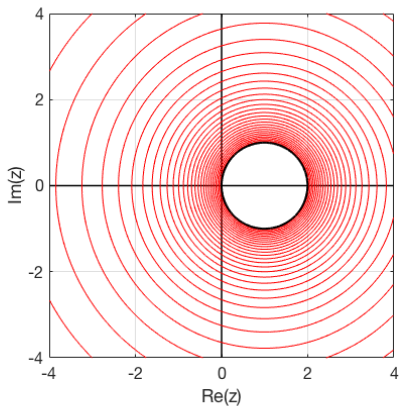
As the small grid size h approaches zero, the time step k becomes much smaller. For example, if we divide h by a factor of 2, then k must be divided by a factor of 4. The Forward Euler scheme is not an optimally efficient scheme to numerically solve the heat equation. Ideally, in time stepping schemes k should be equal or proportional to h, not h^2.

There are several popular time stepping schemes including Forward Euler, Backward Euler, Trapezoidal, and fourth order Runge-Kutta schemes with properties that make some of them suitable for stiff problems.. Figure 2.1 shows the stability regions of these time stepping schemes in the complex plane. The Forward Euler and the fourth order Runge-Kutta schemes in Figures 2.1a and 2.1c, respectively, are explicit schemes with limited stability intervals on the negative real axis. Explicit methods have some advantages and disadvantages including:

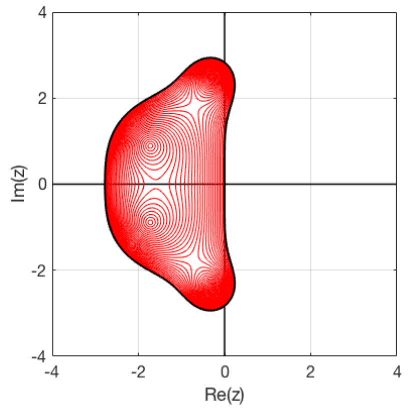
- They are relatively straightforward to implement, especially for coupled systems of parabolic PDEs.
- They do not require the solution to a linear system of equations, which is potentially expensive.



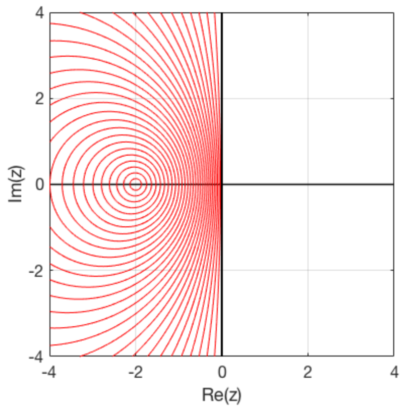
(a) Forward Euler



(b) Backward Euler



(c) Fourth order Runge-Kutta



(d) Trapezoidal

Figure 2.1: Stability regions of four basic time-stepping schemes: Forward Euler, Backward Euler, Trapezoidal, and fourth order Runge-Kutta schemes in the complex plane. The red shaded region is the stability region, and $z = k\lambda$ where λ is the eigenvalue and k is the time step size.

- However, explicit schemes, when applied to stiff problems, can have severe time step restrictions.

The Backward Euler and Trapezoidal schemes in Figures 2.1b and 2.1d, respectively, are implicit schemes with extended stability regions that contain the entire left half plane. Implicit schemes, have some advantages and disadvantages as well, which include:

- They often have infinite stability regions that contain the entire left half plane. Such schemes are "unconditionally stable".
- Work for a reasonable time step size.
- However, they are challenging to implement on non-uniform meshes.
- Coupled systems may require operator splitting techniques, which can introduce time stepping errors.
- Require solutions to linear systems, which can be expensive.

While implicit schemes are unconditionally stable for parabolic stiff problems, they require matrix assembly for complicated (adaptively refined) meshes. They can be complicated because the stencils required to approximate the Laplacian in each cell are not easily computed. For these reasons, we would like to avoid using implicit methods, and seek to exploit the properties of stabilized explicit methods for stiff and mildly stiff parabolic equations.

Chapter 3

THE RUNGE-KUTTA-CHEBYSHEV (RKC) SCHEMES

A family of explicit stabilized Runge-Kutta schemes that have been developed for mildly stiff heat or diffusion problems are the Runge-Kutta-Chebyshev (RKC) schemes [12, 13, 15, 21, 22, 26, 27, 28]. The RKC schemes were designed by Van Der Houwen and Sommeijer (1980) for the explicit time integration of stiff systems of ODEs which originate from spatial discretization of parabolic PDEs [13, 28]. The RKC schemes are easy to implement because they are explicit; and they avoid errors that can occur with operator splitting approaches in implicit schemes. For mildly stiff reaction-diffusion problems, the RKC schemes can numerically solve the problem efficiently [21].

The RKC schemes have several advantages. They possess an extended stability region, and because they are explicit, they are easy to implement. They also exploit the recursion relations of the Chebyshev polynomials which will be explained later in this chapter. The RKC schemes only require six vectors of working storage, independent of the number of stages [21]. In addition, they have internal stability within the intermediate stages [13]. Some of the disadvantages of the RKC schemes include being challenging to design for higher orders of accuracy and non-intuitive formulas and parameters.

3.1 Stability functions

Consider the well known Dahlquist test problem [15, 16],

$$U'(t) = \lambda U(t), \qquad U(0) = 1, \tag{3.1}$$

where λ is a large negative scalar and $U(t)$ is the unknown function. The exact solution of this problem is the exponential function $U(t) = e^{\lambda t}$ [15, 16]. A typical one-step time stepping scheme for this test problem can be written as

$$U^{n+1} = R(z)\, U^n,$$

where $z = k\lambda$, k is the time step, and $R(z)$ is called the "stability function" of the scheme. For example, for the Forward Euler scheme we have,

$$\begin{aligned} &U^{n+1} = U^n + kF(U^n) = U^n + k\lambda U^n = (1 + z)\, U^n, \\ &R(z) = 1 + z. \end{aligned}$$

The stability function must satisfy two requirements: accuracy and stability. For accuracy, the stability function must approximate the exponential function with some order of accuracy [15],

$$R(z) \approx e^z = 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots. \tag{3.2}$$

For example, to produce a first order accurate time stepping scheme, the stability function must include at least the first two terms of the Taylor series expansion of the exponential function in equation (3.2),

$$R(z) = 1 + z + O\left(z^2\right).$$

Similarly, to produce a second order accurate time stepping scheme, the stability function must include at least the first three terms of the Taylor series expansion in equation (3.2),

$$R(z) = 1 + z + \frac{1}{2}z^2 + O\left(z^3\right).$$

The accuracy of the time stepping scheme improves as the stability function $R(z)$ includes more terms of the Taylor series expansion of the exponential function in equation (3.2); therefore, for a p-th order accurate time stepping scheme, the stability function must exactly include the first $p+1$ terms,

$$R(z) = e^z + O\left(z^{p+1}\right).$$

For stability, in order for the solutions not to grow exponentially, the magnitude of the stability function must be less than or equal to 1 [15],

$$|R(z)| \leq 1.$$

The goal is to find or design a stability function $R(z)$ so that the stability region in the complex plane

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\},$$

contains a large interval on the negative real axis

$$[-\beta_R, 0\ ],$$

with β_R, the boundary of absolute stability, as large as possible.

3.1.1 First order RKC schemes

Consider the following potential stability function (for first order RKC schemes),

$$P_s(z) = T_s\left(1 + \frac{z}{s^2}\right), \tag{3.3}$$

where $T_s(x)$ is a Chebyshev polynomial of the first kind of degree s. Evaluating $P_s(z)$ for any s values, the stability function produces first order accuracy,

$$P_s(z) = T_s\left(1 + \frac{z}{s^2}\right) = 1 + z + O\left(z^2\right).$$

For example, for $P_3(z)$ and $P_6(z)$ we have,

$$P_3(z) = T_3\left(1 + \frac{z}{3^2}\right) = 1 + z + \frac{4z^2}{27} + \frac{4z^3}{729}, \tag{3.4}$$

$$P_6(z) = T_6\left(1 + \frac{z}{6^2}\right) = 1 + z + \frac{35z^2}{216} + \frac{7z^3}{729} + \frac{z^4}{3888} + \frac{z^5}{314928} + \frac{z^6}{68024448}. \tag{3.5}$$

The stability interval and the boundary of absolute stability β_R are obtained by finding z such that $|P_s(z)| \leq 1$,

$$\left|T_s\left(1 + \frac{z}{s^2}\right)\right| \leq 1.$$

From properties of Chebyshev polynomials we have $|T_s(x)| \leq 1 \implies |x| \leq 1$ [25]; therefore,

$$\left|T_s\left(1 + \frac{z}{s^2}\right)\right| \leq 1 \quad \implies \quad \left|1 + \frac{z}{s^2}\right| \leq 1.$$

We can solve for all z such that this inequality holds. In general, z is a complex number; however, since we are looking for an interval on the real axis, we can assume that z is real, and obtain

$$\begin{aligned} &\left|1 + \frac{z}{s^2}\right| \leq 1, \\ \Longrightarrow &-1 \leq 1 + \frac{z}{s^2} \leq 1, \\ \Longrightarrow &-2s^2 \leq z \leq 0. \end{aligned}$$

The interval of absolute stability for the first order RKC schemes with the stability function in equation (3.3) is

$$[-\beta_R, 0], \text{ where } \beta_R = 2s^2. \tag{3.6}$$

This interval increases quadratically in s. Figure 3.1 shows a number of stability functions $P_s(z)$ for different s values $2 \leq s \leq 5$. This stability boundary is optimal in the following sense.

Theorem 3.1.1. *(Van Der Houwen, Verwer) For any explicit, consistent Runge-Kutta method, the boundary of absolute stability β_R depends on the number of stages s with $\beta_R \leq 2s^2$, and the optimal stability function is the shifted Chebyshev polynomial of the first kind [12, 15, 26, 27].*

The proof of this theorem was provided by Markoff in 1892 and Van Der Houwen in 1996 and it can be found in [15, 26].

As an example to illustrate the stability regions in the complex plane, we consider the stability functions of $P_3(z)$ and $P_6(z)$ in equations (3.4) and (3.5). Figures

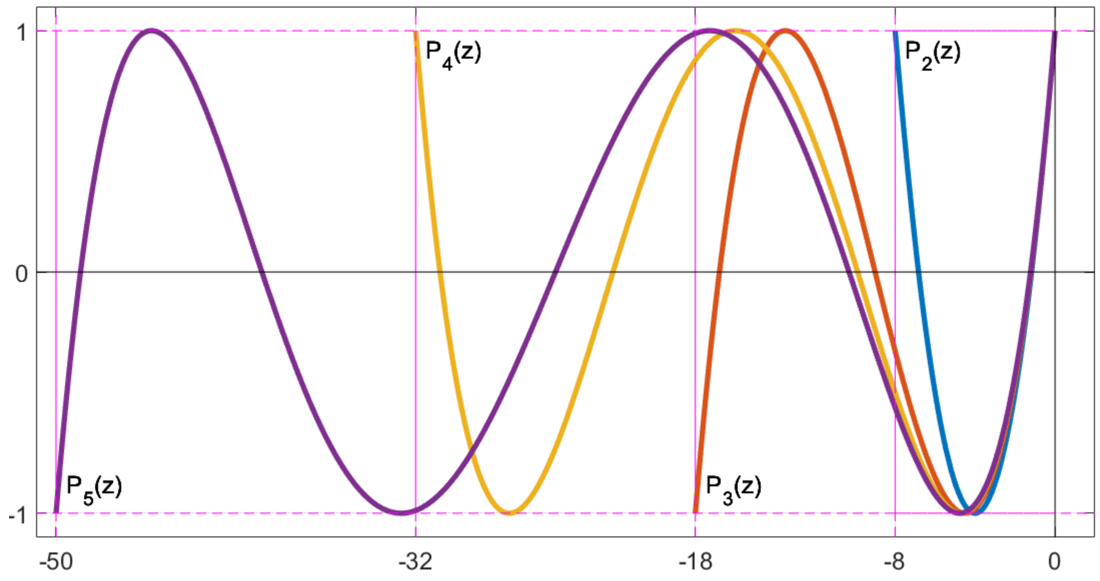


Figure 3.1: The stability functions $P_s(z)$ for $2 \leq s \leq 5$

3.2 and 3.3 show the stability regions of $P_3(z)$ and $P_6(z)$ in the complex plane, respectively. The stability intervals can be computed using equation (3.6) as

$$[-\beta_R, 0] = [-18, 0], \quad \text{for } P_3(z),$$
$$[-\beta_R, 0] = [-72, 0], \quad \text{for } P_6(z).$$

A simple one-step multi-stage Runge-Kutta-Chebyshev scheme that can be shown to produce the stability function for the first order RKC schemes in equation (3.3) is given by [21, 27, 28, 29, 30],

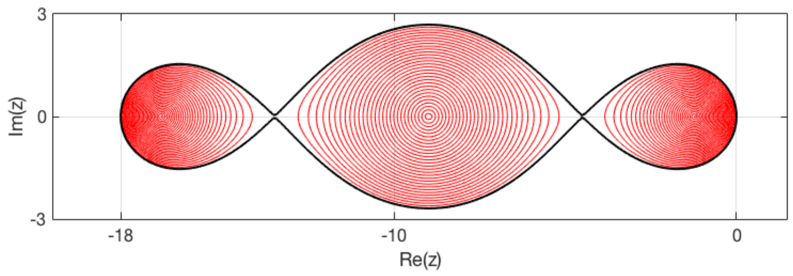


Figure 3.2: The stability region of $P_3(z)$. The stability region is the red shaded region.

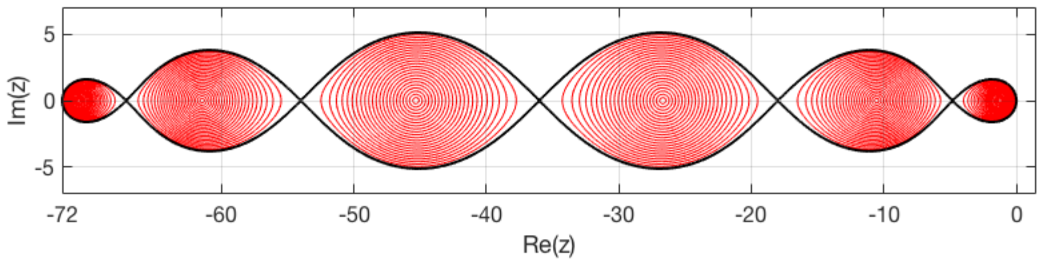


Figure 3.3: The stability region of $P_6(z)$.

$$
\begin{aligned}
Y_0 &= U^n, \\
Y_1 &= Y_0 + \frac{k}{s^2} F_0, \\
Y_j &= 2Y_{j-1} - Y_{j-2} + \frac{2k}{s^2} F_{j-1}, \qquad 2 \le j \le s, \\
U^{n+1} &= Y_s.
\end{aligned}
\tag{3.7}
$$

where Y_j values are the intermediate solutions at each RKC stage, and they depend on the previous two stages, s is the number of stages, and F_j's are the values of the right hand side F at stage Y_j. We are again solving the Dahlquist test problem in equation (3.1). Applying the scheme in equation (3.7) to this test problem, we recover the stability functions in equation (3.3) for each stage

$$
\begin{aligned}
Y_0 &= U^n, \\
Y_1 &= \left(1 + \frac{z}{s^2}\right) Y_0 = T_1 \left(1 + \frac{z}{s^2}\right) U^n, \\
Y_2 &= 2Y_1 - Y_0 + \frac{2z}{s^2} Y_1 = T_2 \left(1 + \frac{z}{s^2}\right) U^n, \\
Y_3 &= 2Y_2 - Y_1 + \frac{2z}{s^2} Y_2 = T_3 \left(1 + \frac{z}{s^2}\right) U^n, \\
Y_j &= 2Y_{j-1} - Y_{j-2} + \frac{2z}{s^2} Y_{j-1} = T_j \left(1 + \frac{z}{s^2}\right) U^n, \qquad 4 \le j \le s, \\
U^{n+1} &= Y_s.
\end{aligned} \tag{3.8}
$$

In this derivation, we have used the three-term recursion relations for Chebyshev polynomials of the first kind [25]

$$
\begin{aligned}
&T_0(x) = 1, \qquad T_1(x) = x, \\
&T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x), \qquad 2 \le j \le s.
\end{aligned}
$$

3.1.2 Second order RKC schemes

For higher order RKC schemes (second order and higher), there are no analytical expressions for truly optimal stability functions in the sense of $\beta_R = 2s^2$ [1, 12, 27]. For the second order stability functions, there are two approximate functions in analytic form for arbitrary $s \ge 2$. One approximate function in analytical form was given by Bakker in 1971 [5]. In 1996, Van Der Houwen and Sommeijer constructed another stability function for the second order RKC schemes; however, they preferred Bakker's function above theirs. [15, 27]. Bakker's stability functions for the second order RKC schemes are given by

$$
B_s(z) = \frac{2}{3} + \frac{1}{3s^2} + \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s \left(1 + \frac{3z}{s^2 - 1}\right). \tag{3.9}
$$

This stability function is not optimal (it generates about 80% of the optimal stability region); however, it has been extensively tested in the literature and has been proven to perform well [12, 27]. The derivation and proof of this stability function can be found in [15, 26]. If we evaluate $B_s(z)$ for different s values, we obtain second order accuracy:

$$B_s(z) = \frac{2}{3} + \frac{1}{3s^2} + \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s\left(1 + \frac{3z}{s^2-1}\right) = 1 + z + \frac{1}{2}z^2 + O\left(z^3\right).$$

For example, for $B_5(z)$ and $B_6(z)$ we have

$$\begin{aligned} B_5(z) &= \frac{2}{3} + \frac{1}{3(5)^2} + \left(\frac{1}{3} - \frac{1}{3(5)^2}\right) T_5\left(1 + \frac{3z}{(5)^2-1}\right) \\ &= 1 + z + \frac{z^2}{2} + \frac{7z^3}{80} + \frac{z^4}{160} + \frac{z^5}{6400} \qquad (3.10) \\ B_6(z) &= \frac{2}{3} + \frac{1}{3(6)^2} + \left(\frac{1}{3} - \frac{1}{3(6)^2}\right) T_6\left(1 + \frac{3z}{(6)^2-1}\right) \\ &= 1 + z + \frac{z^2}{2} + \frac{16z^3}{175} + \frac{324z^4}{42875} + \frac{432z^5}{1500625} + \frac{216z^6}{52521875}. \qquad (3.11) \end{aligned}$$

The stability interval, and the boundary of absolute stability β_R, are obtained by finding z such that $|B_s(z)| \le 1$:

$$\left|\frac{2}{3} + \frac{1}{3s^2} + \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s\left(1 + \frac{3z}{s^2-1}\right)\right| \le 1,$$

$$-1 - \left(\frac{2}{3} + \frac{1}{3s^2}\right) \le \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s\left(1 + \frac{3z}{s^2-1}\right) \le 1 - \left(\frac{2}{3} + \frac{1}{3s^2}\right).$$

Then dividing all sides by $\left(\frac{1}{3} - \frac{1}{3s^2}\right)$ and simplifying further, we have,

$$-1 - \frac{4s^2+2}{s^2-1} \le T_s\left(1 + \frac{3z}{s^2-1}\right) \le 1.$$

For $s \ge 2$, the expression on the left is strictly less than -1, therefore, we only need to check,

$$-1 \le T_s\left(1 + \frac{3z}{s^2-1}\right) \le 1,$$

and from properties of Chebyshev polynomials we have $|T_s(x)| \le 1 \implies |x| \le 1$ [25], implying

$$\left|T_s\left(1 + \frac{3z}{s^2-1}\right)\right| \le 1 \quad \implies \quad \left|1 + \frac{3z}{s^2-1}\right| \le 1.$$

Therefore, we can solve for all z such that this inequality holds. Similar to the first order case, since we are looking for the interval on the real axis, we assume that z is real,

$$\begin{aligned} &\left|1 + \frac{3z}{s^2-1}\right| \le 1, \\ \implies &-1 \le 1 + \frac{3z}{s^2-1} \le 1, \\ \implies &-2 \le \frac{3z}{s^2-1} \le 0, \\ \implies &-\frac{2}{3}\left(s^2-1\right) \le z \le 0. \end{aligned}$$

The interval of absolute stability for the second order RKC schemes with the stability function in equation (3.9) is

$$[-\beta_R, 0], \quad \text{where} \qquad \beta_R \approx \frac{2}{3}\left(s^2 - 1\right). \tag{3.12}$$

Again, the interval increases quadratically in s. Figure 3.4 shows a number of stability functions $B_s(z)$ for different s values $2 \leq s \leq 5$. As an example to illustrate the stability

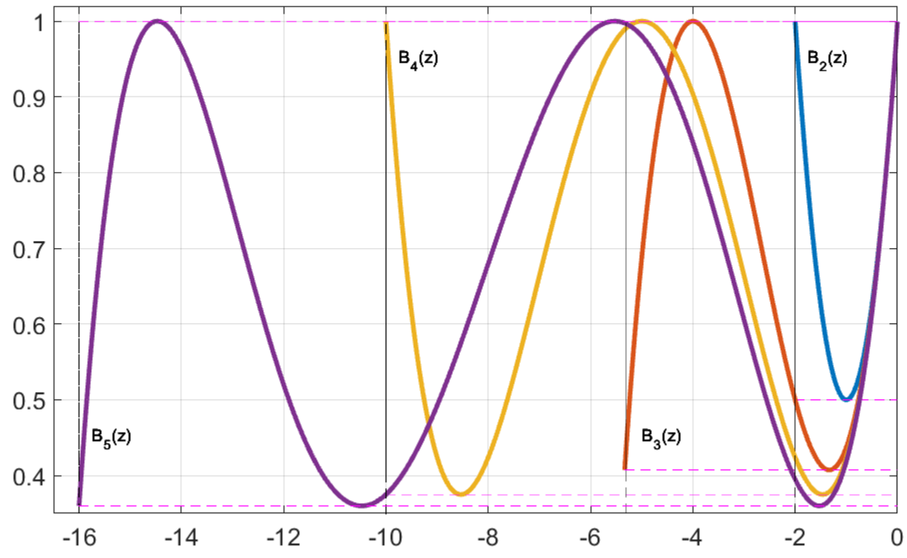


Figure 3.4: The stability functions $B_s(x)$ for $2 \leq s \leq 5$.

regions in the complex plane, we consider the stability functions of $B_5(z)$ and $B_6(z)$ in equations (3.10) and (3.11). Figures 3.5 and 3.6 show the stability regions of $B_5(z)$ and $B_6(z)$ in the complex plane, respectively. The stability intervals can be computed using the equation (3.12) and are given by

$$[-\beta_R, 0] = [-16, 0], \quad \text{for } B_5(z),$$
$$[-\beta_R, 0] = \left[-\frac{70}{3}, 0\right], \quad \text{for } B_6(z).$$

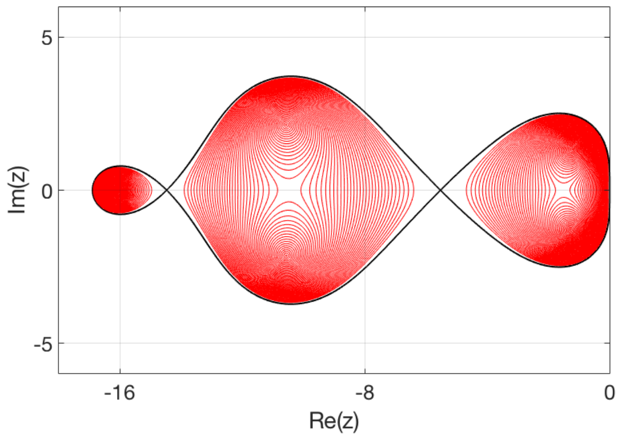


Figure 3.5: The stability region of $B_5(z)$.

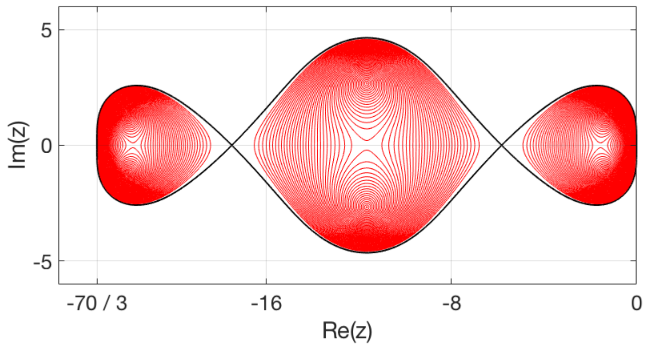


Figure 3.6: The stability region of $B_6(z)$.

As pointed out in the literature, odd and even degree Chebyshev polynomials produce slightly different stability intervals for the stability function in equation (3.9) [15, 27]. For an even degree, the stability interval is exactly equal to the interval in equation (3.12), whereas for an odd degree, the stability interval is slightly larger. In Figure 3.7, we plot the stability functions $B_5(z)$ and $B_6(z)$, and show their intersections with the lines $y = 1$ and $y = -1$. In this plot, the even function $B_6(z)$ alternates between $\left(\frac{1}{3} + \frac{2}{3s^2}\right)$ and 1, and intersects the line $y = 1$ exactly at $\beta_R = \frac{70}{3}$. The odd function $B_5(z)$, also alternates between $\left(\frac{1}{3} + \frac{2}{3s^2}\right)$ and 1, but intersects the line $y = -1$ at a point $x^* < -16$, so we can still take the interval with absolute stability boundary $\beta_R = -16$. We compute the actual intersection point x^* numerically, and show that it is always less than the boundary of absolute stability $\beta_R = -\frac{2}{3}\left(s^2 - 1\right)$. In Table 3.8, we show the difference between x^* and β_R, which converges to approximately 0.88 as the number of stages increases. This does not have a major effect on the stability region; in fact, it increases the interval of absolute stability which is favorable.

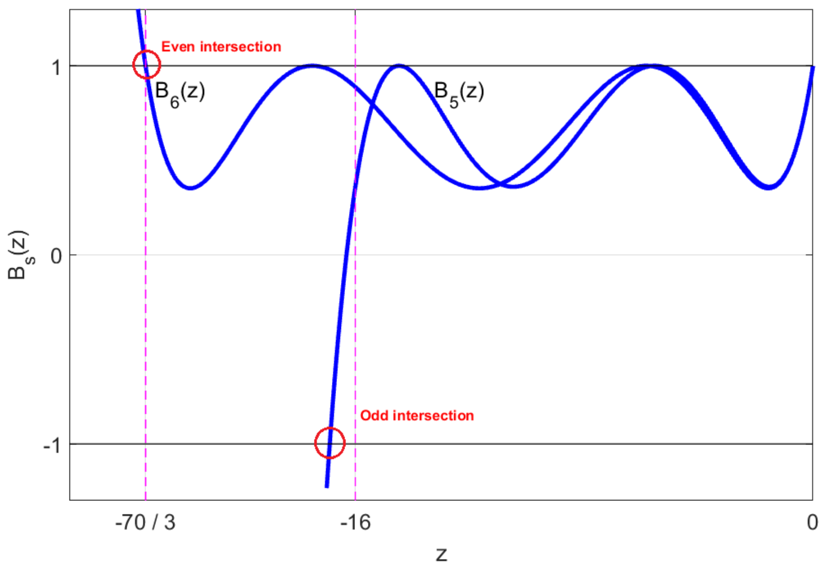


Figure 3.7: The stability functions $B_5\left(z\right)$ and $B_6\left(z\right)$ and their intersections with the lines $y = \pm 1$. The stability interval of an odd degree stability function is slightly larger than an even degree stability function. The odd intersection point (red circle) is the x^* point referred to in the text.

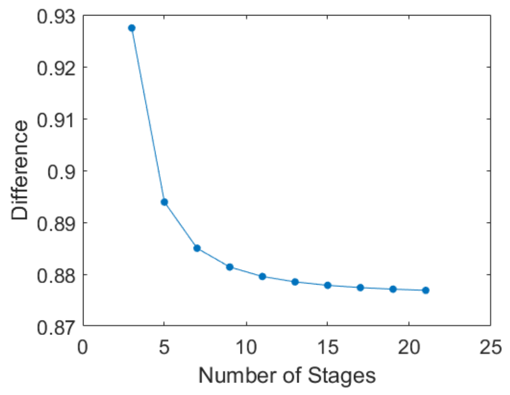


Figure 3.8: The difference between the x^* and β_R for an odd degree stability function for second order RKC schemes.

3.2 Damped stability functions

In Figures 3.2 and 3.3, the stability regions for first order RKC schemes, and Figures 3.5 and 3.6, the stability regions for second order RKC schemes, wherever on the real axis the stability function $|R(z)| = 1$, the stability region $\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}$ contracts to a point on the real axis.

As seen in Figures 3.2, 3.3 (first order) and Figures 3.5 and 3.6 (second order), the stability region $\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}$ contracts to a point on the negative real axis. This is very restrictive in some applications; a small imaginary perturbation on z might cause instability [15]. Since perturbing such a z away from the axis into the complex plane gives a point z where $|R(z)| > 1$, it is preferable to perturb the stability functions with a damping parameter. The choice of damping was first made by Guillou and Lago in 1961 [12, 16, 27]. By damping the stability functions with a damping parameter, the magnitude of the stability function, $|R(z)|$, becomes bounded slightly below 1 on the real axis between $-\beta_R$ and the origin, but it remains the same order accurate. The result is a slight decrease in the value of β_R, leading to a slightly smaller stability interval than the original undamped version [3]. This slight decrease in the interval of absolute stability is acceptable, and it is a small price to pay to avoid the stability regions contracting to a boundary point on the real axis.

In the damped version, some new parameters are introduced, such as the damping parameter

$$w_0 = 1 + \frac{\epsilon}{s^2}, \tag{3.13}$$

where ϵ is a small positive number. A commonly chosen value for ϵ is 0.05 for the first order schemes and $\dfrac{2}{13}$ for the second order schemes [15, 16]. In order to derive the damped stability functions of the RKC schemes, we use the following representation of the RKC schemes [13, 27, 28]. We assume that $R(z)$ has the form

$$R(z) = a_s + b_s T_s(w_0 + w_1 z), \tag{3.14}$$

where by imposing accuracy requirements of first or second order, we can obtain the parameters w_1, a_s, and b_s for first and second order RKC schemes.

3.2.1 First order damped RKC schemes

In order to derive the damped stability function of the first order RKC schemes, assume the stability function $P_s(z)$ has the form given in equation (3.14) and we set

$$P_s(z) = a_s + b_s T_s(w_0 + w_1 z). \tag{3.15}$$

For consistency and first order accuracy, we have,

$$P_s(0) = a_s + b_s T_s(w_0) = 1, \quad \text{and} \quad P_s'(0) = b_s w_1 T_s'(w_0) = 1,$$

where we have two equations and three unknown parameters w_1, a_s, and b_s. Thus, one of the parameters is an arbitrary parameter. In the literature, this arbitrary parameter is a_s, and is set to zero [13, 15, 28]. Then, we obtain an expression for b_s as

$$a_s = 0, \tag{3.16}$$

$$\Longrightarrow P_s(0) = b_s T_s(w_0) = 1 \Longrightarrow b_s = \frac{1}{T_s(w_0)}. \tag{3.17}$$

Then, we can find w_1,

$$P_s'(0) = b_s w_1 T_s'(w_0) = 1 \Longrightarrow w_1 = \frac{1}{b_s T_s'(w_0)} \Longrightarrow w_1 = \frac{T_s(w_0)}{T_s'(w_0)}. \tag{3.18}$$

After writing the expressions for b_s and w_1, we can write the stability function in the format of equation (3.15),

$$P_s(z) = a_s + b_s T_s(w_0 + w_1 z) = \frac{T_s(w_0 + w_1 z)}{T_s(w_0)}, \tag{3.19}$$

which presents the damped stability function of the first order RKC schemes. In Section 3.3, we will numerically show that this stability function is generated by the RKC schemes. The damped stability interval of the first order RKC is

$$[-\beta_R, 0], \text{ where } \beta_R \approx 2s^2. \tag{3.20}$$

The interval increases quadratically by s similar to the undamped version. The boundary of absolute stability is slightly less than $2s^2$ compared to the undamped version. The analytical expression of the boundary of absolute stability β_R, of the damped first order RKC schemes has been derived by Hundsdorfer and Verwer in [15, 27]. In order to show the decrease in the stability region with damping, we plot

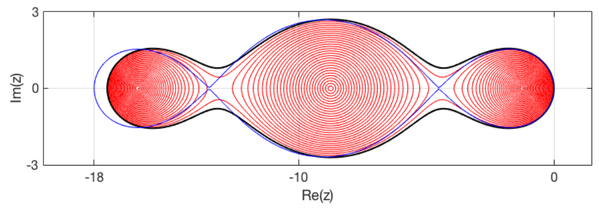


Figure 3.9: The stability region of $P_3(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping.

the damped and undamped stability regions on the same axes. Figures 3.9 and 3.10 show the stability regions of the damped stability functions of $P_3(z)$ and $P_6(z)$ in the complex plane. The slightly stretched (blue) curve in the figures shows the stability region without damping.

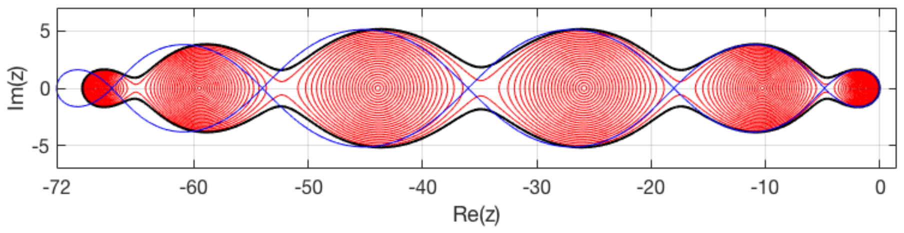


Figure 3.10: The stability region of $P_6(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping.

3.2.2 Second order damped RKC schemes

Similar to first order RKC, we assume that the stability function $B_s(z)$ of the second order RKC schemes has the form given in equation (3.14), so we set

$$B_s(z) = a_s + b_s T_s(w_0 + w_1 z). \tag{3.21}$$

For consistency and second order accuracy, we have

$$B_s(0) = a_s + b_s T_s(w_0) = 1, \qquad B_s'(0) = b_s w_1 T_s'(w_0) = 1, \qquad B_s''(0) = b_s w_1^2 T_s''(w_0) = 1,$$

where we have three equations and three unknown parameters a_s, b_s, and w_1. Solving for w_1 we have,

$$\frac{B_s''(0)}{B_s'(0)} = \frac{b_s w_1^2 T_s''(w_0)}{b_s w_1 T_s'(w_0)} = 1 \implies w_1 = \frac{T_s'(w_0)}{T_s''(w_0)}.$$

Then, solving for b_s we have,

$$B_s'(0) = b_s w_1 T_s'(w_0) = 1 \implies b_s = \frac{1}{w_1 T_s'(w_0)} \implies b_s = \frac{T_s''(w_0)}{(T_s'(w_0))^2}.$$

Finally, we can solve for a_s,

$$a_s = 1 - b_s T_s(w_0) = 1 - \frac{T_s''(w_0)}{(T_s'(w_0))^2} T_s(w_0).$$

After writing the expressions for b_s and w_1, we can write the stability function in the format of equation (3.21),

$$B_s(z) = 1 - \frac{T_s''(w_0)}{(T_s'(w_0))^2} T_s(w_0) + \frac{T_s''(w_0)}{(T_s'(w_0))^2} T_s(w_0 + w_1 z),$$

which can be factored as

$$B_s(z) = 1 + \frac{T_s''(w_0)}{(T_s'(w_0))^2} \left(T_s(w_0 + w_1 z) - T_s(w_0)\right), \tag{3.22}$$

which presents the damped stability function of the second order RKC schemes. In Section 3.3, we will numerically show that this stability function is generated by the RKC schemes. The damped stability interval of the second order RKC schemes is

$$[-\beta_R, 0], \quad \text{where} \qquad \beta_R \approx \frac{2}{3}\left(s^2 - 1\right).$$

The interval increases quadratically by s similar to the undamped version. The boundary of absolute stability is slightly less than $\frac{2}{3}\left(s^2 - 1\right)$ compared to the undamped version. Similar to the first order RKC schemes, the analytical expression of the boundary of absolute stability β_R, of the damped second order RKC schemes can be found in [15, 27]. Figures 3.11 and 3.12 show the decrease in the stability region when using damping. The slightly stretched (blue) curve in the figures shows the original stability region without damping.

3.3 RKC time stepping formulas

Now that we have obtained the desired stability functions for first and second order RKC schemes, we need an s-stage RKC scheme that produces these stability functions for a general test problem. Such a scheme is given in the following Shu-Osher form [9, 11, 21, 23]

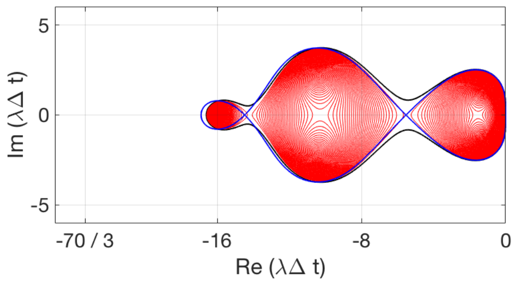


Figure 3.11: The stability region of $B_5(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping.

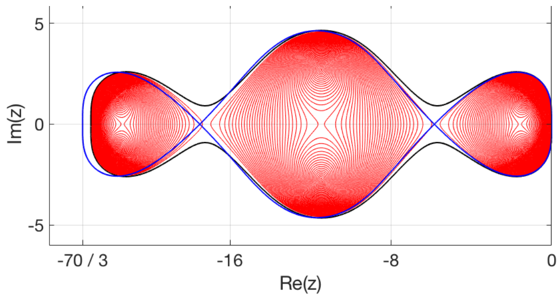


Figure 3.12: The stability region of $B_6(z)$ with damping. The slightly stretched (blue) curve shows the stability region without damping.

$$
\begin{aligned}
&Y_0 = U^n,\\
&Y_1 = Y_0 + \tilde{\mu}_1 k F_0,\\
&Y_j = \mu_j Y_{j-1} + \nu_j Y_{j-2} + (1 - \mu_j - \nu_j) Y_0 + \tilde{\mu}_j k F_{j-1} + \tilde{\gamma}_j k F_0, \qquad 2 \le j \le s,\\
&U^{n+1} = Y_s
\end{aligned}
\tag{3.23}
$$

where $F_j = F(t_n + c_j k, Y_j)$, $U^n \approx U(t_n)$ is the approximation of the solution at time t_n, $k = t_{n+1} - t_n$ is the time step, and Y_j values are the intermediate solutions at each stage j. The solution Y_j at each intermediate stage depends on the two previously computed intermediate solutions and Y_0.

The integration parameters in equation (3.23) for $2 \le j \le s$ are defined analytically as

$$\tilde{\mu_1} = b_1 w_1, \qquad \tilde{\mu_j} = 2w_1 \frac{b_j}{b_{j-1}},$$
$$\mu_j = 2w_0 \frac{b_j}{b_{j-1}}, \qquad \nu_j = -\frac{b_j}{b_{j-2}},$$
$$\tilde{\gamma_j} = -\left(1 - b_{j-1} T_{j-1}\left(w_0\right)\right) \tilde{\mu_j}.$$

These parameters are specifically designed such that the stability functions of the RKC schemes are recovered. In addition, the parameter choices depend on the desired order of accuracy.

For first order RKC schemes, we use the expressions for w_0 and w_1 obtained earlier,

$$w_0 = 1 + \frac{\epsilon}{s^2}, \qquad \epsilon = 0.05, \qquad w_1 = \frac{T_s\left(w_0\right)}{T_s'\left(w_0\right)}.$$

We can derive the expressions for b_j for intermediate stages in a manner analogous to how we derived the b_s values. We assume that the intermediate stages must have the RKC representation in the following form

$$P_j\left(z\right) = a_j + b_j T_j\left(w_0 + w_1 z\right), \qquad 0 \le j < s.$$

For consistency and first order accuracy, we have,

$$P_j\left(0\right) = a_j + b_j T_j\left(w_0\right) = 1, \quad \text{and} \quad P_j'\left(0\right) = b_j w_1 T_j'\left(w_0\right) = 1.$$

Similar to a_s in equation (3.16), we set $a_j = 0$ for the first order RKC schemes, and solve for b_j to get

$$P_j(0) = b_j T_j(w_0) = 1 \implies b_j = \frac{1}{T_j(w_0)}, \qquad 0 \le j < s.$$

Therefore, all of the b values are defined as

$$b_j = \frac{1}{T_j(w_0)}, \qquad 0 \le j \le s.$$

The time increment parameters in the first order RKC schemes are given by [21, 27, 28] as

$$\begin{aligned} c_0 &= 0, \\ c_j &= \frac{T_s(w_0)}{T_s'(w_0)} \frac{T_j'(w_0)}{T_j(w_0)}, \qquad 1 \le j \le s-1, \\ c_s &= 1. \end{aligned}$$

For each stage, the stage time t_j is given by $t_n + c_j k$.

Similarly, for the second order RKC schemes, we use the expressions for w_0 and w_1 obtained earlier,

$$w_0 = 1 + \frac{\epsilon}{s^2}, \qquad \epsilon = \frac{2}{13}, \qquad w_1 = \frac{T_s'(w_0)}{T_s''(w_0)}.$$

We can derive the expressions for b_j for intermediate stages in a manner analogous to how we derived the b_s values. We assume that the intermediate stages must have the RKC representation in the form

$$B_j(z) = a_j + b_j T_j(w_0 + w_1 z), \qquad 2 \le j < s. \tag{3.24}$$

For consistency and second order accuracy, we have

$$B_j(0) = a_j + b_j T_j(w_0) = 1, \qquad B_j'(0) = b_j w_1 T_j'(w_0) = 1, \qquad B_j''(0) = b_j w_1^2 T_j''(w_0) = 1.$$

We solve for a_j and b_j to get

$$\begin{aligned} a_j &= 1 - \frac{T_j''(w_0)}{\left(T_j'(w_0)\right)^2} T_j(w_0), \qquad 0 \le j < s, \\ b_j &= \frac{T_j''(w_0)}{\left(T_j'(w_0)\right)^2}, \qquad 2 \le j < s. \end{aligned}$$

Note that b_0 and b_1 are free parameters. In the literature, they are chosen to be equal to b_2 [22]. Therefore, all of the b values are defined as

$$\begin{aligned} b_0 &= b_2, \qquad b_1 = b_2, \\ b_j &= \frac{T_j''(w_0)}{\left(T_j'(w_0)\right)^2}, \qquad 2 \le j \le s. \end{aligned}$$

The time increment parameters in the second order RKC schemes are given by [21, 27, 28]

$$\begin{aligned} c_0 &= 0, \qquad c_1 = \frac{c_2}{T_2'(w_0)}, \\ c_j &= \frac{T_s'(w_0)}{T_s''(w_0)} \frac{T_j''(w_0)}{T_j'(w_0)}, \qquad 2 \le j \le s-1, \\ c_s &= 1. \end{aligned}$$

Again, for each stage, the stage time t_j is given by $t_n + c_j k$.

It is challenging to analytically show that the RKC schemes in equation (3.23) generate the damped stability functions of the first and second order RKC schemes. Instead, we show this numerically by computing one time step using the RKC schemes, and one time step using the damped stability functions of the first and second order RKC schemes. We solve the Dahlquist test problem, $U'(t) = \lambda U(t)$, $U(0) = 1$ using $\lambda = -10$, $k = 0.001$, and we set the number of stages to vary from 1 to 15. The exact solution at this time step is equal to $U^{n+1} = 0.9900498337491681$. Note that in Tables 3.1 and 3.2, the difference between taking one time step using the RKC schemes and taking one time step using the stability functions is near machine precision. Thus, the formulas in equation (3.23) in fact recover the damped stability functions of first and second order RKC schemes in equations (3.19) and (3.22), respectively.

s	U^{n+1} From RKC	$U^{n+1} = P_s(z)\,U^n$	Relative diff.
4	0.9900160206759102	0.9900160206759130	2.80×10^{-15}
5	0.9900164066850475	0.9900164066850554	7.96×10^{-15}
6	0.9900166163993918	0.9900166163993891	2.69×10^{-15}
7	0.9900167428608667	0.9900167428608604	6.39×10^{-15}
8	0.9900168249433932	0.9900168249433960	2.92×10^{-15}
9	0.9900168812207998	0.9900168812208083	8.63×10^{-15}
10	0.9900169214766443	0.9900169214766517	7.40×10^{-15}
11	0.9900169512619237	0.9900169512618967	2.73×10^{-14}
12	0.9900169739163572	0.9900169739163640	6.84×10^{-15}
13	0.9900169915470152	0.9900169915470188	3.70×10^{-15}
14	0.9900170055365031	0.9900170055364751	2.83×10^{-14}
15	0.9900170168225733	0.9900170168226244	5.17×10^{-14}

Table 3.1: Comparing one time step solved with a first order RKC scheme and one time step solved using a stability function. Parameters: s is the number of stages, and $P_s(z)$ is the stability function of first order RKC schemes.

3.4 Efficiency of the RKC schemes

The number of time steps taken in a time stepping scheme can be important for efficiency considerations. For example, for the Forward Euler scheme, the number of time steps M, is a large number since we must take a large number of time steps

s	U^{n+1} From RKC	$U^{n+1} = B_s(z)\,U^n$	Relative diff.
4	0.9900499191391322	0.9900499191391311	1.12×10^{-15}
5	0.9900499115710127	0.9900499115710123	4.49×10^{-16}
6	0.9900499076074784	0.9900499076074787	3.36×10^{-16}
7	0.9900499052656166	0.9900499052656088	7.96×10^{-15}
8	0.9900499037645023	0.9900499037645024	1.12×10^{-16}
9	0.9900499027437906	0.9900499027437789	1.18×10^{-14}
10	0.9900499020178435	0.9900499020178370	6.50×10^{-15}
11	0.9900499014829413	0.9900499014829345	6.84×10^{-15}
12	0.9900499010773610	0.9900499010773606	4.49×10^{-16}
13	0.9900499007624769	0.9900499007624618	1.53×10^{-14}
14	0.9900499005130893	0.9900499005130815	7.85×10^{-15}
15	0.9900499003121912	0.9900499003122032	1.21×10^{-14}

Table 3.2: Comparing one time step solved with a second order RKC scheme and one time step solved using a stability function. Parameters: s **is the number of stages, and** $B_s(z)$ **is the stability function of second order RKC schemes.**

because of the small time step size k. Due to the stiffness of the heat problem, the small time steps k of the Forward Euler scheme are to maintain stability rather than to improve accuracy. In this section we compare the efficiency of the RKC schemes with the Forward Euler scheme in terms of the number of time steps. As explained in Chapter 2, we can find the suitable time step size based on the stability of the scheme. For the Forward Euler scheme, we have, a severe restriction on the time step size,

$$k_{FE} \leq \frac{h^2}{2}.$$

For the purposes here, we will then assume that $k_{FE} = \dfrac{h^2}{2}$. We can compute the number of time steps M by dividing the final time T by the time step size k_{FE},

$$M_{FE} = \frac{T}{k_{FE}} = \frac{2T}{h^2}. \tag{3.25}$$

For the RKC schemes, we assume that $k_{RKC} = h$. To compute the total number of time steps, we also take into account the number of stages required, which is obtained by

$$s = \sqrt{\frac{\rho(A)\, k_{RKC}}{\sigma}}, \tag{3.26}$$

where $\rho(A)$ is the spectral radius of the MOL discretization matrix of the heat equation, and σ is a positive scalar [28]. For the first order RKC schemes, the literature suggests $\sigma = 1.9$, and for the second order RKC schemes, $\sigma = 0.653$ [28]. The spectral radius is

$$\rho(A) \approx \left|\frac{-4}{h^2}\right| \implies s = \sqrt{\frac{4\ k_{RKC}}{h^2\ \sigma}}.$$

Assuming $k_{RKC} = h$, we can compute the number of stages,

$$s = \sqrt{\frac{4\ h}{\sigma\ h^2}} = \frac{2}{\sqrt{\sigma h}}.$$

The total number of steps taken is then

$$M_{RKC} = s\frac{T}{k_{RKC}} = s\frac{T}{h} = \frac{2\ T}{\sqrt{\sigma h^3}}. \tag{3.27}$$

By comparing equations (3.25) and (3.27), we observe that the number of time steps to be taken by the RKC schemes is smaller than that of Forward Euler by a factor of $\sqrt{\frac{h}{\sigma}}$:

$$\frac{M_{RKC}}{M_{FE}} = \sqrt{\frac{h}{\sigma}} \implies M_{RKC} = \sqrt{\frac{h}{\sigma}} M_{FE} \ll M_{FE}.$$

For small values of h, the RKC schemes will generally require much fewer time steps than the Forward Euler scheme.

3.5 Numerical results of the RKC schemes

Consider the following example of the heat equation without a heat source term

$$U_t(x,t) = U_{xx}(x,t), \tag{3.28}$$

with inhomogeneous Dirichlet boundary conditions,

$$U(0,t) = U(1,t) = e^{-4\pi^2 t}$$

and the initial condition

$$U(x,0) = \cos(2\pi x).$$

The exact solution to this problem is

$$U(x,t) = e^{-4\pi^2 t}\cos(2\pi x).$$

We solve this problem using the first and second order RKC schemes and show the results in Tables 3.3 and 3.4. As expected, the estimated numerical rate of convergence is 1 for the first order RKC schemes, and 2 for second order RKC schemes.

In the following figures, we compare the results of the first and second order RKC schemes and the Forward Euler scheme. Figure 3.13 shows how error decreases as we reduce the spatial grid size h. The numerical convergence rates for both the second order RKC schemes and the Forward Euler scheme are second order, and first order for the first order RKC schemes. Formally, the Forward Euler scheme is first order; however, due to the severe time step size restrictions, it gains a second order accuracy when it is used to numerically solve the parabolic heat equation.

N	s	Error	Time (s)	Rates
32	8	6.8552×10^{-2}	1.1082×10^{-2}	–
64	11	2.9187×10^{-2}	6.9333×10^{-3}	1.23
128	15	1.4373×10^{-2}	1.0491×10^{-2}	1.02
256	20	6.9592×10^{-3}	1.5902×10^{-2}	1.05
512	28	3.4508×10^{-3}	2.5712×10^{-2}	1.01
1024	39	1.7167×10^{-3}	7.5811×10^{-2}	1.01

Table 3.3: Errors from the problem given in equation (3.28) using the first order RKC schemes. The last column shows the convergence rate of the scheme.

N	s	Error	Time (s)	Rates
32	13	1.3066×10^{-2}	1.7781×10^{-3}	–
64	17	2.7318×10^{-3}	3.3888×10^{-3}	2.26
128	24	6.7761×10^{-4}	5.3779×10^{-3}	2.01
256	33	1.4934×10^{-4}	1.4656×10^{-2}	2.18
512	46	3.7002×10^{-5}	4.3555×10^{-2}	2.01
1024	65	9.1346×10^{-6}	1.2392×10^{-1}	2.02

Table 3.4: Errors from the problem given in equation (3.28) using the second order RKC schemes. The last column shows the convergence rate of the scheme.

Figure 3.14 shows that the amount of computational work (in time) increases as the number of grid cells N, increases. For a given N, the amount of work required by both RKC schemes is lower, and increasing at a lower rate than the Forward Euler scheme. The amount of computational work (in time) required for the first order RKC schemes is lower compared to the second order RKC schemes because it takes fewer stages.

Figure 3.15 shows that the amount of computational work (in time) increases as the error decreases. We observe that the second order RKC schemes require the least amount of computational work for a given level of accuracy. The first order RKC

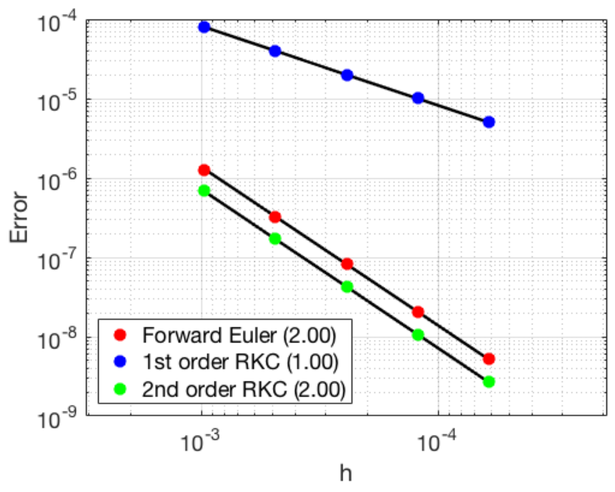


Figure 3.13: The plot of error versus the spatial grid size h: as h decreases, the error decreases as well. The Forward Euler and second order RKC schemes are second order, and the first order RKC scheme is first order. The numbers in the parentheses show the numerical convergence rates.

schemes are not preferable in this case, since the computational work increases at the highest rate (2.54 in Figure 3.15) to achieve a better accuracy.

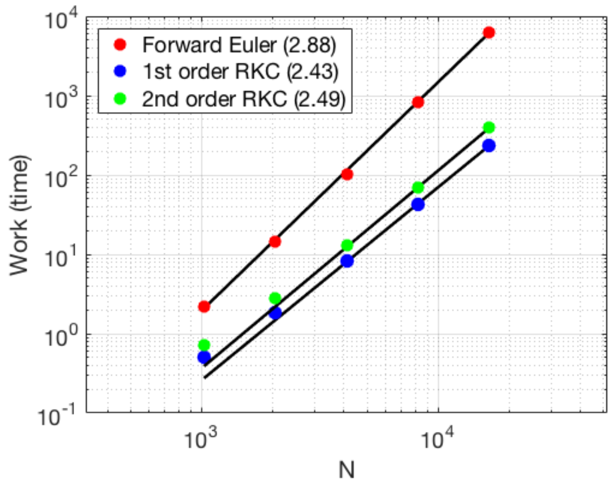


Figure 3.14: The plot of computational work (in time) versus the number of grid cells N: as N increases, the computational work increases as well. The amount of work required for both RKC schemes is lower and increasing at a rate slower than the Forward Euler scheme. The numbers in the parentheses show the numerical convergence rates.

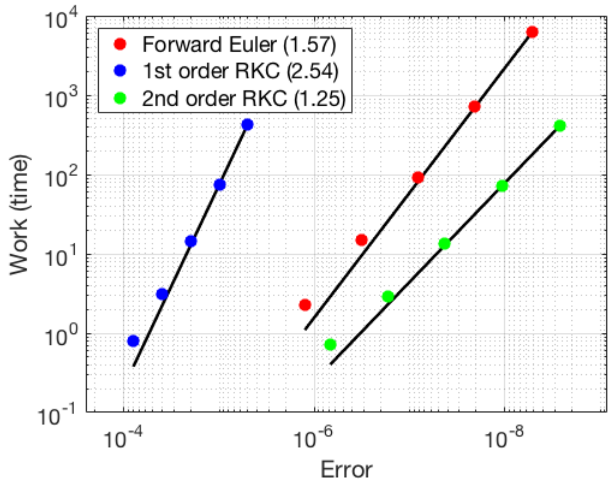


Figure 3.15: The plot of work (in time) versus the error: the computational cost increases as the error decreases. For better accuracy, more computational work is needed. The convergence rates in parentheses show that the second order RKC schemes are the most efficient, while the first order RKC schemes are the least efficient.

Chapter 4

MULTI-RATE RKC TIME STEPPING ON ADAPTIVELY REFINED MESHES

The solutions to the heat or diffusion equations is typically smooth; however, for problems involving heat source terms, the solutions could produce sharp fronts. For the spatial discretization, we could use a uniformly fine mesh, where we will be restricted by the smallest grid resolution h required on the sharp fronts. This approach could be computationally expensive and inefficient, because we could use a larger resolution in areas where the solution is smooth, yet obtain an acceptable level of accuracy. An alternative approach is to take advantage of adaptive mesh refinement (AMR) on the spatial domain to lower the computational cost while producing comparable results to those on uniformly fine grids. In this approach, we only refine the mesh in regions of the domain where more resolution is needed to better resolve any sharp fronts. This is particularly appropriate for stiff problems with sharp gradients which are present only in some parts of the domain. As an example, Figure 4.1 demonstrates a two dimensional adaptively refined mesh, where each rectangle represents a grid, and the resolution of the grids increase as we approach the interface (blue swirl) in the figure.

When using time stepping schemes, if we take the same global time step on all of the grids with different resolutions, we will be restricted by the smallest time

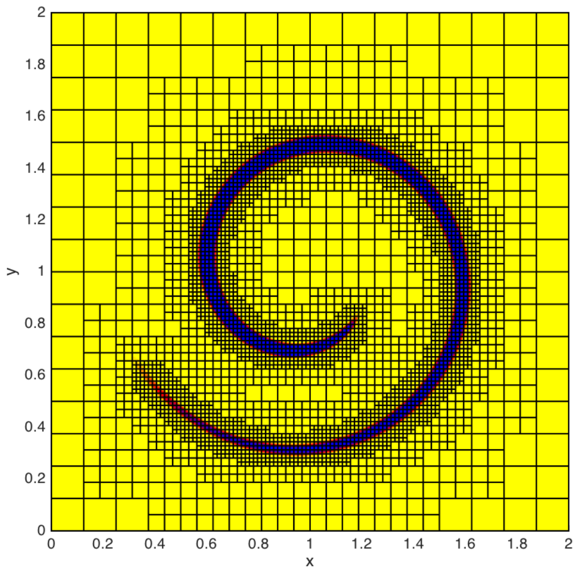


Figure 4.1: An exmple of a two dimensional adaptively refined mesh, where each rectangle represents a grid. The resolution of the grids increases as we get closer to the interface (the blue swirl).

step which is required on the finest grid due to stability and accuracy considerations. Similar to the argument about spatial refinement, if we take the same global time step on all of the refined grids, our time stepping scheme is likely to be computationally expensive and inefficient, since we are potentially wasting computational effort on areas where we could take larger time steps. An alternative approach is to devise a scheme where each refined grid is able to take a time step that is appropriate for its spatial grid size. Such a scheme is referred to as a "multi-rate time stepping scheme" [14, 19]. The multi-rate scheme allows us to take different local time steps over regions of the domain with different spatial resolutions. In a multi-rate scheme, we would like the time step k to be equal or proportional to the spatial grid size h, so that we have spatial and temporal refinement simultaneously. Implementing the

RKC schemes using multi-rate time stepping requires coordinating the stages across multiple grids and grid resolutions. In this thesis, we have developed an algorithm that carries out this coordination on a one dimensional adaptively refined grid.

4.1 Spatial refinement using tree data structures

In order to store multiple grids with different resolutions, we can use a tree data structure in which data are organized in nodes that can have zero or more children nodes. The first node is the root and has children, but it does not have any parents or siblings. Nodes that have no children are called leaves. Since a tree is not a linear array, there are many ways to traverse through the tree and analyze the nodes. In order to iterate through the tree to parse the nodes, we can use a depth-first approach. Once we meet a node, we parse all of its children and sub-children before parsing its siblings. In other words, once we start from the root through the first branch, we do not stop traversing until we have reached the leaf of that branch; then, we go up one level and repeat the process until we are done with all of the branches.

In order to store our grids, we use a binary tree, which is a specific type of tree where each node has zero or two children, and their position is either left or right. In Figure 4.2 we show a binary tree which includes one head, 31 total number of nodes, and 16 leaves. Each node in the tree has a bit index consisting of 0s and 1s which uniquely encodes its position in the tree. We identify different "levels" of spatial refinement by the position of the leaves. Each time a node splits into two children, we have a new level of refinement. For example, in Figure 4.2, we have five levels of refinement. The first level called level 0, includes the head of the tree, and level 1 includes the two children of the head node (nodes 10 and 11).

In order for the grids stored in the leaves to exchange ghost cell values (boundary conditions on each grid), we must know their positions on the tree. In order to link the leaves together, we create a doubly linked list of the leaves. Each leaf in the doubly linked list contains information about the data array (which contains the grid), a link to the next leaf, and a link to the previous leaf. After each node splits into two children nodes, the linked list adds the children to the leaf list, and deletes the parent node from the leaf list. As we use the leaf list to exchange ghost cell values between the grids, depending on the neighboring grids' levels, the cell values might be averaged, copied, or interpolated to obtain the ghost cell values of the current grid we are on. This will be discussed in more detail in Section 4.4.

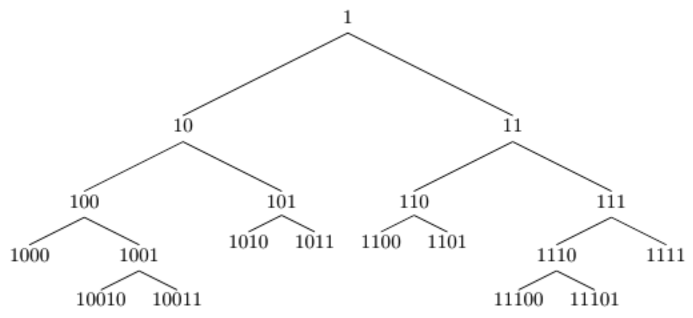


Figure 4.2: A binary tree data structure. The $0-1$ bit index uniquely locates each node in the tree structure.

4.2 Temporal refinement using multi-rate time stepping

After refining in space, we would like to refine in time by using a multi-rate time stepping scheme. As mentioned earlier, we would like to implement the RKC schemes using multi-rate time stepping, and so we must coordinate the stages with different time levels across multiple grids. In order to visually demonstrate the multi-rate scheme and how we coordinate the stages, we plot the stages of RKC for one time step in Figure 4.3. Assume that we have two levels of spatial and temporal refinement: a coarse grid of level 0, which takes one full time step k from time t_n to time t_{n+1}, and a fine grid of level 1, which takes two half time steps $\frac{1}{2}k$ from t_n to $t_{n+\frac{1}{2}}$, and from $t_{n+\frac{1}{2}}$ to t_{n+1} to complete one full time step. The dashed line shows where the half time step $t_{n+\frac{1}{2}}$ is in time. Similar to the time steps, the spatial resolution on level 1 is twice as much as the spatial resolution on level 0.

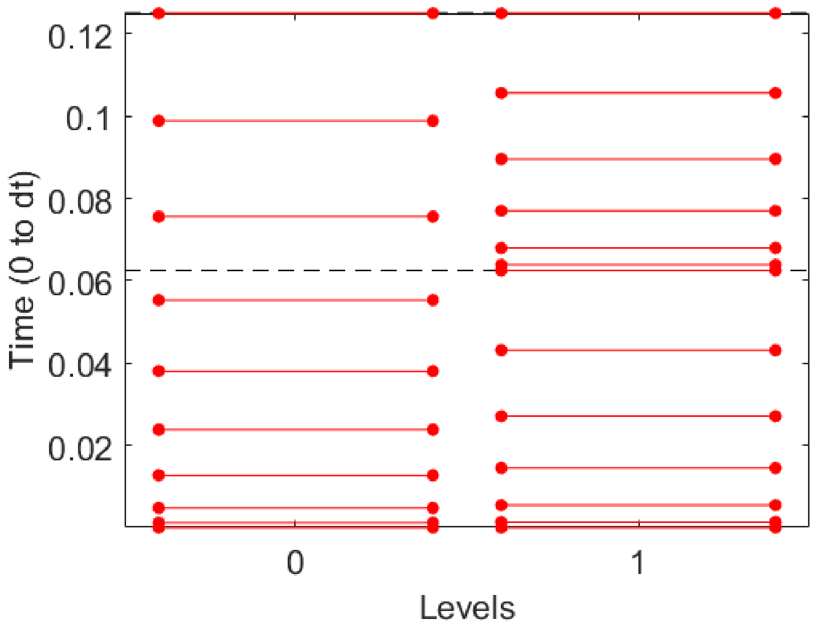
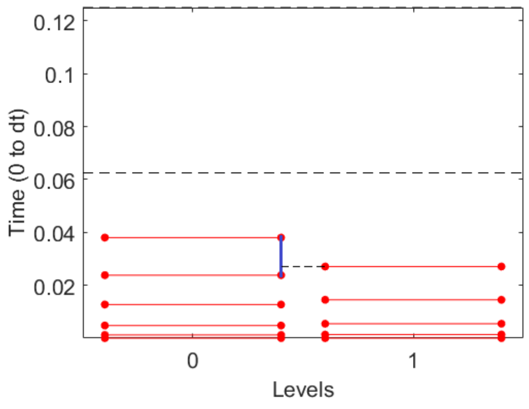
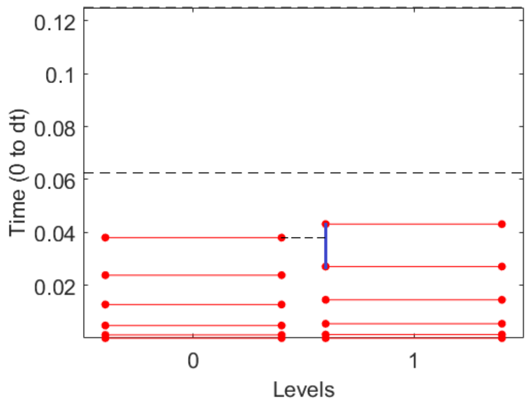


Figure 4.3: Multi-rate time stepping on two levels of refinement: a coarse grid of level 0, which takes one full time step, and a fine grid of level 1, which takes two half time steps. The spatial resolution on level 1 is twice as much as the spatial resolution on level 0.

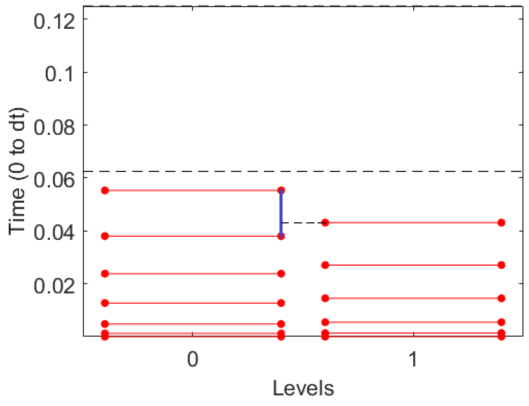
In the beginning the two levels are time synchronized at t_n, so they have the required ghost cell values to advance one stage. After taking the first stage on each grid, the stages become unsynchronized because they do not advance in equal time increments. This is because the distribution of the time increments c_j depend on the number of stages s, which in turn depends on the spectral radius of the discretization matrix and the time step size. In one time step, level 0 completes 9 stages and level 1 completes two sets of 6 stages. Therefore, we encounter a problem: the stages are not time synchronized and cannot advance without valid ghost cell values.



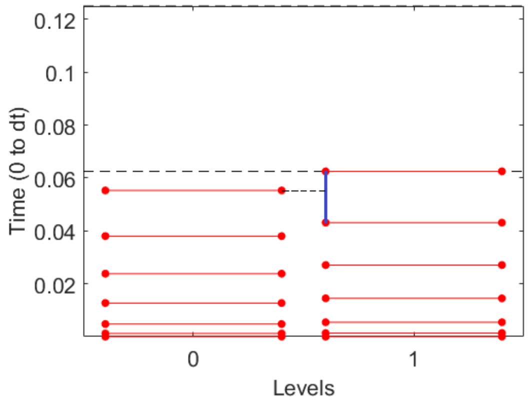
(a) Interpolation in time on level 0



(b) Interpolation in time on level 1



(c) Interpolation in time on level 0



(d) Interpolation in time on level 1

Figure 4.4: Obtaining ghost cell values on each stage by linearly interpolating in time using the adjacent grids' stages while using multi-rate time stepping on two levels of refinement.

In order to obtain the ghost cell values, we linearly interpolate in time using the adjacent grids' stages. For example, in Figure 4.4a, level 1 cannot advance because of the lack of ghost cell values on the left. So we linearly interpolate in time on the right end values of the two stages of level 0, and obtain the ghost cell value for level 1 in Figure 4.4a. Thus, level 1 advances a stage as shown in Figure 4.4b. Now, the right end ghost cell value of level 0 is not available (Figure 4.4b). By linearly interpolating on the left end values of the stages in level 1, we obtain the ghost cell value, and level 0 advances a stage as shown in Figure 4.4c. As long as any stage has valid ghost cell values, it can advance in time. Only when it has advanced beyond one or two of its neighbors grids, does it have to wait to fill in its ghost values. This process continues until we complete the stages on both grids for one complete time step in Figure 4.3.

As mentioned earlier, each of the horizontal lines in Figure 4.3 represents an RKC stage. The computation of each stage requires some amount of work in terms of computational time, or the number of evaluations of the right hand side function. From the previous example with two levels, it may not seem that there is much to be gained using multi-rate time stepping: we computed 21 stages with multi-rate time stepping compared to 24 stages with global time stepping. However, an example with more levels of refinement demonstrates that multi-rate time stepping can lead to significant performance gains. With four levels of refinement as in Figure 4.5, we compute 97 stages with multi-rate time stepping, instead of 160 stages with global time stepping. Thus, multi-rate time stepping computes the results with significantly less computational cost.

In the above performance analysis, we assume that each level does the same amount of work, i.e. has the same number of grids. This might not always be true, so it is important to consider adaptive refinement patterns, some of which are more suited to take advantage of multi-rate time stepping than others. But to fully evaluate the effectiveness of the multi-rate time stepping, we would need to do further

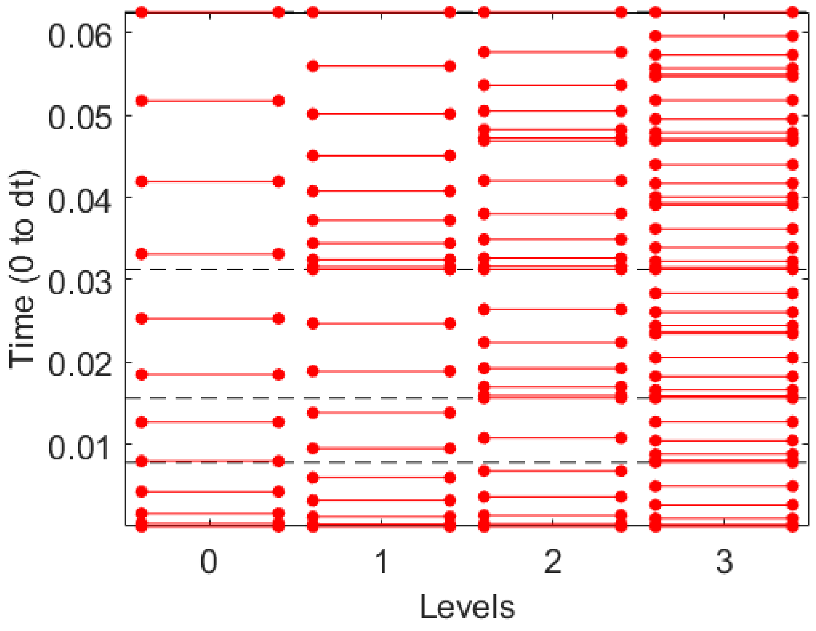


Figure 4.5: Performance gains using multi-rate time stepping on four levels of refinement: the number of stages computed by multi-rate time stepping is 97 as opposed to global time stepping with 160 stages.

analysis.

4.3 Heat equation with AMR and multi-rate time stepping

We consider the following example of the heat problem with a source function

$$U_t(x,t) = U_{xx}(x,t) + S(x,t), \qquad x \in (0,1), \quad t \in (0,T), \tag{4.1}$$

where we assume that the solution to this problem has the following form

$$U(x,t) = P(t)\,Q(x). \tag{4.2}$$

We have chosen the following equations for $P(t)$ and $Q(x)$:

$$Q(x) = T\left(\frac{x-0.25}{c}\right) - T\left(\frac{x-0.75}{c}\right),$$
$$P(t) = e^{-t},$$

where $T(x) = \frac{1}{2}(\tanh(x)+1)$, and c is a small positive scalar $c \ll 1$ (small c produces a sharp square wave). To compare the numerical solution to the exact solution, we compute the source function as,

$$S(x,t) = P'(t)\,Q(x) - Q''(x)\,P(t).$$

We have chosen the heat source term based on the exact solution in equation (4.2). This numerical verification approach is called the "method of manufactured solutions". This choice of $S(x,t)$ causes the solution to maintain its sharp fronts as the heat diffuses over time. The initial condition of the PDE in equation (4.1) is:

$$U(x,0) = P(0)\,Q(x) = T\left(\frac{x-0.25}{c}\right) - T\left(\frac{x-0.75}{c}\right),$$

and the Dirichlet boundary conditions are:

$$U(0,t) = P(t)\,Q(0),$$
$$U(1,t) = P(t)\,Q(1).$$

The plot of the solution is shown in Figure 4.6 where the grids on the sharp fronts are refined. Figure 4.7 shows the tree structure and the composite grid.

In order to exchange ghost cell values for the finite volume discretization, we copy,

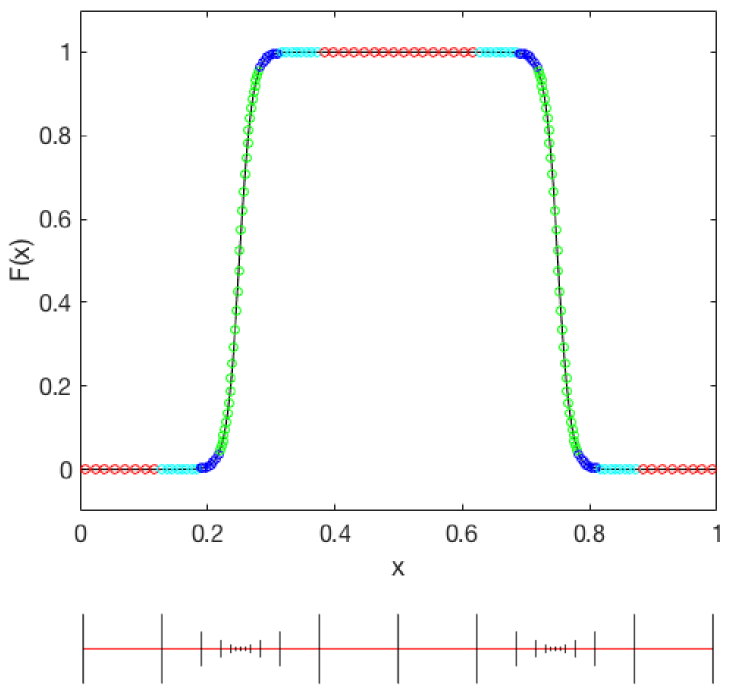


Figure 4.6: One dimensional refinement pattern for a regularized square wave of radius 0.25 centered at $x = 0.5$. The bottom plot is the composite grid of the problem with refinement

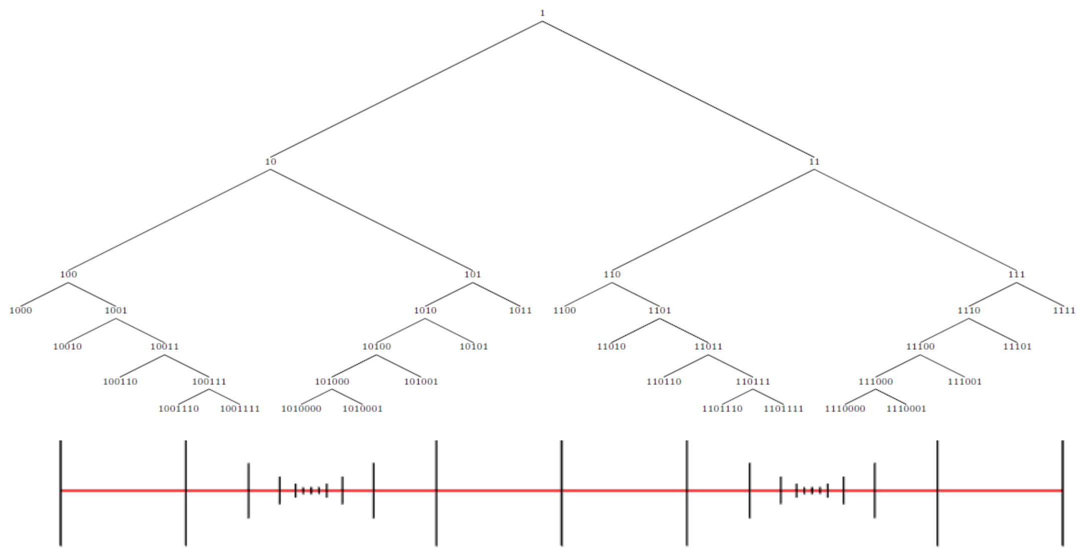


Figure 4.7: The binary tree structure of the grid in Figure 4.6 with the composite grid that shows the refinement pattern.

average, or interpolate the cell values from the adjacent grids. If two adjacent grids have the same resolution, we copy the ghost cell values from the interior of one grid to the ghost cells of the adjacent grid. If two adjacent grids have different resolutions, we average or interpolate the cell values to obtain the ghost cell values. If the neighboring grid is a fine grid, we average its two cell values to obtain the ghost cell value for the coarse grid. If the neighboring grid is a coarse grid, we interpolate coarse grid values to obtain the ghost cell value for the fine grid. This process will be discussed in detail in Section 4.4.

In Figure 4.8, we compare the computational time versus the number of evaluations of the right hand side function by solving the heat problem in equation (4.1) using AMR with multi-rate time stepping. Since the relationship is linear, we conclude that we can use the number of evaluations of the right hand side function as a measure of computational work.

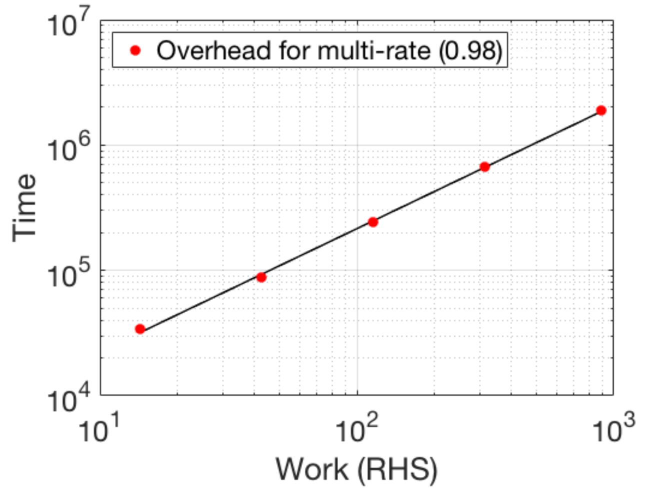


Figure 4.8: The linear relationship of the computational time and the number of evaluations of the right hand side function. This shows that the computational time does not involve overhead.

We solve the heat problem in equation (4.1) with three different approaches: uniformly refined mesh with global time stepping (no adaptivity), AMR using global time stepping (adaptivity in space only), and AMR using multi-rate time stepping (fully adaptive). For global time stepping, we compute using a global time step size which is restricted by the smallest time step required on the finest grid. To do a convergence study, we fix the number of grids, and increase the resolution of each grid. We define the "effective resolution" of the whole domain as the resolution of the finest level grid, (i.e. as if the domain were uniformly refined). An effective h value is the length of the domain divided by the effective resolution.

In Table 4.1, as we increase the spatial resolution, the error decreases, and the number of evaluations of the right hand side function increases. As expected, using a uniformly refined mesh with global time stepping has the smallest error since we have fully refined in both space and time; however, it is the most computationally expensive approach. Using AMR and global time stepping reduces the computational cost by about $\frac{1}{3}$, and using AMR and multi-rate time stepping, reduces the computational cost by about $\frac{1}{6}$ compared to the uniformly refined case. Using AMR and multi-rate time stepping reduces computational cost by half compared to using AMR and global time stepping. Therefore, as expected, using AMR and multi-rate time stepping produces comparable results with significantly lower computational cost.

Time-step./Mesh	Eff. N	Res.	# grids	Error	RHS	Rate	Ratio
	512	8	64	2.75×10^{-4}	193 344	–	1.00
Global/	1024	16	64	6.94×10^{-5}	532 480	1.98	1.00
Uniform	2048	32	64	1.73×10^{-5}	1 489 792	2.00	1.00
	4096	64	64	4.34×10^{-6}	4 208 640	1.99	1.00
	8192	128	64	1.08×10^{-6}	11 822 400	2.00	1.00
	512	8	20	1.84×10^{-3}	66 720	–	0.35
Global/	1024	16	20	4.48×10^{-4}	172 760	2.04	0.32
Adaptive	2048	32	20	1.11×10^{-4}	480 060	2.02	0.32
	4096	64	20	2.74×10^{-5}	1 332 420	2.01	0.32
	8192	128	20	6.84×10^{-6}	3 729 440	2.00	0.32
	512	8	20	1.91×10^{-3}	33 856	–	0.18
Multi-rate/	1024	16	20	4.42×10^{-4}	87 528	2.10	0.16
Adaptive	2048	32	20	1.10×10^{-4}	242 784	2.00	0.16
	4096	64	20	2.73×10^{-5}	674 160	2.01	0.16
	8192	128	20	6.77×10^{-6}	1 884 896	2.01	0.16

Table 4.1: Comparing uniform versus adaptively refined grids, as well as comparing global versus multi-rate time stepping. The uniform mesh has a fixed number of levels (minlevel=maxlevel=6) while the adaptive mesh varies in refinement levels (minlevel=3, maxlevel=6).

The results of Table 4.1 are also displayed in Figures 4.9 - 4.12. Figure 4.9 shows that as we decrease the grid size h, the error decreases as well. We also see that the multi-rate time stepping is comparable to global time stepping as their plots overlap, and their rates of convergence are both very close to 2. On the other hand, the uniform mesh also has the same rate of convergence; however, its accuracy is better because we are solving the problem on a uniformly refined mesh.

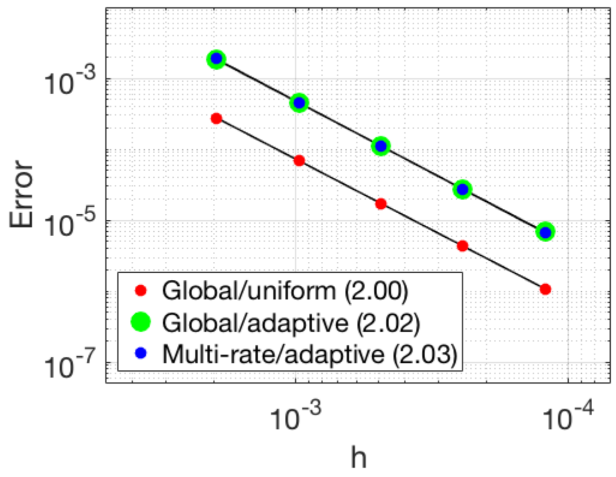


Figure 4.9: The plot of error versus the effective grid size h. As we decrease the grid size h, the error decreases as well. Multi-rate time stepping produces comparable results compared to global time stepping.

In Figure 4.10, as we increase the effective grid resolution N, the amount of computational work (number of right hand side evaluations) increases. The three schemes have similar convergence rates; however, the computational work for the multi-rate scheme is lower than the others.

In Figure 4.11, we compare the amount of computational work required to improve accuracy. As the error decreases, the amount of computational work increases almost at the same rate for all of the schemes. For a certain error, the uniformly refined approach might be an acceptable choice, since it is slightly better than the AMR with global time stepping approach; however, the multi-rate approach produces the most satisfactory results.

In Figure 4.12, we compare the three approaches in a different way. On the

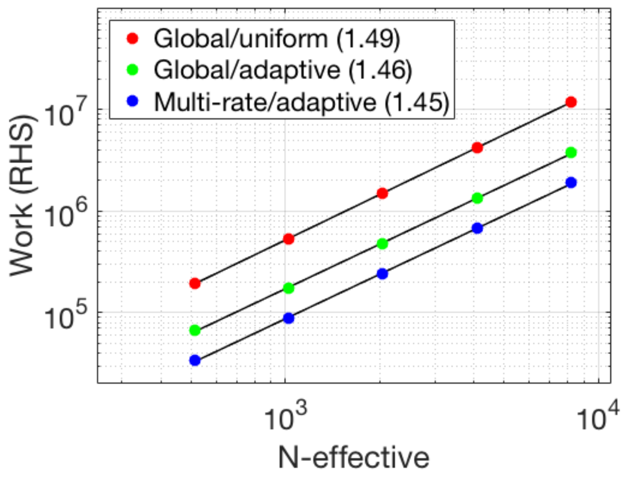


Figure 4.10: The plot of computational work (number of evaluations of the RHS function) versus the effective grid resolution N. As we increase the effective N, the computational work increases. The multi-rate scheme takes far less computational work.

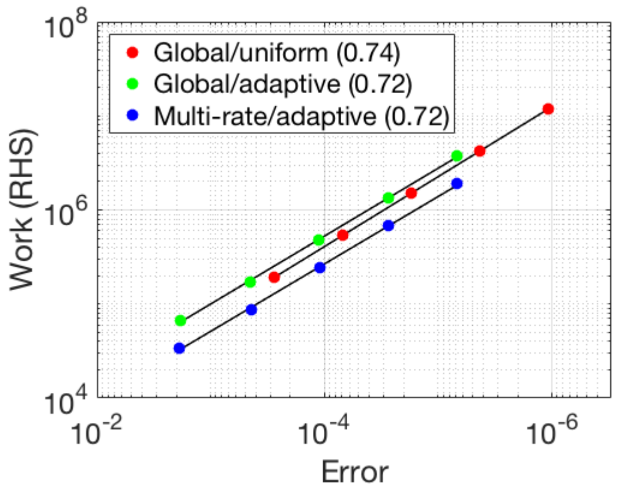


Figure 4.11: The plot of computational work (number of evaluations of the RHS) versus the error. The computational cost increases as the error decreases. The multi-rate approach with AMR has the smallest error.

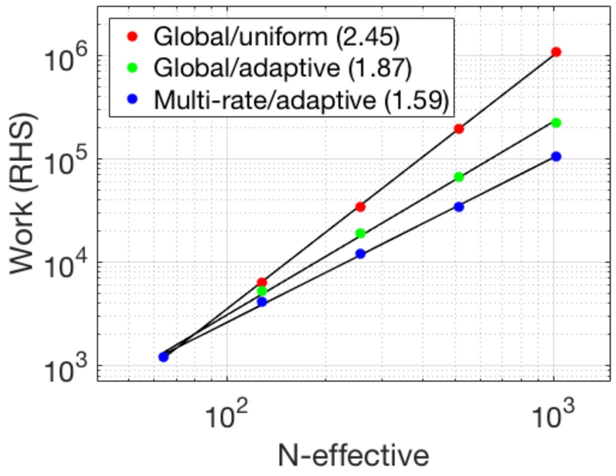


Figure 4.12: The plot of computational work (number of evaluations of the RHS) versus the effective N. In the uniformly refined approach, we increase the levels from $3-3$ to $6-6$ (on the uniform grid, the minimum and maximum levels are equal). On the AMR grids, we increase the maximum level each time from $3-3$ to $3-6$. The multi-rate approach has the smallest rate of increase.

uniformly refined grid, we increase the refinement levels from $3-3$ to $6-6$ (the minlevel and maxlevel are equal). On the adaptive grids, we increase the refinement levels from level $3-3$ to $3-6$ (increasing the maximum refinement level from 3 to 6). In other words, we start by a uniformly refined grid, and then increase the refinement levels. The multi-rate approach has the smallest rate of increase.

4.4 Interpolation schemes on the spatial grids

As mentioned earlier, the grids must communicate their ghost cell values at each time step. Based on the initial refinement pattern, we average, copy, or interpolate the cell values on the adjacent grid to obtain the ghost cell values. In this section, we compare three different ways of interpolation. In the first scheme, we interpolate pointwise quadratically using three values from the coarse grid. In Figure 4.13, the two cell center values on the fine grid are averaged to obtain the ghost cell value on the coarse grid; then, this value along with the two neighboring cell values on the coarse grid are used in an interpolation stencil to fill in the ghost value on the fine grid.

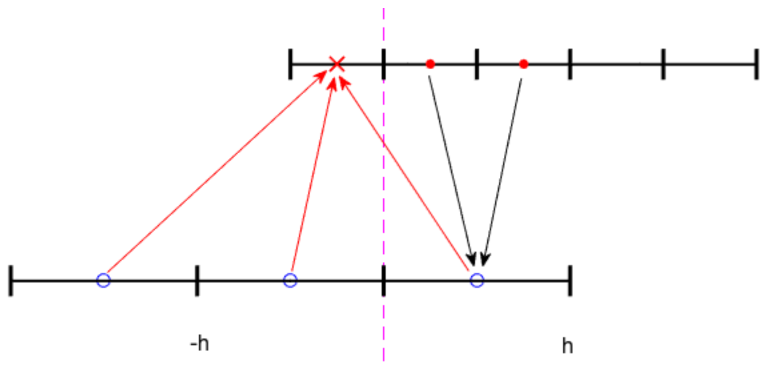


Figure 4.13: Pointwise interpolation using coarse grids. The average value of the two cell values on the fine grid is computed to fill in the ghost cell value on the coarse grid. Then, this value is used with two other cell values on the coarse grid to interpolate quadratically and obtain a ghost cell value on the fine grid.

In a second interpolation scheme, we again use a pointwise stencil, but this time, we use cell values from both the fine and coarse grids. We use two cell values from the fine grid directly along with one cell value from the coarse grid. In Figure 4.14, the arrows show the cell values that are used to compute the pointwise interpolation.

Finally, a third approach is to use conservative interpolation. In this interpolation scheme, we calculate the average values of the coarse grid by enforcing the condition that the integral of the interpolating polynomial equals the interpolated value. In this approach, we are viewing our solution values as average values, not just pointwise values. Figure 4.15 shows the cells that are used to compute the integral. For example, we can compute the integral to obtain the average value on the cell from $-h$ to 0 on

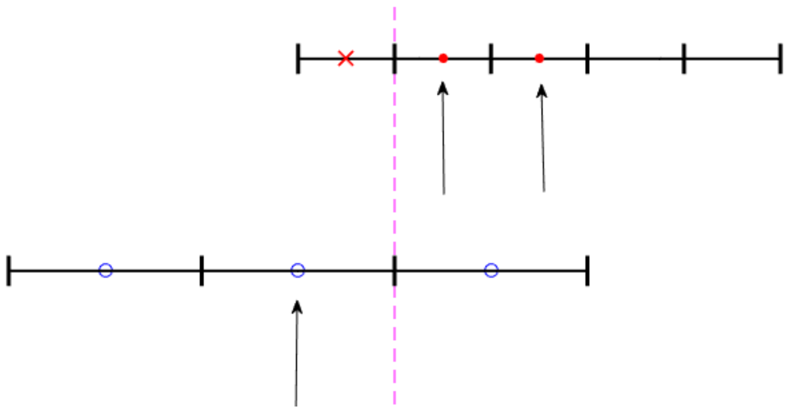


Figure 4.14: Pointwise interpolation using two cell values from the fine grid and one cell value from the coarse grid. The arrows show the cell values used for the pointwise interpolation.

the coarse grid, from 0 to $\frac{h}{2}$, and from $\frac{h}{2}$ to h on the fine grid. For example, on the coarse grid, we compute the integral

$$\frac{2}{h}\int_{-\frac{h}{2}}^{0}\left(a_2x^2 + a_1x + a_0\right)dx \equiv U\left(-\frac{h}{4}\right).$$

After we compute the integral over the required intervals, we can create a system and compute the interpolation coefficients a_2, a_1, a_0.

We compare these three interpolation schemes in Table 4.2. As the resolution increases, the error decreases in all of the interpolation schemes; however, the pointwise interpolation schemes converge at a rate closer to 1, than 2. On the other hand, the conservative interpolation is second order, and produces results comparable to using the exact solution for ghost cell values.

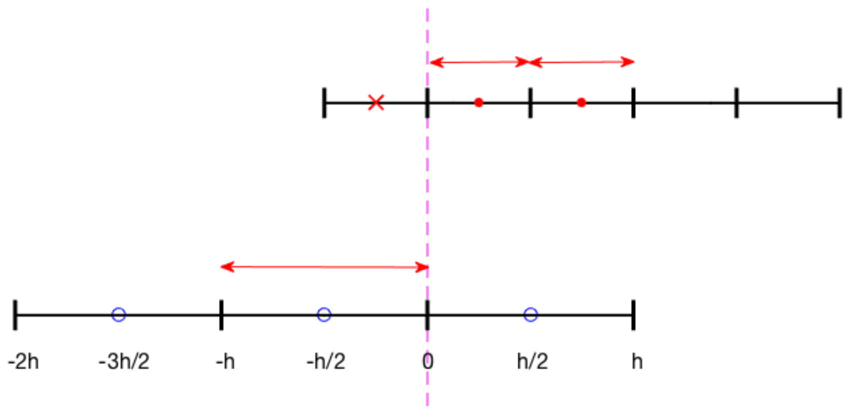


Figure 4.15: Conservative interpolation using an integral to compute the cell averages. The arrows show the cells on which the integrals are imposed.

Interpolation type	Eff. N	Res.	# grids	Error	RHS	Rate
Pointwise coarse interpolation	512	8	20	7.55×10^{-3}	33 856	–
	1024	16	20	2.67×10^{-3}	87 528	1.50
	2048	32	20	1.04×10^{-3}	242 784	1.36
	4096	64	20	4.51×10^{-4}	674 160	1.21
Pointwise mixed interpolation	512	8	20	4.27×10^{-3}	33 856	–
	1024	16	20	1.61×10^{-3}	87 528	1.40
	2048	32	20	6.83×10^{-4}	242 784	1.24
	4096	64	20	3.13×10^{-4}	674 160	1.12
Conservative interpolation	512	8	20	1.91×10^{-3}	33 856	–
	1024	16	20	4.41×10^{-4}	87 528	2.12
	2048	32	20	1.10×10^{-4}	242 784	2.00
	4096	64	20	2.73×10^{-5}	674 160	2.01
Exact, no interpolation	512	8	20	3.47×10^{-4}	33 856	–
	1024	16	20	6.11×10^{-5}	87 528	2.50
	2048	32	20	1.48×10^{-5}	242 784	2.04
	4096	64	20	3.67×10^{-6}	674 160	2.02

Table 4.2: Errors produced using three different interpolation schemes to fill in fine grid ghost cell values. The "mixed" scheme refers to the scheme that uses pointwise values from both the coarse and fine grid. The last row shows the errors computed using the exact solution to fill in the fine grids ghost cell values.

Chapter 5

CONCLUSION

The family of explicit stabilized RKC time stepping schemes were first introduced by Van Der Houwen and Sommeijer in 1980 [20, 21, 28, 29, 31]. These schemes have favorable stability properties making them ideal to efficiently solve stiff parabolic partial differential equations. These schemes provide a way to efficiently time step the parabolic heat or diffusion equation in an explicit manner. The RKC schemes are effective for solving coupled systems by avoiding the need to solve a linear system of equations.

When using the RKC time stepping schemes, we are guaranteed to solve in a pre-determined number of stages, rather than relying on iterations typically required for the type of schemes that require solving a linear system. Explicit schemes have particular advantages when solving on adaptive meshes because we can implement them in multi-rate time stepping. Thus, the RKC schemes are ideal for solving the parabolic heat equation in an AMR setting.

In this thesis, we have described the RKC time stepping strategy in detail and have shown its ideal stability properties. The main contribution is the development of a multi-rate time stepping strategy for adaptively refined meshes. We have demonstrated this in a one dimensional tree code in Matlab. Our numerical evidence suggests that this approach is second order accurate.

The RKC schemes have been used in a variety of settings. In MRI signaling of water diffusion in biological tissues, RKC can be effectively used while coupling a standard Cartesian spatial discretization with an adaptive time discretization [6]. In air quality modeling, RKC can also be used to perform air quality simulations which are computationally intense and require efficient parallelization to make them practical [24]. Other applications include combustion modeling [8], projection methods for the solution of incompressible Navier–Stokes systems [32], and simulation of reacting flow with detailed kinetics and transport [18].

In the future, we would like to apply the RKC schemes to biological pattern formations or crystal growth, and show their effectiveness for solving general reaction-diffusion equations on adaptive meshes. One of the future goals of this thesis is to extend this multi-rate time stepping scheme into higher dimensions in ForestClaw, which is a library for solving PDEs on adaptively refined, multi-block quadtree meshes [7]. Another goal is to apply our multi-rate algorithm to the class of fourth order ROCK methods, designed by Abdulle [2, 3, 4].

Bibliography

[1] A class of high-order Runge-Kutta-Chebyshev stability polynomials. *Journal of Computational Physics*, 300:665–678, 2015.

[2] A. Abdulle. Fourth order Chebyshev methods with recurrence relation. *SIAM J. Sci. Comput.*, 23(6):2041–2054, 2002.

[3] A. Abdulle. Explicit stabilized Runge-Kutta methods. *Springer, Encyclopedia of Applied and Computational Mathematics*, 2011.

[4] A. Abdulle and A. A. Medovikov. Second order Chebyshev methods based on orthogonal polynomials. *Numer. Math.*, 90(1):1–18, 2001.

[5] M. Bakker. Analytical aspects of a minimax problem (in Dutch). *Mathematical Centre, Technical Note TN*, 62, 1971.

[6] D. Calhoun, D. Le Bihan, J. R. Ling, , and C. Poupon. Numerical simulation of diffusion MRI signals using an adaptive time-stepping method. *Physics in Medicine and Biology*, 59(2):441–454, January 2014.

[7] D. Calhoun and C. Burstedde. ForestClaw : A parallel algorithm for patch-based adaptive mesh refinement on a forest of quadtrees. *submitted*, 2017.

[8] T. Echekki and E. Mastorakos, editors. *Turbulent Combustion Modeling: Advances, New Trends and Perspectives*, volume 95 of *Fluid Mechanics and its Applications*. Springer, Dordrecht, 2011.

[9] L. Ferracina and M. N. Spijker. An extension and analysis of the Shu-Osher representation of Runge-Kutta methods. *Mathematics of Computation*, 74(249):201–219, 2005.

[10] E. Hairer, S. P. Norsett, and G. Wanner. *Solving ordinary differential equations I: Nonstiff problems*. Springer Series in Computational Mathematics, 1991.

[11] I. Higueras. Representations of Runge-Kutta methods and strong stability preserving methods. *SIAM Journal on Numerical Analysis*, 43(3):924–948, 2005.

[12] P. J. Van Der Houwen. The development of Runge-Kutta methods for partial differential equations. *Applied Numerical Mathematics*, 20:261–272, 1996.

[13] P. J. Van Der Houwen and B. P. Sommeijer. On the internal stability of explicit, m-stage Runge-Kutta methods for large m-values. *ZAMM Journal of Applied Mathematics and Mechanics*, 60:479–485, 1980.

[14] W. Hundsdorfer, Anna Mozartova, and Valeriu Savcenco. *Monotonicity conditions for multirate and partitioned explicit Runge-Kutta schemes*, pages 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[15] W. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*. Springer Series in Computational Mathematics, 2003.

[16] R. J. LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Society for Industrial and Applied Mathematics, 2007.

[17] T. Myint-U and L. Debnath. *Partial differential equations for scientists and engineers*. North Holland, 1987.

[18] H. H. Najm and O. M. Knio. Modeling low mach number reacting flow with detailed chemistry and transport. *J. Sci. Comp.*, 25(1/2):263–287, 2005.

[19] V. Savcenco, W. Hundsdorfer, and J. G. Verwer. A multirate time stepping strategy for stiff ordinary differential equations. *BIT Numerical Mathematics*, 47:137–155, 2007.

[20] L. F. Shampine, B. P. Sommeijer, and J. G. Verwer. IRKC: An IMEX solver for stiff diffusion-reaction PDEs. *J. of Computational and Applied Mathematics*, 196:485–497, 2006.

[21] B. P. Sommeijer, L. F. Shampine, and J. G. Verwer. RKC: An explicit solver for parabolic PDEs. *Journal of Computational and Applied Mathematics*, 88(2):315–326, 1997.

[22] B. P. Sommeijer and J. G. Verwer. *A performance evaluation of a class of Runge-Kutta-Chebyshev methods for solving semi-discrete parabolic differential equations*. Afdeling Numerieke Wiskunde: NW. Mathematisch Centrum, 1980.

[23] B. P. Sommeijer and J. G. Verwer. A two-step RKC method for time-dependent PDEs. In T. E. Simos and C. Tsitouras, editors, *American Institute of Physics Conference Series*, volume 1048 of *American Institute of Physics Conference Series*, pages 896–899, September 2008.

[24] A. Srivastava, J. C. Linford, and A. Sandu. Performance of Stabilized Explicit Time Integration Methods for Parallel Air Quality Models. In *Proceedings of the*

2007 Spring Simulation Multiconference - Volume 2, SpringSim '07, pages 350–356, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[25] L. N. Trefethen. *Approximation theory and approximation practice.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013.

[26] P. J. Van Der Houwen. *Construction of integration formulas for initial value problems.* North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977. North-Holland Series in Applied Mathematics and Mechanics, Vol. 19.

[27] J. G. Verwer. Explicit Runge-Kutta methods for parabolic partial differential equations. *Applied Numerical Mathematics*, 22:359–379, 1996.

[28] J. G. Verwer, W. H. Hunsdorfer, and B. P. Sommeijer. Convergence properties of the Runge-Kutta-Chebyshev method. *Numer. Math.*, 57:157–178, 1990.

[29] J. G. Verwer and B. P. Sommeijer. An implicit-explicit Runge-Kutta-Chebyshev scheme for diffusion-reaction equations. *SIAM Journal on Scientific Computing*, 25(5):1824–1835, 2004.

[30] J.G. Verwer, B.P. Sommeijer, and W. Hundsdorfer. {RKC} time-stepping for advection–diffusion–reaction problems. *Journal of Computational Physics*, 201(1):61 – 79, 2004.

[31] C. J. Zbinden. Partitioned Runge-Kutta-Chebyshev methods for diffusion-advection-reaction problems. *SIAM Journal on Scientific Computing*, 33(4):1707–1725, 2011.

[32] Z. Zheng and L. Petzold. Runge–Kutta–Chebyshev projection method. *J. Comput. Phys.*, 219(2):976–991, 12 2006.

Appendix A

BUTCHER TABLEAU OF RKC

We can organize the RKC scheme in a standard Runge-Kutta form. Here, we present the RKC scheme on a Butcher tableau.

$$
\begin{array}{l|ccccc}
c_0 = 0 & 0 & 0 & 0 & 0 & 0 \\
c_1 = \tilde{\mu}_1 & \tilde{\mu}_1 & 0 & 0 & 0 & 0 \\
\mathbf{c_2} & \mathbf{a_{21}} & \tilde{\mu}_2 & 0 & 0 & 0 \\
\mathbf{c_3} & \mathbf{a_{31}} & \mu_3\tilde{\mu}_2 & \tilde{\mu}_3 & 0 & 0 \\
\mathbf{c_4} & \mathbf{a_{41}} & \mathbf{a_{42}} & \mu_4\tilde{\mu}_3 & \tilde{\mu}_4 & 0 \\
\mathbf{c_5} & \mathbf{a_{51}} & \mathbf{a_{52}} & \mathbf{a_{53}} & \mu_5\tilde{\mu}_4 & \tilde{\mu}_5 \\
\hline
 & b_1 & b_2 & b_3 & b_4 & b_5
\end{array}
$$

Table A.1: The Butcher tableau for the RKC Scheme

$$c_2 = \mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}$$

$$a_{21} = \mu_2\tilde{\mu_1} + \tilde{\gamma_2}$$

$$c_3 = \mu_3(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) + \nu_3\tilde{\mu_1} + \tilde{\mu_3} + \tilde{\gamma_3}$$

$$a_{31} = \mu_3(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}) + \nu_3\tilde{\mu_1} + \tilde{\gamma_3}$$

$$\begin{aligned} c_4 &= \mu_4\left(\mu_3 c_2 + \nu_3 c_1 + \tilde{\mu_3} + \tilde{\gamma_3}\right) + \nu_4\left(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}\right) + \tilde{\mu_4} + \tilde{\gamma_4} \\ &= \mu_4\left(\mu_3\mu_2\tilde{\mu_1} + \mu_3\tilde{\mu_2} + \mu_3\tilde{\gamma_2} + \nu_3\tilde{\mu_1} + \tilde{\mu_3} + \tilde{\gamma_3}\right) + \nu_4\left(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}\right) + \tilde{\mu_4} + \tilde{\gamma_4} \end{aligned}$$

$$\begin{aligned} a_{41} &= \mu_4[\mu_3(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}) + \nu_3\tilde{\mu_1} + \tilde{\gamma_3}] + \nu_4(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}) + \tilde{\gamma_4} \\ &= \mu_4\mu_3\mu_2\tilde{\mu_1} + \mu_3\mu_4\tilde{\gamma_2} + \mu_4\tilde{\gamma_3} + \mu_4\nu_3\tilde{\mu_1} + \nu_4\mu_2\tilde{\mu_1} + \tilde{\gamma_4} + \nu_4\tilde{\gamma_2} \end{aligned}$$

$$a_{42} = \mu_4\mu_3\tilde{\mu_2} + \nu_4\tilde{\mu_2}$$

$$\begin{aligned} c_5 = {}& \left[\mu_4\left(\mu_3 c_2 + \nu_3 c_1 + \tilde{\mu_3} + \tilde{\gamma_3}\right) + \nu_4(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) + \tilde{\mu_4} + \tilde{\gamma_4}\right] \\ &+ \nu_5\left[\mu_3\left(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}\right) + \nu_3\tilde{\mu_1} + \tilde{\mu_3} + \tilde{\gamma_3}\right] + \tilde{\mu_5} + \tilde{\gamma_5} \end{aligned}$$

$$\begin{aligned} a_{51} = {}& \mu_5\left[\mu_4\left[\mu_3(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}) + \nu_3\tilde{\mu_1} + \tilde{\gamma_3}\right] + \nu_4\left(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}\right) + \tilde{\gamma_4}\right] \\ &+ \nu_5\left[\mu_3\left(\mu_2\tilde{\mu_1} + \tilde{\gamma_2}\right) + \nu_3\tilde{\mu_1} + \tilde{\gamma_3}\right] + \tilde{\gamma_5} \end{aligned}$$

$$a_{52} = \mu_5\left(\mu_4\tilde{\mu_2} + \nu_4\tilde{\mu_2}\right)$$

$$a_{53} = \mu_5\mu_4\tilde{\mu_3} + \tilde{\mu_3}$$

Notice that the following holds for all of the c values.

$$c_1 = a_{11}$$

$$c_2 = a_{21} + a_{22}$$

$$c_3 = a_{31} + a_{32} + a_{33}$$

$$c_4 = a_{41} + a_{42} + a_{43} + a_{44}$$

$$c_5 = a_{51} + a_{52} + a_{53} + a_{54} + a_{55}$$

Therefore we have,

$$c_j = \sum_{l=0}^{j-1} a_{jl}$$

Due to the recursive form of the method (recursion on Y_j values), it is more convenient to show the stages using the stages like in the RKC formulas in Chapter 3 rather than with the Butcher Tableau.

Appendix B

THE ORDER OF ACCURACY OF RKC

Consider solving the test problem $U' = \lambda U$ using the RKC scheme. According to the RKC formulas in Section 3.3 (equation 3.23) we have s stages. Assume that we have 3 stages, Y_1 , Y_2, and Y_3. Referring to Section 3.3, we can write the formulas for the Y's accordingly.

$$
\begin{aligned}
Y_0 =& u^n \\
Y_1 =& Y_0 + \tilde{\mu_1} k F_0 = u^n + \tilde{\mu_1} k \lambda u^n = \left[1 + \tilde{\mu_1} k \lambda\right] u^n \\
Y_2 =& \ \mu_2 Y_1 + \nu_2 Y_0 + (1 - \mu_2 - \nu_2) Y_0 + \tilde{\mu_2} k F_1 + \tilde{\gamma_2} k F_0 \\
=& \mu_2 (1 + \tilde{\mu_1} k \lambda) u^n + \nu_2 u^n + (1 - \mu_2 - \nu_2) u^n + \tilde{\mu_2} k \lambda (1 + \tilde{\mu_1} k \lambda) u^n + \tilde{\gamma_2} k \lambda u^n \\
=& \left[\mu_2 \tilde{\mu_1} k \lambda + 1 + \tilde{\mu_2} k \lambda + \tilde{\mu_2} \tilde{\mu_1} k^2 \lambda^2 + \tilde{\gamma_2} k \lambda\right] u^n \\
=& \left[1 + (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k \lambda + \tilde{\mu_2} k^2 \lambda^2\right] u^n \\
Y_3 =& \mu_3 Y_2 + \nu_3 Y_1 + (1 - \mu_3 - \nu_3) Y_0 + \tilde{\mu_3} k F_2 + \tilde{\gamma_3} k F_0 \\
=& \mu_3 \left[1 + (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k \lambda + \tilde{\mu_2} \tilde{\mu_1} k^2 \lambda^2\right] u^n + \nu_3 (1 + \tilde{\mu_1} k \lambda) u^n + (1 - \mu_3 - \nu_3) u^n \\
& + \tilde{\mu_3} k \lambda \left[1 + (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k \lambda + \tilde{\mu_2} \tilde{\mu_1} k^2 \lambda^2\right] u^n + \tilde{\gamma_3} k \lambda u^n
\end{aligned}
$$

$$
\begin{aligned}
=& \Big[\mu_3 (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k \lambda + \mu_3 \tilde{\mu_2} \tilde{\mu_1} k^2 \lambda^2 + \nu_3 \tilde{\mu_1} k \lambda + 1 + \tilde{\mu_3} k \lambda + \tilde{\mu_3} (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k^2 \lambda^2 \\
& + \tilde{\mu_3} \tilde{\mu_2} \tilde{\mu_1} k^3 \lambda^3 + \tilde{\gamma_3} k \lambda\Big] u^n \\
=& \Big[1 + \mu_3 (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k \lambda + \nu_3 \tilde{\mu_1} k \lambda + \tilde{\mu_3} k \lambda + \tilde{\gamma_3} k \lambda + \mu_3 \tilde{\mu_2} \tilde{\mu_1} k^2 \lambda^2 + \tilde{\mu_3} (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) k^2 \lambda^2 \\
& + \tilde{\mu_3} \tilde{\mu_2} \tilde{\mu_1} k^3 \lambda^3\Big] u^n \\
=& \Big[1 + [\mu_3 (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) + \nu_3 \tilde{\mu_1} + \tilde{\mu_3} + \tilde{\gamma_3}] k \lambda + [\mu_3 \tilde{\mu_2} \tilde{\mu_1} + \tilde{\mu_3} (\mu_2 \tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2})] k^2 \lambda^2 \\
& + \tilde{\mu_3} \tilde{\mu_2} \tilde{\mu_1} k^3 \lambda^3\Big] u^n
\end{aligned}
$$

$$u^{n+1} = Y_3$$

Now, in order to determine the accuracy, we check the coefficients of $k\lambda$ and $k^2\lambda^2$. Assume $z = k\lambda$, so we are checking the coefficients of z, and z^2, as we saw in Section 3.1. Using formulas of the first order RKC, we verify that the coefficient of z is equal to 1 to prove the first order accuracy.

$$\begin{aligned}
\mu_3(\mu_2\tilde{\mu_1} + \tilde{\mu_2} + \tilde{\gamma_2}) + \nu_3\tilde{\mu_1} + \tilde{\mu_3} + \tilde{\gamma_3} &= \frac{2w_0b_3}{b_2}\left[\frac{2w_0b_2}{b_1}\frac{w_1}{w_0} + \frac{2w_1b_2}{b_1}\right] + \frac{-b_3}{b_1}\frac{w_1}{w_0} + \frac{2w_1b_3}{b_2} \\
&= \frac{4w_0^2b_2b_3w_1}{b_1b_2w_0} + \frac{4w_1w_0b_3b_2}{b_1b_2} + \frac{-b_3w_1}{b_1w_0} + \frac{2w_1b_3}{b_2} \\
&= \frac{4w_0^2b_2b_3w_1 + 4w_1w_0^2b_2b_3 - b_3w_1b_2 + 2w_1b_3b_1w_0}{b_1b_2w_0} \\
&= \frac{1}{w_0b_1b_2}\left[8w_0^2w_1b_2b_3 - w_1b_2b_3 + 2w_0w_1b_1b_3\right] \\
&= \frac{T_1T_2}{w_0}\left[8w_0^2\frac{T_3}{T_3'}\frac{1}{T_2T_3} - \frac{T_3}{T_3'}\frac{1}{T_2T_3} + 2w_0\frac{T_3}{T_3'}\frac{1}{T_3}\frac{1}{T_4}\right] \\
&= T_2\left[\frac{8w_0^2}{T_3'T_2} - \frac{1}{T_3'T_2} + \frac{2}{T_3'}\right] \\
&= \frac{8w_0^2}{T_3'} - \frac{1}{T_3'} + \frac{2T_2}{T_3'} \quad \left(T_2 = 2w_0^2 - 1,\ T_3' = 12w_0^2 - 3\right) \\
&= \frac{8w_0^2 + 2(2w_0^2 - 1) - 1}{12w_0^2 - 3} \\
&= \frac{12w_0^2 - 3}{12w_0^2 - 3} \\
&= 1
\end{aligned}$$

This shows first order accuracy. Similarly, for the second order case, the coefficient of $z = k\lambda$ must be equal to 1; furthermore, the coefficient of $z^2 = k^2\lambda^2$ must be equal to $\frac{1}{2}$ to fulfill the second order accuracy requirement. For the first equation, we have,

$$\mu_3\left(\mu_2\tilde{\mu_1}+\tilde{\mu_2}+\tilde{\gamma_2}\right)+\nu_3\tilde{\mu_1}+\tilde{\mu_3}+\tilde{\gamma_3}=$$

$$\begin{aligned}
&=2w_0\frac{b_3}{b_2}\left[2w_0\frac{b_2}{b_1}b_1w_1+2w_1\frac{b_2}{b_1}-(1-b_1T_1(w_0)).2w_1\frac{b_2}{b_1}\right]\\
&\quad+\frac{-b_3}{b_1}b_1w_1+2w_1\frac{b_3}{b_2}-\big(1-b_2T_2(w_0)\big)2w_1\frac{b_3}{b_2}\\
&=2w_0\frac{b_3}{b_2}\left[\frac{2w_0b_2b_1w_1}{b_1}+\frac{2w_1b_2}{b_1}-\frac{2w_1b_2(1-b_1T_1(w_0))}{b_1}\right]\\
&\quad+\frac{-b_3b_1w_1}{b_1}+\frac{2w_1b_3}{b_2}-\frac{2w_1b_3(1-b_2T_2(w_0))}{b_2}\\
&=\frac{2w_0b_3}{b_2}\left[\frac{2w_0b_2b_1w_1}{b_1}+\frac{2w_1b_2b_1T_1(w_0)}{b_1}\right]+\frac{-b_3b_1w_1}{b_1}+\frac{2w_1b_3b_2T_2(w_0)}{b_2}\\
&=\frac{4w_0^2w_1b_1b_2b_3}{b_1b_2}+\frac{4w_0^2w_1b_1b_2b_3}{b_1b_2}-\frac{w_1b_1b_2b_3}{b_1b_2}+\frac{2w_1b_2b_3b_1T_2(w_0)}{b_1b_2}\\
&\quad\big(\text{where } T_2(w_0)=2w_0^2-1\big)\\
&=\frac{8w_0^2w_1b_1b_2b_3}{b_1b_2}-\frac{w_1b_1b_2b_3}{b_1b_2}+\frac{4w_0^2w_1b_1b_2b_3}{b_1b_2}-\frac{2w_1b_1b_2b_3}{b_1b_2}\\
&=\frac{12w_0^2w_1b_1b_2b_3}{b_1b_2}-\frac{3w_1b_1b_2b_3}{b_1b_2}\\
&=12w_0^2w_1b_3-3w_1b_3\\
&=\left(12w_0^2-3\right)b_3w_1\\
&=3\left(4w_0^2-1\right).\frac{T_3'(w_0)}{T_3''(w_0)}.\frac{T_3''(w_0)}{(T_3'(w_0))^2}\qquad\left(\text{where } T_3'(w_0)=12w_0^2-3\right)\\
&=\frac{3\left(4w_0^2-1\right)}{12w_0^2-3}\\
&=1
\end{aligned}$$

Now we check the second equation,

$$
\begin{aligned}
\mu_3\tilde{\mu_2}\tilde{\mu_1}+\tilde{\mu_3}\left(\mu_2\tilde{\mu_1}+\tilde{\mu_2}+\tilde{\gamma_2}\right) = & \\
&=2w_0\frac{b_3}{b_2}2w_1\frac{b_2}{b_1}b_1w_1+2w_1\frac{b_3}{b_2}\left(2w_0\frac{b_2}{b_1}b_1w_1+2w_1\frac{b_2}{b_1}-\left(1-b_1T_1\left(w_0\right)\right)2w_1\frac{b_2}{b_1}\right)\\
&=\frac{4w_0w_1^2b_2b_3b_1}{b_1b_2}+\frac{2w_1b_3}{b_2}\left[\frac{2w_0w_1b_1b_2}{b_1}+\frac{2w_1b_2}{b_1}-\frac{2w_1b_2}{b_1}+\frac{2w_1b_2b_1T_1(w_0)}{b_1}\right]\\
&=\frac{4w_0w_1^2b_1b_2b_3}{b_1b_2}+\frac{4w_0w_1^2b_1b_2b_3}{b_1b_2}+\frac{4w_1^2b_1b_2b_3w_0}{b_1b_2}\\
&=\frac{12w_0w_1^2b_1b_2b_3}{b_1b_2}\\
&=12w_0w_1^2b_3\\
&=12w_0\frac{\left(T_3'\left(w_0\right)\right)^2}{\left(T_3''\left(w_0\right)\right)^2}\frac{T_3''\left(w_0\right)}{\left(T_3'\left(w_0\right)\right)^2}\\
&=\frac{12w_0}{T_3''\left(w_0\right)}\qquad\left(\text{where } T_3''\left(w_0\right)=24w_0\right)\\
&=\frac{1}{2}
\end{aligned}
$$

This shows second order accuracy.

Appendix C

EIGENVALUES OF THE MOL DISCRETIZATION MATRIX OF THE HEAT EQUATION

In this appendix, we find the eigenvalues of matrix A discussed in Chapter 2,

$$U'(t) = AU(t) + b(t) \implies A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & 0 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 0 & & & & 1 & -2 \end{bmatrix}$$

where h is the spatial scale. We note that we want to find λ such that

$$Av_p = \lambda_p v_p$$

where v_p is the corresponding eigenvector. Since we are looking at the 1D case, these will be scalar values. So, we look to solve

$$\frac{v_{p+1} - 2v_p + v_{p-1}}{h^2} = \lambda_p v_p \qquad p = 1, 2, \cdots, N$$

Now, there are only n eigenvalues and n eigenvectors, so we let $v_0 = v_{n+1} = 0$. In order to utilize Chebyshev polynomials, we let $2\alpha = (2 + h^2\lambda)$ and we scale v so that $v_1 = 1$.

Then the problem becomes the recurrence

$$\begin{aligned} &v_0 = 0 \\ &v_1 = 1 \\ &v_{p+1} = 2\alpha v_p - v_{p-1} \end{aligned}$$

which gives $v_{p+1} = U_p(\alpha)$, where U_k is the p^{th} Chebyshev polynomial of the second kind. Since $v_{N+1} = 0$, we see that $U_n(\alpha) = 0$. This implies that we can find the eigenvalues λ_k by first finding the roots of U_n. These roots are known to be

$$\alpha_k = \cos\left(\frac{p\pi}{N+1}\right).$$

Recall that we let $2\alpha = (2 + h^2\lambda_p)$. So

$$2\cos\left(\frac{p\pi}{N+1}\right) = 2 + h^2\lambda_p$$

implying that

$$\lambda_k = \frac{2}{h^2}\left(\cos\left(\frac{p\pi}{N+1}\right) - 1\right) \quad \text{for} \quad p = 1, 2, \cdots, N.$$