A Comparison of Discretization Methods on Coarse Non-Uniform Grids

Bachelor's Project Applied Mathematics

June 2025

Student: Elgars Dobelis

Supervisor: Prof. Dr. A.E.P Veldman

First examiner: Prof. Dr. ir R.W.C.P Verstappen

Second examiner: Dr. A.E. Sterk

Abstract

This paper studies the behavior of discretization methods on coarse, non-uniform grids using the convection–diffusion equation as a test case. We implement grids with both abrupt and smooth transitions in grid cell size, and compare the performance of finite difference and finite volume methods. Our findings show that discretizations whose coefficient matrices preserve skew-symmetry when modeling convection yield more accurate results. Interestingly, this holds even if such schemes may exhibit larger local truncation error. In particular, we observe that only skew-symmetric discretization methods conserve energy when evolution of time is introduced, which aligns with underlying physical properties of convection. These results highlight the importance of selecting appropriate discretization methods when working with coarse, non-uniform grids and when aiming to preserve real-life properties.

Acknowledgments

I would like to acknowledge and give my warmest thanks to my supervisor Professor Veldman who made this work possible. His guidance and provided references carried me through all the stages of writing this report. Thank you for showing me that the field of computational mathematics is what I want to base my future on.

I am also grateful to my examiners, Professor Verstappen and Professor Sterk. Thank you for the guidance regarding the organizational process and for the questions posed during the final presentation, which encouraged me to think about my research more broadly.

Contents

1 Introduction

The study of discretization methods is fundamental to the field of computational mathematics. Numerical methods can help us solve complicated differential equations, whose analytical solutions are rarely available. Moreover, they provide a cost-effective way of simulating real-life scenarios. Specifically, the field of computational fluid dynamics makes way for research concerned with real-world processes, such as the flight of an airplane, dyke construction near shores or drug transport in the bloodstream. Since the research is critical for real-life applications, improving the accuracy of discretization methods should always be an important topic. That said, computers used for such simulations also have in-built limits. Therefore, one is forced to find discretization methods that both preserve accuracy while maintaining low computational costs [1, Section 1].

By placing more grid points in regions where the function displays significant change yet relaxing the discretization where the function stays constant, one can get a good approximation while significantly reducing computational cost. Such approach is not possible with a uniform grid, which also discretizes regions with little change. However, the use of a non-uniform grid complicates the analysis of discretization methods. As this report will show, methods that are identical on uniform grids can behave very differently on non-uniform grids, therefore comparing them is necessary to identify the better method.

In this paper, we investigate discretization methods on non-uniform grids constructed with a small number of grid cells. We will analyze methods from the finite difference and finite volume frameworks. For comparing the performance of each method we will use the convection-diffusion equation, which, as shown later, is a perfect test case for non-uniform grid analysis. By analyzing both the numerical results and the properties of the coefficient matrices, in each discretization framework we will identify a method which is worth recommending.

Our findings, however, are quite counterintuitive. Local truncation error, which usually is a good estimate of the convergence properties of discretization methods, turns out to be a misleading metric. Discretization methods with a larger local truncation error will produce lower global error. As this report shows, a key quality of discretization methods is to conserve physical properties of real-life scenarios they model, which provides the underlying reason why some methods behave better than others.

The structure of this report is as follows: Section 2 introduces the convection-diffusion equation, noting its form and relevance in the field. Section 3 includes detailed construction of non-uniform grids used throughout this report. Moreover, it motivates the need for such grids when discretizing the convection-diffusion equation and modeling real-life events. Sections 4 and 5 present an analysis of finite difference and finite volume frameworks. For each framework, we provide the construction of the methods, illustrate their numerical results and analyze coefficient matrix properties. In Section 6, we reintroduce physics into the analysis by noting the conservation of physical properties. Further, in Section 7 we present some noteworthy results concerned with randomly distributed grids. Lastly, we present our conclusions and key findings, with respective limitations and suggestions for further research in Section 8.

2 The Convection-Diffusion Equation

For discretization method comparison, we will be using a specific partial differential equation. In this paper, we will be discretizing the convection-diffusion equation:

$$\frac{\partial \phi}{\partial t} + \underbrace{u\frac{\partial \phi}{\partial x}}_{convection} - \underbrace{k\frac{\partial^2 \phi}{\partial x^2}}_{diffusion} = 0. \tag{1}$$

The convection-diffusion equation describes how a scalar quantity ϕ is moved by a fluid through two physical processes:

- Convection - representing the transport of ϕ due to motion of the surrounding fluid, modeled by the first order derivative in space.
- Diffusion - describing the quantity spreading out on its own, modeled by the second order spatial derivative.

To capture evolution of ϕ over time we include the time derivative [2, Section 1.1].

Coefficient u is the convective coefficient, which denotes the speed of convection. The diffusive coefficient k measures the strength of diffusion. For simplicity in this report we keep $u = 1$, meaning, we model convection at unit speed. The diffusive coefficient k, however, will be varied throughout experiments, with smaller values of k introducing more difficulties to our discretization methods.

For most experiments, we keep $k = 10^{-2}$. With such a diffusive coefficient and convective coefficient equaling 1, it is possible to model many real-life properties. Therefore, understanding properties about the convection-diffusion equation can give insights into simulating, for example, drug transport in the bloodstream, improving treatment efficiency. This is highlighted by [3], where the authors note that for modeling blood flow a high Péclet number (Pe) is used, which denotes the ratio between rates of convection and diffusion.

The discretization methods implemented in this report will only discretize in space. Therefore, without the time derivative and with $u = 1$, we will be discretizing the following ODE, with corresponding domain and boundary conditions:

$$\begin{gathered} \phi_x - k\phi_{xx} = 0, \\ x \in [0,1], \\ \phi(0) = 0, \quad \phi(1) = 1. \end{gathered} \tag{2}$$

As illustrated by plots in the following section, such boundary conditions can be motivated through the function experiencing smooth, constant regions and sharp jumps to match the given boundary conditions. Moreover, any problem on a domain $[a, b]$ may be rescaled on the interval $[0, 1]$, with the transformation $\xi = (x-a)/(b-a)$. Therefore, we do not lose generality with such domain conditions.

For the boundary value problem (2) the analytical solution is given by:

$$\phi(x) = \frac{e^{x/k} - 1}{e^{1/k} - 1}, \tag{3}$$

which allows us to compute the global error of each discretization method introduced in this report. In further sections, we denote the analytical solution as ϕ_{exact} and the discretized solution as ϕ.

The motivation for analyzing specifically the convection-diffusion equation extends further than having the equation model real-life properties. It is also connected to the famous Navier-Stokes equations, that model all flow problems around us. Both have convective and diffusive terms. Therefore, by understanding discretization properties of the convection-diffusion equation, it can help figure out discretization strategies for the Navier-Stokes equations [2, Section 2.3].

3 Non-Uniform Grids

3.1 Need for Coarse Non-Uniform Grids

We start the section concerned with non-uniform grids by motivating the need for such grids. In the following figures, one can note how the convection-diffusion (2) equation behaves for various positive, small k values.

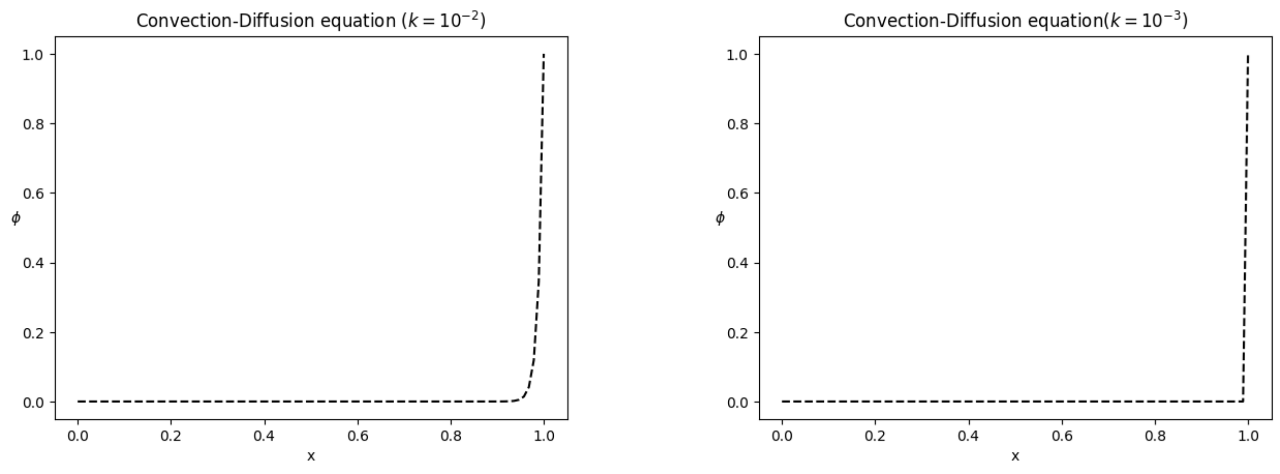


Figure 1: The convection-diffusion equation for different k values.

For most of the domain the function remains constant at approximately 0. However, at the end, the function increases rapidly to meet the boundary condition $\phi(1) = 1$. We can note that for decreasing k values this increase becomes steeper. For $k = 10^{-3}$ in the right plot, the function even resembles a step function.

For discretization methods to be successful when working with these types of functions, it is necessary to use a sufficient number of grid cells in the interval where the function displays a steep gradient. It is possible to discretize such functions with uniform grids, however, this approach wastes a lot of computational power. Although uniform grids can sufficiently cover the interval where the function increases sharply, they also introduce many unnecessary grid points in the region where the function does not display significant change.

This is where non-uniform grids come into play. With non-uniform grids, we can ensure the interval with the sharp increase to be discretized with many small grid cells while discretizing the rest of the domain with fewer large grid cells.

We now introduce some key terminology. The interval in which the function experiences this sharp increase is called the boundary layer. Grids that discretize with a few large grid cells are called coarse grids. Grids with many small grid cells are referred to as fine grids. With such terminology, we can state how our non-uniform grids for discretizing the convection-diffusion equation will behave: the goal of non-uniform grids is to discretize the boundary layer finely, while discretizing the rest of the domain coarsely.

Moreover, in this paper we will limit-test our methods by decreasing the number of grid cells our grids have. For a discretization method to be considered good, it needs to both yield a good approximation and be able to do so with a limited number of underlying grid cells.

Understanding the notion of non-uniform grids is essential when one is concerned with computational fluid dynamics [2, Section 1.5]. Having functions modeling real-life events, that display both constant behavior and intervals of rapid change, calls for using a non-uniform grid structure. Moreover, by making the non-uniform grids coarser, we enable simulations in fluid dynamics to run faster and more efficiently.

3.2 Non-Uniform Grid Construction

3.2.1 Boundary layer definition

The goal of our non-uniform grids is to finely discretize the boundary layer or the interval in which the function displays a sharp increase. Hence, we must identify where the boundary layer begins. To characterize where the boundary layer starts, one can introduce conditions on the gradient of the function. However, in this paper, we take a simplified approach and identify the location of this rapid increase visually from the plots in Figure 1.

The diffusive coefficient k controls how fast the solution rises near the boundary. We define the threshold coefficient T, such that the expression Tk represents the width of the boundary layer for a fixed k. In this report, we take $T = 5$, meaning the boundary layer is considered to begin at $5k$ distance from the right boundary. This threshold describes it best where the boundary layer starts for the value of $k = 10^{-2}$ [4, Section 3.2]. Such a value for the diffusive coefficient will be used for most experiments in this report. Hence, we use a fine grid on the interval $[1 - 5k, 1]$ and a coarse one on $[0, 1 - 5k]$.

With smaller k values, the value of T should be redefined as the starting location of the boundary layer does not scale linearly with k. A thorough investigation of how the boundary layer location scales with k lies beyond the scope of this report. There is no concrete definition (for example, including the gradient computation of the function) to define where the rapid increase of the function starts. Moreover, the main focus of this report is the comparison of discretization methods, not on non-uniform grid construction.

In the definitions of non-uniform grids and the code (Appendix C) constructing them, we have left in the threshold coefficient T. This allows us to reuse the code if further research considering the boundary layer is to be done. Moreover, this also makes the grid definitions more readable and reusable than if we had used the arbitrary integer 5.

3.2.2 General Grid Structure and Notation

We impose a rule that all our non-uniform grids must satisfy: half of the grid cells lie in the boundary layer and half lie outside. This ensures that the boundary layer is discretized with more care. We use a fine grid where the function exhibits rapid changes and use fewer grid cells (saving computational time) elsewhere.

In this report, we introduce five grid structures. All grids will follow the same notation, where we denote the grid points by x_i. Our grids will have N grid cells. Therefore, the grid point denoting the left boundary $x = 0$ will be x_0 and the grid point on the right boundary $x = 1$ will be x_N. Such grids will have $N + 1$ grid points. In all example plots, red vertical lines mark the grid points. In the following section, we introduce two non-uniform grid structures: the abrupt grid and the exponential grid. These grids will be used to generate most of the results in this paper.

3.2.3 The Abrupt Grid

The abrupt grid will consist of two uniform parts. Half of the grid cells will divide the boundary layer, while the other half are allocated to the rest of the domain. The grid gets its name, as the change between the subsequent grid cells, when the two uniform parts meet, is abrupt. Such a structure is introduced with 10 grid cells in [4, Section 3.2]. In this paper we generalize to an arbitrary number of grid cells.

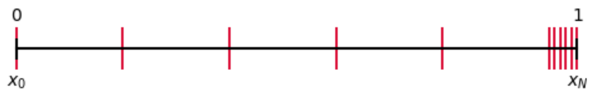
The formal definition for the grid point placement is given here. To calculate the distance between grid points in each uniform part, we need the number of grid cells in each part which will differ for grids with even and odd number of grid cells. We denote $M := \left\lfloor \frac{N}{2} \right\rfloor$. This will be the number of grid cells in the interval outside the boundary layer, meaning we will have

$N-M$ grid cells in the boundary layer. For N even, both intervals will have the same amount of grid cells ($M = N - M$), whereas for N odd, the boundary layer will have an extra cell. With this, the general formula for defining the abrupt grid then becomes:

Definition (Abrupt grid):

$$x_i = i\frac{(1-Tk)}{M}, \text{ for } i = 0, ..., M,$$
$$x_i = (1-Tk) + (i-M)\frac{Tk}{N-M}, \text{ for } i = M+1, ..., N.$$

Example (The abrupt grid with 10 grid cells):



3.2.4 The Exponential Grid

Unlike for the abrupt grid, for the exponential grid, all grid cells will have different width. Furthermore, the change in the grid cell size will be smooth, which makes the grid better for discretizing real-life scenarios. The construction of this grid is based on the approach shown in [1, Section 3].

The grid point placement for the exponential grid will be obtained by the transformation:

$$x = \frac{1-e^{-s\sigma}}{1-e^{-s}}, \quad \text{for } \sigma \in [0,1]. \tag{4}$$

We again make half of the grid cells lie in the boundary layer by choosing the value s such that the value $\sigma = 0.5$ corresponds to the edge of the boundary layer $1 - Tk$:

$$s = 2\ln[(1-Tk)/Tk]. \tag{5}$$

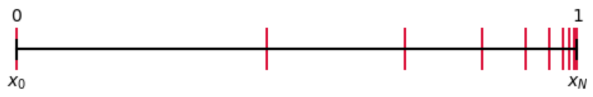
We define the exponential grid by making the transformation (4) generated from σ_i forming a uniform grid. Therefore, the formulation becomes:

Definition (Exponential Grid):

$$x_i = \frac{1-e^{-s\sigma_i}}{1-e^{-s}}, \quad \text{with } \sigma_i = i\frac{1}{N}, \quad \text{for } i = 0, ..., N, \tag{6}$$

where s is as defined in 5.

Example (Exponential grid with 10 *grid cells):*



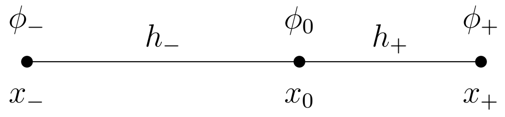
4 Finite Difference Methods

Having established the non-uniform grid structures for discretizing the sharply increasing convection-diffusion equation, we now turn to defining the discretization methods operating on such grids.

We begin our discretization analysis by approximating the derivatives using finite difference methods (FDM). We introduce two methods named Method A and Method B. The second order derivative for both methods will be approximated in exactly the same manner. That said, for each method, we approximate the first order derivative in a different way. This, as we will see in this section, will be enough to impact the results of the discretization drastically. The construction of the discretization methods follows the ones introduced in [2, Section 1.5.1]. As with any finite difference method, we begin this section with the Taylor series.

4.1 Taylor Series and the Approximation of ϕ_x

Consider any triplet of grid points x_-, x_0 and x_+ with corresponding function values ϕ_-, ϕ_0 and ϕ_+ as defined in the following picture:



Here, $h_- = x_0 - x_-$ and $h_+ = x_+ - x_0$ denote the differences of respective adjacent grid points. In a non-uniform grid setting, these values are generally not equal.

The idea of our chosen finite difference approximation, central differences, is to make approximations of the derivative of ϕ_0 on all such triplet of points. We will use combinations of Taylor series at adjacent grid points to approximate both the first and the second order derivative of ϕ_0. For a better approximation, one can involve more neighbors of ϕ_0, however in this paper, we will only be using the adjacent function values.

We start by writing out the Taylor expansion of both ϕ_+ and ϕ_- :

$$\phi_+ = \phi_0 + h_+\phi_x + \frac{1}{2}h_+^2\phi_{xx} + \frac{1}{6}h_+^3\phi_{xxx} + \ldots \tag{7}$$

$$\phi_- = \phi_0 - h_-\phi_x + \frac{1}{2}h_-^2\phi_{xx} - \frac{1}{6}h_-^3\phi_{xxx} + \ldots \tag{8}$$

One can now combine these series in an arbitrary manner to approximate the values of derivatives. By simply subtracting (8) from (7) we get:
Approximation of ϕ_x (Method A):

$$\phi_+ - \phi_- = (h_+ + h_-)\phi_x + \frac{1}{2}(h_+^2 - h_-^2)\phi_{xx} + \frac{1}{6}(h_+^3 + h_-^3)\phi_{xxx} + ... \tag{9}$$

$$\Longleftrightarrow$$

$$-(h_+ + h_-)\phi_x = -(\phi_+ - \phi_-) + \frac{1}{2}(h_+^2 - h_-^2)\phi_{xx} + \frac{1}{6}(h_+^3 + h_-^3)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$\phi_x = \frac{\phi_+ - \phi_-}{h_+ + h_-} - \frac{1}{2}(h_+ - h_-)\phi_{xx} - \frac{1}{6}\frac{h_+^3 + h_-^3}{h_+ + h_-}\phi_{xxx} + ... \tag{10}$$

$$\Longrightarrow$$

$$\phi_x \approx \frac{\phi_+ - \phi_-}{h_+ + h_-} \tag{11}$$

Method B, however, will require a more involved derivation. By combining $h_-^2 \times (7) - h_+^2 \times (8)$, the first-order derivative for Method B is computed in the following way:
Approximation of ϕ_x (Method B):

$$h_-^2\phi_+ - h_+^2\phi_- = (h_-^2 - h_+^2)\phi_0 + h_+h_-(h_+ + h_-)\phi_x + \frac{1}{6}h_+^2h_-^2(h_+ + h_-)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$h_+h_-(h_+ + h_-)\phi_x = h_-^2\phi_+ - h_+^2\phi_- + (h_+^2 - h_-^2)\phi_0 - \frac{1}{6}h_+^2h_-^2(h_+ + h_-)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$\phi_x = \frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)} - \frac{1}{6}h_+h_-\phi_{xxx} + ... \tag{12}$$

$$\Longrightarrow$$

$$\phi_x \approx \frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)} \tag{13}$$

The approximations for the first derivative, ϕ_x, in Method A and Method B are therefore the first terms on the right-hand side of equations (10) and (12), respectively. The rest of the terms are omitted, creating an error in our approximations, called the local truncation error, denoted by τ.

4.2 First-Glance Comparison

From the derived first derivative approximation formulas of Method A (11) and Method B (13) we can already draw several insights. We first note that both methods are, in fact, identical in an equidistant grid setting. With h denoting the grid cell size for a uniform grid, $h_- = h_+ = h$.

Therefore, for Method B we get the following:

$$\begin{aligned}\phi_x &\approx \frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)}\\ &= \frac{h^2\phi_+ + (h^2 - h^2)\phi_0 - h^2\phi_-}{h^2(h+h)}\\ &= \frac{\phi_+ - \phi_-}{2h},\end{aligned}$$

which would equal Method A in a uniform grid setting. This already highlights the introduced complexity if one were to use non-uniform grids. On a uniform grid, since the methods are identical, no comparison would be necessary.

The geometric interpretation of both methods is given in Figure 2, below. We can see that Method B approximates the first order derivative evaluating the slope of the function at the point ϕ_0, whereas Method A approximates the derivatives just by estimating the slope from interpolating between adjacent points ϕ_- and ϕ_+. With this geometrical interpretation in mind, one could speculate that Method B approximates the derivative better.

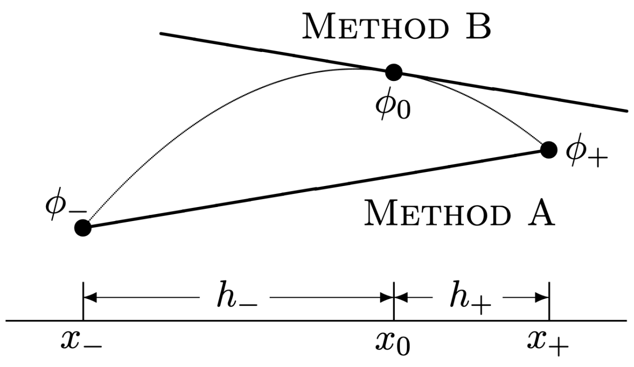


Figure 2: The geometrical interpretation of the first order derivative approximations of Method A and Method B. Source: [2, Section 1.5.1].

A standard way of comparing discretization methods is to use the local truncation error τ, which represents the error introduced by our approximations. We write out the corresponding local truncation error for both methods:

$$\begin{aligned}&\text{Method A: } \tau_A = -\frac{1}{2}(h_+ - h_-)\phi_{xx} - \frac{1}{6}\frac{h_+^3 + h_-^3}{h_+ + h_-}\phi_{xxx} + O(h^3),\\ &\text{Method B: } \tau_B = -\frac{1}{6}h_+h_-\phi_{xxx} + O(h^3).\end{aligned}$$

One can notice that for τ_A there exists a term containing the coefficient proportional to the second order derivative. However, by construction, the term containing ϕ_{xx} cancels out for τ_B, for Method B. Therefore, Method A contains an additional $O(h_+ - h_-)$ term proportional to ϕ_{xx}, which implies the error is worse for Method A. Furthermore, for Method A the coefficient for ϕ_{xxx} is never smaller than the corresponding coefficient for Method B:

$$\frac{1}{6}\frac{h_+^3 + h_-^3}{h_+ + h_-} = \frac{1}{6}(h_+^2 - h_+h_- + h_-^2) \geq \frac{1}{6}h_+h_-.$$

We can also estimate the order of accuracy for each method. On a non-uniform grid, this depends on the underlying grid construction. It has been shown that on grids obtained from a coordinate transformation (which applies to the abrupt grid), both methods have a second-order local truncation error [2, Section 1.5.1]. The second-order accuracy of Method B is straightforward as it can be noted by the h_+h_- term in its local truncation error.

For Method A, we provide a brief justification here, showing that indeed if the non-uniform grid originates from a coordinate transform $x = x(\xi)$ then the truncation error of Method A is also second order accurate despite the term $-\frac{1}{2}(h_+ - h_-)$ in τ_A. The proof follows a similar approach as in [2, Section 1.7], where fourth-order accuracy was shown for another discretization method. A more precise proof is given in [5], where grid conditions for second-order accuracy are specified. We have the following, when one uses the chain rule on the first order derivative:

$$\frac{d\phi}{dx} = (\frac{d\phi}{d\xi})/(\frac{dx}{d\xi}).$$

We take a uniform grid ξ with grid cell size Δ, then

$$\begin{aligned} \frac{d\phi}{d\xi} &= \frac{\phi_+ - \phi_-}{2\Delta} + O(\Delta^2), \\ \frac{dx}{d\xi} &= \frac{x_+ - x_-}{2\Delta} + O(\Delta^2) \\ &\implies \\ \frac{d\phi}{dx} &= \frac{\phi_+ - \phi_-}{x_+ - x_-} + O(\Delta^2), \end{aligned}$$

which implies second-order accuracy.

On the exponential grid however, with the stretching factor $h_+/h_- = \text{const.} \neq 1$, the local truncation error of Method A is only first-order accurate. This is worse than it is for Method B, for which the error remains second-order accurate [4, Section 2].

At first glance, the analysis appears to favor Method B. Later sections will highlight that this can be misleading. The local truncation error is not the only element governing the global error of discretizations. To compare both methods, we must find an approximation for the second order derivative, which will allow us to obtain the resulting discretized systems.

4.3 The Approximation of ϕ_{xx} and the Resulting Systems

As stated before, for both finite difference approximations, the second order derivative ϕ_{xx} will be approximated in the same way using the central-difference approximation.

Combining $h_- \times (7) + h_+ \times (8)$ we get:
Approximation of ϕ_{xx}:

$$h_-\phi_+ + h_+\phi_- = (h_+ + h_-)\phi_0 + \frac{1}{2}h_+h_-(h_+ + h_-)\phi_{xx} + \frac{1}{6}h_+h_-(h_+^2 - h_-^2)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$\frac{1}{2}h_+h_-(h_+ + h_-)\phi_{xx} = h_-\phi_+ + h_+\phi_- - (h_+ + h_-)\phi_0 - \frac{1}{6}h_+h_-(h_+^2 - h_-^2)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$\phi_{xx} = \frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} - \frac{1}{3}(h_+ - h_-)\phi_{xxx} + ...$$

$$\Longrightarrow$$

$$\phi_{xx} \approx \frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} \tag{14}$$

As we have the approximations for ϕ_x and ϕ_{xx}, we discretize the convection-diffusion equation (2). The governing discretized equations for both methods will be given by:
Resulting system (Method A):

$$\frac{\phi_+ - \phi_-}{h_+ + h_-} - k\frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} = 0$$

$$\Longleftrightarrow$$

$$\begin{aligned}\phi_-\left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)}\right) + \phi_0\left(k\frac{1}{\frac{1}{2}h_+h_-}\right) + \\ +\phi_+\left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)}\right) = 0\end{aligned} \tag{15}$$

Resulting system (Method B):

$$\frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)} - k\frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} = 0$$

$$\Longleftrightarrow$$

$$\begin{aligned}\phi_-\left(\frac{-h_+}{h_-(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)}\right) + \phi_0\left(\frac{h_+ - h_-}{h_+h_-} + k\frac{1}{\frac{1}{2}h_+h_-}\right) + \\ +\phi_+\left(\frac{h_-}{h_+(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)}\right) = 0\end{aligned} \tag{16}$$

From these expressions we have derived two solvable linear systems representing both methods, where the coefficient matrix is tridiagonal. The full derivation of each system is given in the Appendix B.1, which also emphasizes the careful consideration of both boundary conditions. From the boundary conditions it can be seen that the last element of the right-hand-side vector in the resulting linear system will be non-zero.

For both of these methods, the authors of [6] have theoretically proven second-order convergence on grids where the ratio between the smallest and the largest grid cell size is bounded.

With the resulting systems in mind, we turn to the numerical results of both methods, where we start by verifying the second-order convergence of both methods.

4.4 Numerical Results

In the numerical results section, we present various plots that illustrate how well do the discretization methods approximate the analytical solution (3). The notion of the global error is also introduced, which describes how much the analytical solution differs from the discrete solution. Numerically, this has been obtained using:

$$\|\phi_{\text{exact}} - \phi\|_2, \tag{17}$$

computed with the trapezoidal rule because of the non-uniform grid structure [4, Section 3.3].

4.4.1 Second Order Convergence

As noted before, on grids where the ratio between the smallest and the largest grid cell size is bounded, which is true for both the introduced abrupt and the exponential grid, both methods show second-order convergence. In simple terms, on fine grids this implies that when we increase the number of grid cells by a factor of 2, we decrease the global error by a factor of 4. This we can indeed demonstrate with the following log-log plot, on which the underlying grid for discretizations is the abrupt grid.

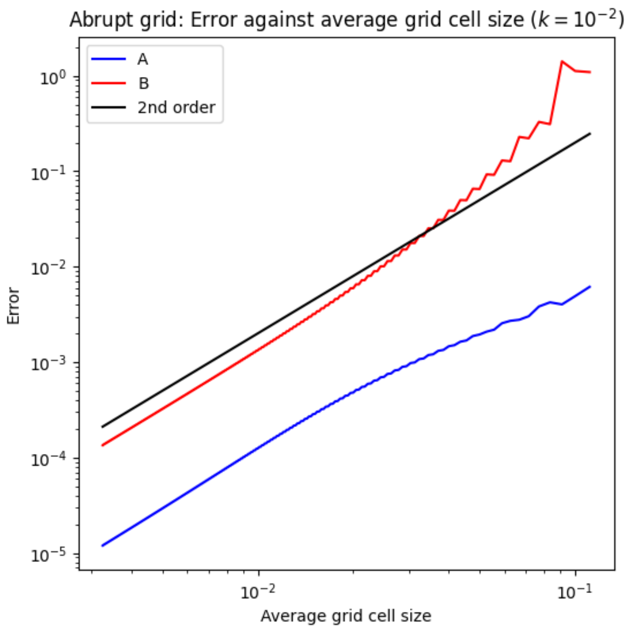


Figure 3: Demonstrating the second order convergence on the abrupt grid.

The plots show the discretization error (17) on the y-axis versus the average grid cell size on the x-axis. Meaning, on the left side of this graph we have errors on fine grids and on the right

- coarser grids. The number of used grid cells varies from 10 to 300.

When the grids are fine both error lines are indeed parallel to the second order line in black which showcases the second-order convergence mentioned before. However, as the grid gets coarser we can see that the red line representing error of Method B starts to increase.

We can also note the position of the error lines for both methods. The error of Method B is always higher than the error for Method A in blue. A similar result can be noted on the exponential grid [1, Section 3.1].

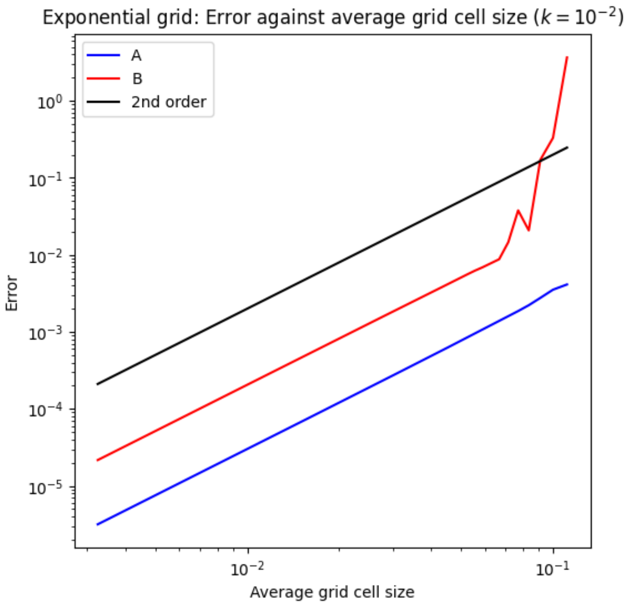


Figure 4: Demonstrating the second order convergence on the exponential grid.

On the exponential grid, the errors for both methods are lower, likely because of the smoother change in grid cell size. However, we again see Method B breaking second-order convergence as the grid gets coarser. For grids with less than 30 grid cells, the error for Method B starts to rapidly increase.

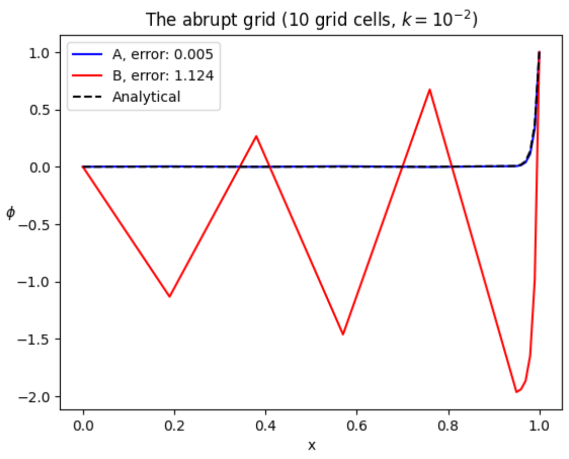
This is consistent behavior for method B [2, Section 1.5.3]. On finely constructed grids the method exhibits the correct convergence rate (although always worse than Method A). Nevertheless, when the grid gets coarser Method B breaks down.

Remark. *The black line representing the second-order line, has been plotted using* $Error(h_{avg}) = 20h_{avg}^2$ *on the log-log scale, where* h_{avg} *denotes the average grid cell size. The coefficient* 20 *has been chosen arbitrarily, for visual clarity. Mainly, to demonstrate that the error line of Method B indeed increases, as it goes over the black line. It does not prove anything related to the second-order convergence as that can be shown using only the slope of the function.*

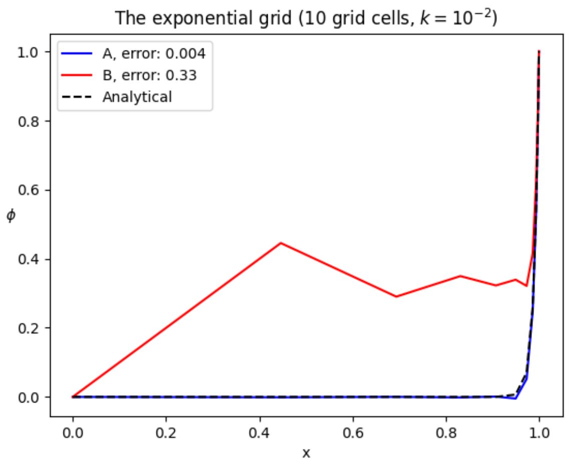
Remark. *One can notice wiggles in Figure 3 plotting the 2nd order convergence on the abrupt grid. This can be explained by the phenomena called even-odd decoupling. Essentially, on* *the abrupt grid, it matters whether the grid point corresponding to the boundary of the two uniform parts is an even-numbered point. Even-numbered grid points are better coupled to the first boundary condition [4, Section 3.2]. It is possible to construct abrupt grids that do not experience bad coupling, yet that is outside the scope of this project.*

4.4.2 Coarse Grid Approximations

To understand characteristics of each discretization method, we now examine the computed approximations. We specifically test how well the approximations of both methods match the analytical solution when the grid is coarse. The following figures show results on both grids with 10 grid cells.



(a) Methods applied on the abrupt grid



(b) Methods applied on the exponential grid

Figure 5: Resulting approximations from finite difference methods on a coarse grid.

In the given plots, the blue line representing the approximation of Method A closely matches the analytical solution given with the black, dashed line. However, Method B showcases poor results, exhibiting oscillations. On the abrupt grid, the resulting global error of Method B even exceeds 1.

The results are quite counterintuitive. The local truncation error, which is usually a decent estimator of a method's behavior is larger for Method A yet it gives better approximations. This unexpected outcome calls for further investigation, specifically an investigation concerning the equations governing the global error of discretizations.

4.5 Global Error Analysis

Following derivations in [4, Section 4], we will derive the equation that relates the local truncation error to the global error.

The discrete solution ϕ satisfies the following linear system:

$$L\phi = f, \tag{18}$$

where the matrix L denotes the coefficient matrix of the discretization and f denotes a vector originating from boundary conditions.

The exact solution ϕ_{exact} satisfies a similar linear system. However, this system includes the local truncation error τ, which arises from the approximation of each derivative:

$$L\phi_{\text{exact}} = f + \tau. \tag{19}$$

Subtracting equation (18) from equation (19) yields the expression for the global error:

$$\phi_{exact} - \phi = L^{-1}\tau. \tag{20}$$

The global error depends not only on the local truncation error but also on the inverse of the coefficient matrix. This provides a clear explanation of the poor results shown by Method B.

Roughly speaking, when the diagonal of L is weakened the smallest eigenvalues may move towards 0. This increases L^{-1}, therefore increasing the global error. We can hypothesize that for Method B the increase of the inverse of the coefficient matrix outweighs the decreased truncation error τ_B [1, Section 3.2].

To demonstrate this formally, we investigate both the coefficient matrix of Method A and Method B in the following section.

4.6 Coefficient Matrix Analysis

4.6.1 Coefficient Matrix of Method A

The coefficient matrix of Method A, which we denote by L_A, is a tridiagonal matrix, whose form we get from the discretized equation given in (15):

$$\phi_- \left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_0 \left(k\frac{1}{\frac{1}{2}h_+ h_-} \right) + \\ + \phi_+ \left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right)$$

We will show that such a matrix has some valuable properties for a discretization method: specifically the coefficient matrix of Method A has only eigenvalues with positive real parts. In the proof we will use Bendixson's Theorem A.5, lemma about scaling matrices with a positive definite matrix Lemma A.4 and standard linear algebra fundamentals, which can be found in the Appendix A.

We now turn to the main theorem of the report, which will note many beneficial properties of Method A.

Theorem 4.1 (L_A - positive-stable)**.** *The coefficient matrix of Method A, L_A is positive-stable (i.e. all eigenvalues have a positive real part) for any underlying grid construction.*

We follow and extend the proof given in [2, Section 1.5.3].

Proof. We start by scaling the coefficient matrix L_A with the following diagonal matrix, which incorporates the local grid cell sizes:

$$H = \operatorname{diag}(h_- + h_+).$$

Note, for each diagonal entry of this matrix the respective h_- and h_+ values will differ. The matrix H is a positive definite matrix, as its values on the diagonal are always positive. The resulting scaled matrix will have the following form:

$$HL_A = \begin{pmatrix} \ddots & & \\ & 1 - k\frac{1}{\frac{1}{2}h_-} & \\ -1 - k\frac{1}{\frac{1}{2}h_-} & k\frac{h_+ + h_-}{\frac{1}{2}h_+ h_-} & 1 - k\frac{1}{\frac{1}{2}h_+} \\ & -1 - k\frac{1}{\frac{1}{2}h_+} & \\ & & \ddots \end{pmatrix}$$

The matrix will have such a structure throughout all of its rows with different h_- and h_+ values.

We further show that the scaled coefficient matrix HL_A is positive-stable, by noticing the following decomposition:

$$\underbrace{\begin{pmatrix} \ddots & & \\ & 1 - k\frac{1}{\frac{1}{2}h_-} & \\ -1 - k\frac{1}{\frac{1}{2}h_-} & k\frac{h_+ + h_-}{\frac{1}{2}h_+ h_-} & 1 - k\frac{1}{\frac{1}{2}h_+} \\ & -1 - k\frac{1}{\frac{1}{2}h_+} & \\ & & \ddots \end{pmatrix}}_{HL_A} = \underbrace{\begin{pmatrix} \ddots & & \\ & 1 & \\ -1 & 0 & 1 \\ & -1 & \\ & & \ddots \end{pmatrix}}_{\text{skew-symmetric}} + \underbrace{\begin{pmatrix} \ddots & & \\ & -k\frac{1}{\frac{1}{2}h_-} & \\ -k\frac{1}{\frac{1}{2}h_-} & k\frac{h_+ + h_-}{\frac{1}{2}h_+ h_-} & -k\frac{1}{\frac{1}{2}h_+} \\ & -k\frac{1}{\frac{1}{2}h_+} & \\ & & \ddots \end{pmatrix}}_{\text{symmetric}}$$

With such a decomposition, we note that the scaled coefficient matrix is a sum of a skew-symmetric matrix and a symmetric matrix.

The symmetric matrix is also weakly diagonally dominant A.3, which is shown using the following inequality, which holds true for each row:

$$\left| -k\frac{1}{\frac{1}{2}h_-} \right| + \left| -k\frac{1}{\frac{1}{2}h_+} \right| = k\frac{h_+ + h_-}{\frac{1}{2}h_+ h_-} \leq \left| k\frac{h_+ + h_-}{\frac{1}{2}h_+ h_-} \right|.$$

As the symmetric matrix is weakly diagonally dominant it is in fact positive definite.

We apply Bendixson's Theorem A.5. The theorem states that all eigenvalues of a matrix lie in the rectangle formed by the eigenvalues of the symmetric part and the eigenvalues of the skew-symmetric part.

The skew-symmetric matrix will only have purely imaginary eigenvalues. The symmetric matrix, as it is positive definite, will only have positive real eigenvalues. Bendixson's theorem then implies that all eigenvalues of the resulting matrix HL_A will be situated in a rectangle on the right half of the complex plane. This proves that indeed all eigenvalues of HL_A will have a positive real part, therefore the matrix is positive-stable.

One can illustrate this with the following example, where in the picture bellow we have plotted the eigenvalues of the skew-symmetric and the symmetric part together with the rectangle enclosing the eigenvalues of the resulting matrix.

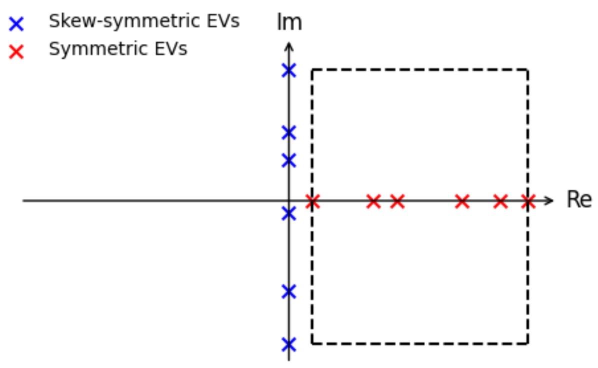


Figure 6: Bendixson's theorem applied to the eigenvalues of the skew-symmetric and the symmetric parts of HL_A.

As H is a positive definite matrix, its inverse H^{-1} will also be positive definite. We now use the lemma on positive scaling shown in Appendix A.4.

The matrix HL_A is positive-stable and H^{-1} is positive definite. Therefore, the matrix product $H^{-1}HL_A = L_A$ is positive-stable, meaning its eigenvalues have a positive real part.

□

There are several benefits of having such a coefficient matrix. For one, the matrix is always non-singular, implying that the linear system in (18) has a unique solution and its global error can never become infinite.

4.6.2 Coefficient Matrix of Method B

In contrast to Method A, the coefficient matrix of Method B, denoted by L_B, lacks certain properties. The reader is reminded of its structure, introduced in (16):

$$\phi_- \left(\frac{-h_+}{h_-(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_0 \left(\frac{h_- - h_+}{h_+ h_-} + k\frac{1}{\frac{1}{2}h_+ h_-} \right) +$$
$$+\phi_+ \left(\frac{h_-}{h_+(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right).$$

It suffices to examine the diagonal element of this matrix, to understand why the method performs poorly when the grid becomes too coarse [2, Section 1.5.3]. The diagonal elements of L_B have the following structure:

$$\frac{h_- - h_+}{h_+ h_-} + k\frac{1}{\frac{1}{2}h_+ h_-} = \frac{(h_- - h_+) + 2k}{h_+ h_-}.$$

With the fact that h_-, h_+ and k are positive values, one can note that when the grid gets coarser (implying h_- and h_+ increase) or k gets smaller, such a term, and therefore the diagonal entries of L_B will decrease. It is even possible for the diagonal entries to become negative as $h_- > h_+$ is a standard occurrence on our constructed grids introduced in section 3.2.

Example:

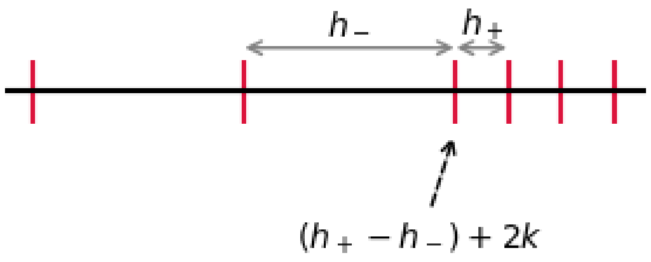


Figure 7: The diagonal element for the abrupt grid at the transition between the two uniform parts.

For the coefficient matrix of Method B on the abrupt grid, the diagonal entries for all except one entry will be equal to $\frac{2k}{h_+ h_-}$ as then $h_- = h_+$ because of the grids semi-uniform structure. That said, the middle diagonal entry, when the two uniform parts swap (shown in Figure 7) will have that h_- is significantly larger than h_+. Therefore, $h_- - h_+$ is negative and if $2k$ cannot compensate for it, the term remains negative.

For the exponential grid we will have that for all diagonal entries $h_- > h_+$. However here, the difference between h_- and h_+ is not that drastic as in the abrupt grid scenario. For the exponential grid, the grid size changes smoothly throughout the grid.

In conclusion, the diagonal entries weaken as the underlying grid gets coarser or as k gets smaller. With decreasing diagonal entries, the smallest eigenvalues of L_B can approach 0. While for non-symmetric matrices we cannot simply state that $\|L_B^{-1}\|$ is inversely proportional to the smallest eigenvalue of L_B, the underlying principle still holds. Eigenvalues approaching zero indicate that the matrix is becoming ill-conditioned and that the norm of its inverse grows. This in turn increases the global error as noted by the equation describing it (20). With further decrease in the diagonal the matrix can even obtain eigenvalues with a negative real part. Furthermore, the eigenvalues of L_B with some select parameters can even become zero. This causes the matrix to become singular, because of which the global error can become unbounded.

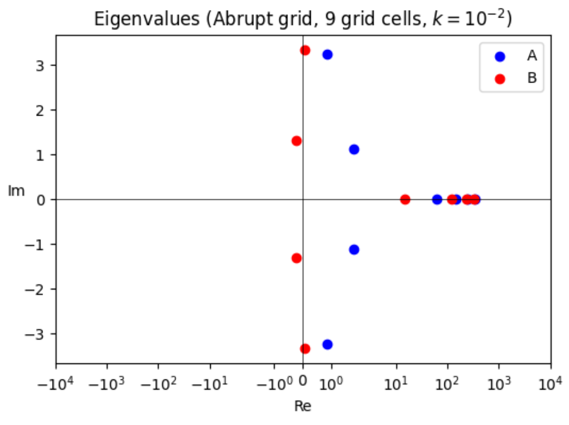
4.6.3 Consequences

By reintroducing the time derivative in our convection-diffusion equation, as in the partial differential equation (1), one can note more problems with having a poor coefficient matrix. If one were to apply the discretization in space of Method B, the discretized partial differential equation would form a linear system of ODEs:

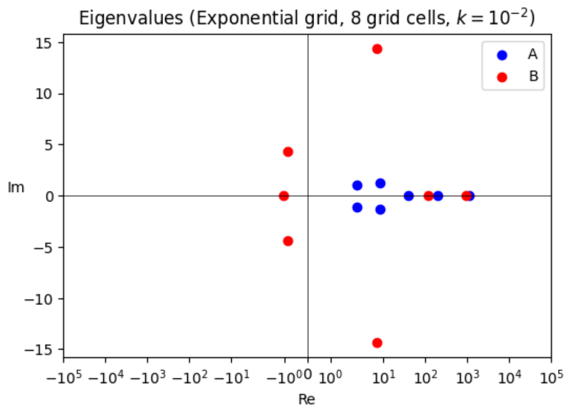
$$\begin{gathered}\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} - k\frac{\partial^2 \phi}{\partial x^2} = 0 \\ \Longrightarrow \\ \frac{d\phi}{dt} + L_B\phi = 0 \\ \Longleftrightarrow \\ \frac{d\phi}{dt} = -L_B\phi.\end{gathered}$$

As noted in the previous section, the coefficient matrix of Method B can have eigenvalues with negative real parts. Hence, if L_B has eigenvalues with negative real parts, the matrix $-L_B$ has eigenvalues with a positive real part. From linear system theory, we know that then the linear system of ODEs is unstable. Such results do not match real-life properties as the underlying convection-diffusion PDE is inherently stable. We further show examples in which time integration is not stable.

Imagine a scenario, in which we have to discretize the convection-diffusion equation (1) with a diffusive coefficient equal to $k = 10^{-2}$. The poor properties of Method B restrict us to use a grid that is too coarse, as we might end up with an unstable system when time integration is introduced. Note the following two plots, where on the complex plane we have plotted the eigenvalues of the coefficient matrices of Method A and Method B.



(a) Methods applied on the abrupt grid

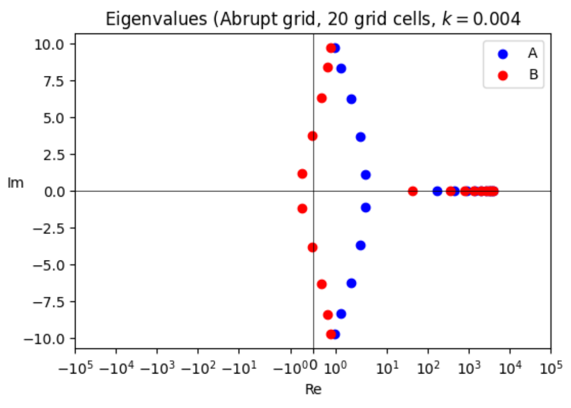


(b) Methods applied on the exponential grid

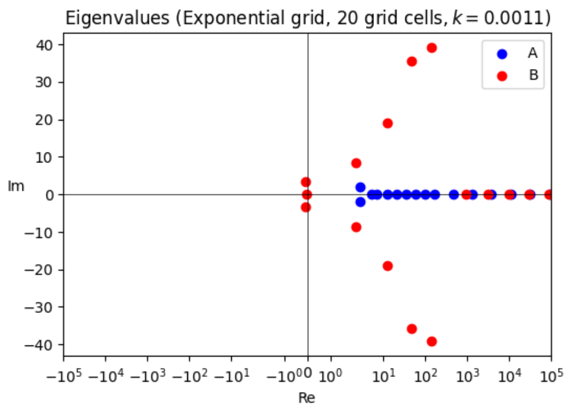
Figure 8: Eigenvalues of Method A and Method B for a fixed k.

For a fixed $k = 10^{-2}$, when one uses Method B on the abrupt grid with 9 grid cells, we can note that the eigenvalues of L_B do in fact cross into the left half of the complex plane. For the exponential grid, the same happens with the grid cell number being reduced to 8. Hence, Method B would give unrealistic results if one were to investigate the evolution in time of ϕ with such parameters. However, the eigenvalues of the coefficient matrix of Method A stay always in the right half of the plane (as also shown in Theorem 4.1). No matter what the parameters for Method A are, the resulting linear system of ODEs will be stable.

One could also imagine a scenario where for all discretization research, we have been given a computer whose CPU limits only support grids with 20 grid cells. Then the use of Method B is restricted by the value of k we take as our diffusive coefficient.



(a) Methods applied on the abrupt grid



(b) Methods applied on the exponential grid

Figure 9: Eigenvalues of Method A and Method B for a fixed number of grid cells.

The figures illustrate that if we were to take a diffusive coefficient k equal to or lower than 0.004 on the abrupt grid, using Method B would result in an unstable time integration. Similarly, on the exponential grid, Method B would produce an unstable system for $k \leq 0.0011$. Nevertheless, one cannot change what diffusive coefficient real life scenarios might have. For example, having $u = 1$ and $k = 0.001$ can still model real-life problems. Therefore, having such restrictions continues to show the poor qualities of Method B.

4.7 Conclusion in the Framework

Based on numerical results and coefficient matrix analysis we can conclude that Method A outperforms Method B. Although Method B has a better local truncation error, it introduces a larger global error and fails to approximate the analytical solution properly. Moreover, the coefficient matrix of Method B exhibits several problematic properties. Its diagonal entries weaken with the grid getting coarser, its eigenvalues can have negative real parts, and it can even become singular. Method A on the other hand has a coefficient matrix that can never become singular and is positive-stable. As noted in the previous section, Section 4.6.3, with select k values or with select computational restraints Method A is the only one that produces physical results.

Remark. *It is interesting to note that the superior method, Method A has been overlooked. Because of its smaller local truncation error, Method B used to be crowned as the better method. Note reference [7], in which the authors state how Method B gives poor results on abrupt grids. However, instead of recommending discretization with Method A, the paper concludes that smoother grids should be implemented to improve Method B. As we have shown in this section, Method A produces better results no matter if the grid is abrupt or smooth.*

5 Finite Volume Methods

We now enter the finite volume framework. The structure of the following section will closely follow the one concerning finite difference methods. We will examine several finite volume methods, noting that, as before some methods obtain a worse local truncation error than others yet globally perform better. We conclude the section by analyzing the coefficient matrices, which again explains the numerical results.

For finite volume methods (FVM), we divide the domain into a set of non-overlapping control volumes. Control volumes enforce conservation of the total flux. By putting our convection-diffusion ODE into the conservation form, we get the corresponding equation describing the flux:

$$\phi_x - k\phi_{xx} = 0$$
$$\Longrightarrow$$
$$\frac{d}{dx}\left(\phi - k\phi_x\right) = 0,$$

therefore, the flux on each cell-face is given by

$$q(\phi) = \phi - k\phi_x. \tag{21}$$

With the resulting flux, the discretization is generated using the fact that the net flux into each control volume is zero. This yields the discrete conservation law, with $q_{i+\frac{1}{2}}$ and $q_{i-\frac{1}{2}}$ denoting the fluxes on the right and left cell-face respectively:

$$q_{i+\frac{1}{2}} - q_{i-\frac{1}{2}} = 0. \tag{22}$$

For future reference, the flux (21) can be decomposed into a sum of the convective and the diffusive flux:

$$q(\phi) = q^{\text{conv}} + q^{\text{diff}},$$
$$\text{where } q^{\text{conv}}(\phi) = \phi, \quad q^{\text{diff}}(\phi) = -\,k\,\phi_x.$$

This decomposition allows us to approximate the convective and diffusive flux derivatives separately using finite volume methods. It is useful, as the main difficulty lies in approximating the convective operator, as noted in the finite difference framework.

A noteworthy challenge for finite volume methods, specifically on non-uniform grids, is how to define the placement of each control volume. We introduce two methods: the vertex-centered method and the cell-centered method.

The vertex-centered method will have its cell-faces defined to lie in the middle of each grid cell. However, for the cell-centered method, the grid points will lie in the middle of each control volume. Note, for uniform grids the following comparisons would not be made. On uniform grids both control volume constructions are equivalent.

The finite volume derivations follow the construction shown in [2, Section 1.6.1]

5.1 Vertex-Centered Method

We start with the vertex-centered method with each cell-face lying in the middle of adjacent grid points. The underlying cell-face placement is given in the drawing below. The orange dashed lines represent the control volumes, where each control volume is of length $\frac{h_- + h_+}{2}$.

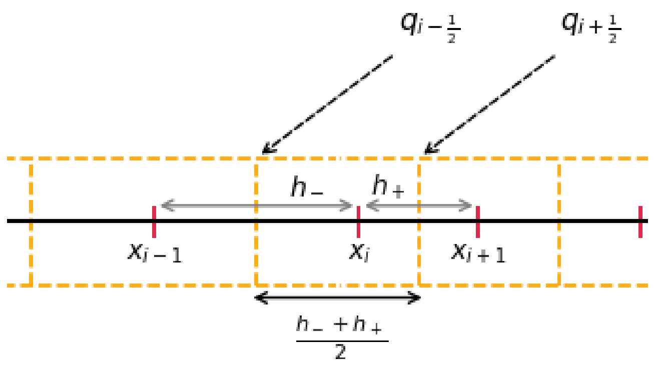


Figure 10: Control volume grid for the vertex-centered FVM.

The convective flux is derived as the average of function values on adjacent grid points, as the cell-face lies in the middle of subsequent grid points. This process describes linear interpolation. For the diffusive flux we use the difference of adjacent function values divided by their corresponding grid point difference. Both the convective and diffusive flux will have second-order accuracy with such an approach.

The flux for each grid point at the right and left cell-face respectively are given by:

$$\begin{aligned} q_{i+\frac{1}{2}} &= \frac{\phi_+ + \phi_0}{2} - k\frac{\phi_+ - \phi_0}{h_+}, \\ q_{i-\frac{1}{2}} &= \frac{\phi_0 + \phi_-}{2} - k\frac{\phi_0 - \phi_-}{h_-}. \end{aligned}$$

Using the discrete conservation law (22) with the calculated fluxes we get the following discretiztion:

Resulting system (Vertex-centered FVM):

$$q_{i+\frac{1}{2}} - q_{i-\frac{1}{2}} = 0$$

$$\Longrightarrow$$

$$\frac{\phi_+ + \phi_0}{2} - k\frac{\phi_+ - \phi_0}{h_+} - \frac{\phi_0 + \phi_-}{2} + k\frac{\phi_0 - \phi_-}{h_-} = 0$$

$$\Longleftrightarrow$$

$$\phi_-\left(-\frac{1}{2} - k\frac{1}{h_-}\right) + \phi_0\left(k(\frac{1}{h_+} + \frac{1}{h_-})\right) + \phi_+\left(\frac{1}{2} - k\frac{1}{h_+}\right) = 0. \tag{23}$$

It turns out that the resulting system of the vertex-centered FVM fully matches Method A from the finite difference framework. Again, full system derivation can be found in the appendix B.2.1.

The fact that the vertex-centered FVM equals the FDM of Method A might indicate that finite volume methods will give adequate results. However, as pointed out before, many control volume construction approaches can be made. In the following section we turn to the cell-centered grid definition. On it, we will construct two finite volume methods one seemingly better than the other in its truncation error. Interestingly, we get similar results as we displayed when comparing Method A and Method B. Also in a finite volume framework the discretization with a larger local truncation error yields better approximations.

5.2 Cell-Centered Methods

In the vertex-centered layout, no face coincides with the given boundaries $x = 0$ or $x = 1$. In this section, we introduce the cell-centered approach, for which each grid point will lie in the middle of the control volume. A key consequence of such an approach is that cell-faces now lie on the boundaries. However, because of this, we need to redefine where the grid points for our non-uniform grids will be situated.

To accurately construct the grid points, we use the previous grid definitions from 3.2 to define where the cell-faces will lie. We then force the grid points to lie in the middle of each control volume.

Ultimately, this implies that we have one more grid point we need to solve for. The cell-centered grid structure gets formed from:

$$x^C_{i+1} = x_i + \frac{1}{2}(x_{i+1} - x_i), \quad \text{for } i = 0, ..., N-1, \tag{24}$$
$$\text{with } x^C_0 = 0, \quad x^C_{N+1} = 1,$$

where x^C_i denote the new grid points for the cell-centered grid and x_i denote the grid points on the original grid as well where the cell-faces lie. On such a grid, we are solving for values lying on the grid points x^C_i for $i = 1, ..., N$. At the function values $\phi(0) = \phi_0 = 0$ and $\phi(1) = \phi_{N+1} = 1$ we will have the boundaries. We define h^C_- and h^C_+ to denote cell sizes for this cell-centered finite volume method. They follow the same logic as on the grids before:

$$h^C_- = x^C_i - x^C_{i-1}, \quad h^C_+ = x^C_{i+1} - x^C_i.$$

Example (Cell-centered abrupt grid):

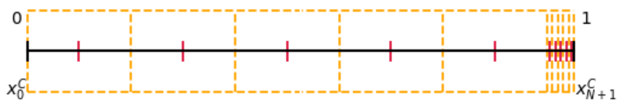


Figure 11: Cell-centered cell-face placement.

For both methods introduced in this section, just like in the finite difference framework, the approximation of the second order derivative (resulting from the same diffusive fluxes) will match. We will again note that just the change in the approximation of the convective flux has a large impact on the global error of the method.

5.2.1 Precise Cell-Centered Construction

We first define the method for which the convective flux is calculated using a precise linear interpolation between grid points. The idea is to use this interpolation to accurately compute the value of ϕ at each cell-face. We introduce notation to define the length for the control volumes, which we denote by H_0 (see figure below):

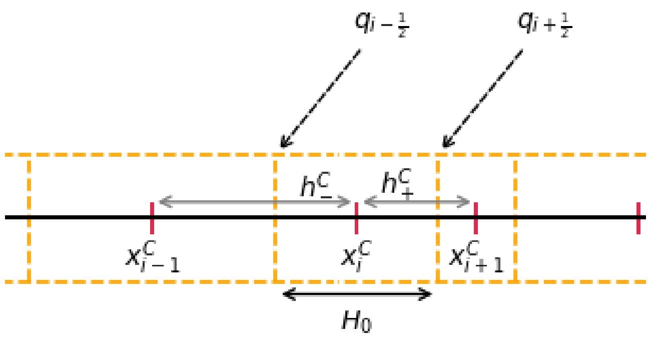


Figure 12: Control volume grid for the cell-centered FVM.

With the notation as shown in the figure, we can compute the fluxes at both cell-faces. The computation of the diffusive flux does not differ from the vertex-centered method. For the convective flux we use precise linear interpolation. That said, the formula is now more involved than linear interpolation in the vertex-centered method. The resulting fluxes are given by:

$$q_{i+\frac{1}{2}} = \phi_0 + \frac{H_0}{2h_+^C}(\phi_+ - \phi_0) - k\frac{\phi_+ - \phi_0}{h_+^C}, \tag{25}$$

$$q_{i-\frac{1}{2}} = \phi_0 - \frac{H_0}{2h_-^C}(\phi_0 - \phi_-) - k\frac{\phi_0 - \phi_-}{h_-^C}. \tag{26}$$

This seems like a natural choice of defining the convective flux, as we precisely describe where the cell-faces lie.

Therefore, by the conservation law (22), we derive the following cell-centered discretization, when convective fluxes are defined by linear interpolation:
Resulting system (Precise cell-centered FVM):

$$q_{i+\frac{1}{2}} - q_{i-\frac{1}{2}} = 0$$

$$\Longrightarrow$$

$$\phi_0 + \frac{H_0}{2h_+^C}(\phi_+ - \phi_0) - k\frac{\phi_+ - \phi_0}{h_+^C} - \phi_0 + \frac{H_0}{2h_-^C}(\phi_0 - \phi_-) + k\frac{\phi_0 - \phi_-}{h_-^C} = 0$$

$$\Longleftrightarrow$$

$$\phi_-\left(-\frac{H_0}{2h_-^C} - k\frac{1}{h_-^C}\right) + \phi_0\left(\frac{H_0}{2h_-^C} - \frac{H_0}{2h_+^C} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_+\left(\frac{H_0}{2h_+^C} - k\frac{1}{h_+^C}\right) = 0$$

With the grid point placement redefined, careful consideration of the boundary values is needed which is given in the Appendix B.2.2.

As exemplified in the finite difference section, the discretization of the first order derivative alone can make noticeable differences in the resulting approximation. We get the convective fluxes q^{conv} for each cell-face from (25) and (26):

$$\begin{aligned} q^{\text{conv}}_{i+\frac{1}{2}} &= \phi_0 + \frac{H_0}{2h^C_+}(\phi_+ - \phi_0), \\ q^{\text{conv}}_{i-\frac{1}{2}} &= \phi_0 - \frac{H_0}{2h^C_-}(\phi_0 - \phi_-). \end{aligned}$$

Using the conservation law, we get the following approximation of the first order derivative in the precise cell-centered method:

$$\begin{gathered} q^{\text{conv}}_{i+\frac{1}{2}} - q^{\text{conv}}_{i-\frac{1}{2}} = 0 \\ \Longleftrightarrow \\ \frac{H_0}{2h^C_+}(\phi_+ - \phi_0) + \frac{H_0}{2h^C_-}(\phi_0 - \phi_-) = 0. \end{gathered} \tag{27}$$

By dividing both sides with H_0 this result then gives the following approximation for the first-order derivative:

$$\phi_x \approx \frac{1}{2}\frac{\phi_+ - \phi_0}{h^C_+} + \frac{1}{2}\frac{\phi_0 - \phi_-}{h^C_-} \tag{28}$$

An interesting result that can be noted is that this expression, on the original grid setting, equals the average of the first derivative approximations of Method A and Method B. The relevance of such a discretized convective operator will be pointed out in later sections.

5.2.2 Jameson's Method

A noteworthy and yet counterintuitive approach was suggested by Antony Jameson [8, Section 2], who suggested that on a cell-centered finite volume grid, we also approximate the convective flux with the average of two subsequent nodes:

$$q^{\text{conv}}_{i+\frac{1}{2}} = \frac{\phi_+ + \phi_0}{2}. \tag{29}$$

Such an approach is counterintuitive as now we approximate the convective flux at the middle of two grid points, rather than at the actual location of the cell-face. This is illustrated in the following figure. The dashed green line represents where our convective flux lies in the Jameson's approach. However, this position does not match the orange line to its right representing the cell faces adjacent to the grid point x^C_i.

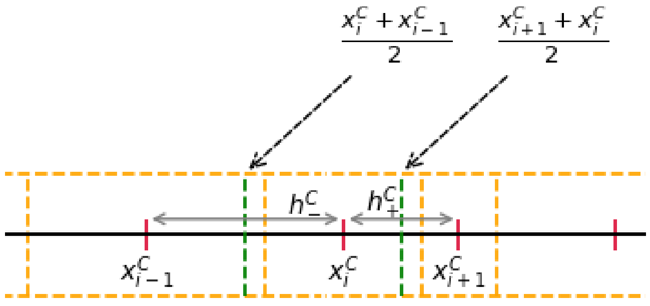


Figure 13: Jameson's approach.

Such an approach in fact aligns with the vertex-centered finite volume method, but as here the cell-faces are not centered between subsequent grid points this introduces a larger local truncation error.

The computation for the coefficient matrix does not differ from the vertex-centered method (23), however, we need to use the newly constructed cell-centered grid. Moreover, the boundary conditions differ from the vertex-centered method as the cell-faces here lie on the boundaries. The full coefficient matrix derivations are given in the Appendix B.2.3, yet the resulting system is given here:

Resulting system (Jameson's cell-centered FVM):

$$\phi_- \left(-\frac{1}{2} - k\frac{1}{h_-^C}\right) + \phi_0 \left(k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_+ \left(\frac{1}{2} - k\frac{1}{h_+^C}\right) = 0$$

To see how the Jameson's method approximates the first order derivative, one has to reintroduce the cell width H_0 as we calculate the fluxes in respective control volumes. With such a convective flux (29) the resulting approximation of the first order derivative will be given by the following:

$$\phi_x \approx \frac{\phi_+ - \phi_-}{2H_0}. \tag{30}$$

5.3 Local Truncation Error Comparison

Just as for the finite difference section, we compare the local truncation error of the first order derivative approximation for the precise cell-centered and Jameson's method.

We can express the first order derivative of the precise cell-centered construction (27) using the complete Taylor series derivation of Method A (10) and Method B (12). The truncation error τ_C of the precise cell-centered method will be the average of the truncation error of Method A and Method B adapted for the cell-centered grid:

$$\begin{aligned}\tau_C &= \frac{1}{2}H_0(\tau_A + \tau_B) \\ &= -\frac{1}{4}H_0(h_+^C - h_-^C)\phi_{xx} - \frac{1}{12}H_0((h_+^C)^3 + (h_-^C)^3)\phi_{xxx} + ...\end{aligned}$$

The control volume width H_0 makes the term involving ϕ_{xx} proportional to h^2 which implies the approximation is second-order accurate.

For the truncation error τ_J of the first order derivative of Jameson's method we have the following:

$$\phi_x = \frac{\phi_+ - \phi_-}{2H_0} + \tau_J.$$

We use the Taylor expansion combination (9) from the construction of the first order derivative for Method A combined with the cell-centered grid:

$$\phi_+ - \phi_- = (h_+^C + h_-^C)\phi_x + \frac{1}{2}((h_+^C)^2 - (h_-^C)^2)\phi_{xx} + \frac{1}{6}((h_+^C)^3 + (h_-^C)^3)\phi_{xxx} + ...$$

$$\Longleftrightarrow$$

$$\frac{\phi_+ - \phi_-}{2H_0} = \frac{h_+^C + h_-^C}{2H_0}\phi_x + \frac{(h_+^C)^2 - (h_-^C)^2}{4H_0}\phi_{xx} + \frac{(h_+^C)^3 + (h_-^C)^3}{12H_0}\phi_{xxx} + ...$$

$$\Longrightarrow$$

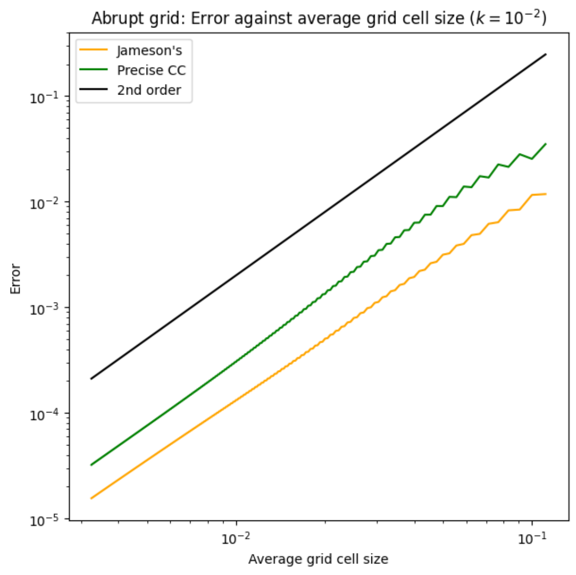
$$\tau_J = \left(\frac{h_+^C + h_-^C}{2H_0} - 1\right)\phi_x + \frac{(h_+^C)^2 - (h_-^C)^2}{4H_0}\phi_{xx} + \frac{(h_+^C)^3 + (h_-^C)^3}{12H_0}\phi_{xxx} + ...$$

We can note that the local truncation error τ_J contains an additional term proportional to the first order derivative, which is not included in the local truncation error τ_C. As $2H_0 \neq h_+^C + h_-^C$, on completely irregular grids, this term can even make the truncation error τ_J zeroth-order accurate, as has been noted in [2, Section 1.6.1]. The local truncation error for Jameson's method will be indeed larger than for the precise cell-centered construction. However, just as in the finite difference framework, the method with the larger local truncation error will be the one performing better as will be shown in the following results sections.

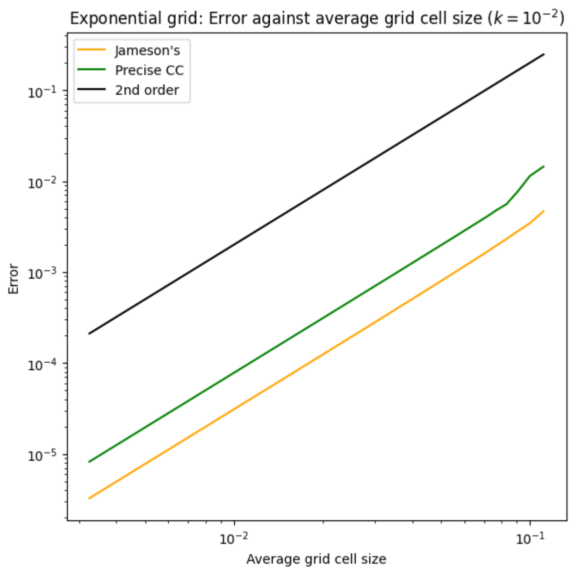
5.4 Numerical Results

5.4.1 Second-order convergence

For both the precise cell-centered and Jameson's methods, the authors of [6] have proven second order convergence, on grids where the ratio between the smallest and the largest cell is bounded. We begin the numerical analysis of finite volume methods by noting this in the following plots on both the abrupt and the exponential grid for $k = 10^{-2}$.



(a) Methods applied on the abrupt grid



(b) Methods applied on the exponential grid.

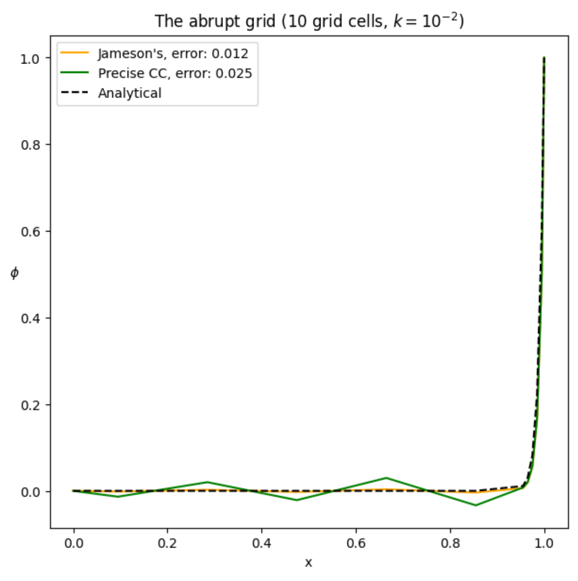
Figure 14: Demonstrating the second order convergence for finite volume methods

The same scenario as for the finite difference method results in Section 4.4.1 can be noticed. The green line representing the precise cell-centered method is always above the orange line representing the error for Jameson's method. The lines representing the error of both methods are parallel to the second order line when the grid is fine, yet when the grid gets coarser the error of the precise cell-centered method increases.

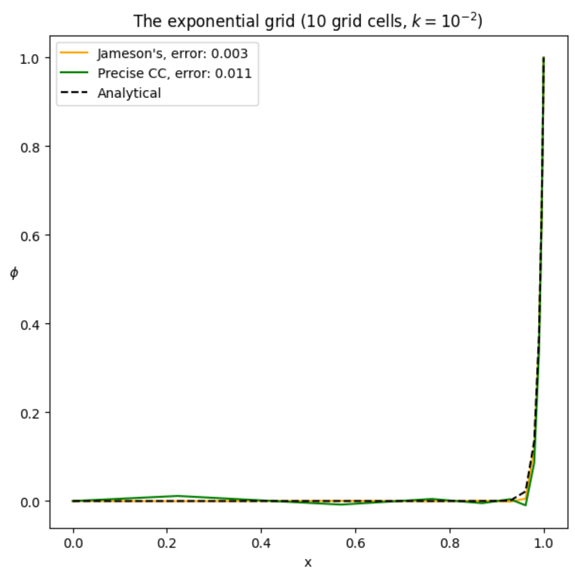
5.4.2 Coarse Grid Approximations

As one might have noted in the plots representing the second order convergence in Figure 14, the gap between the errors of the two finite volume methods is notably smaller than the large gap seen between the methods in the finite difference framework. This is also evident when plotting the approximation of the analytical solution of both the precise cell-centered and Jameson's method in Figure 15.

Although the error and the approximation of the precise cell-centered method give admirable results, the coefficient matrix still experiences the same drawbacks as Method B when time integration is reintroduced.



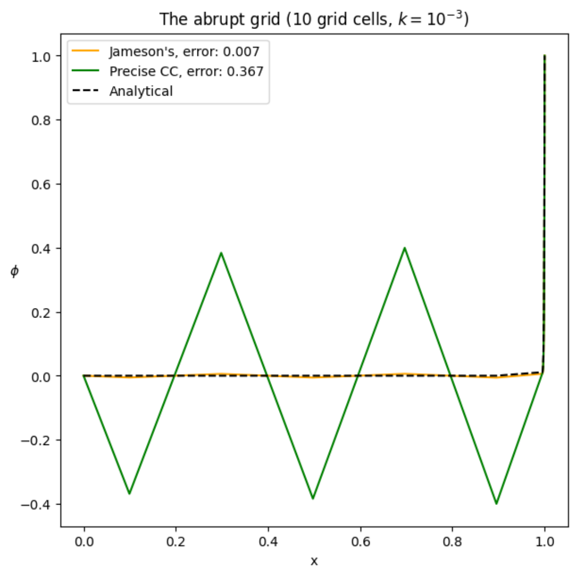
(a) Methods applied on the abrupt grid



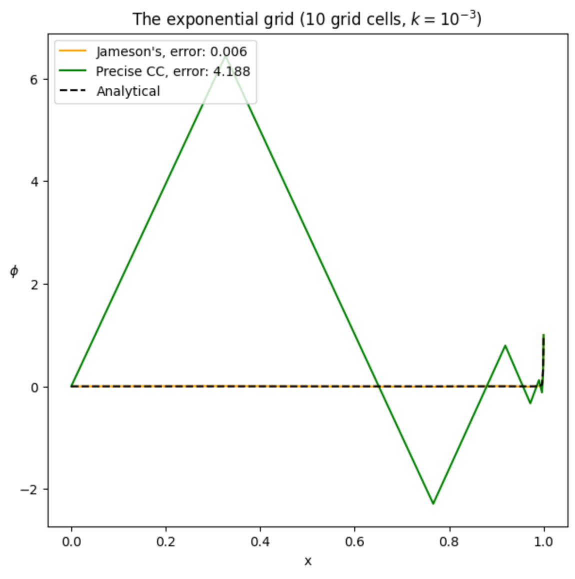
(b) Methods applied on the exponential grid

Figure 15: Resulting approximations of finite volume methods on a coarse grid.

Moreover, one can note that indeed Jameson's method gives significantly better approximations when we decrease the diffusive coefficient to $k = 10^{-3}$. Extremely poor results can be observed from the precise cell-centered method represented by the green line.



(a) Methods applied on the abrupt grid



(b) Methods applied on the exponential grid

Figure 16: Resulting approximations of finite volume methods on a coarse grid with $k = 10^{-3}$.

5.5 Coefficient Matrix Analysis

5.5.1 Coefficient Matrix of Jameson's Method

The coefficient matrix of Jameson's method follows the same construction as the coefficient matrix for Method A. The coefficient matrix of Jameson's method, further denoted by L_J is obtained from the following equation:

$$\phi_- \left(-\frac{1}{2} - k\frac{1}{h^C_-}\right) + \phi_0 \left(k(\frac{1}{h^C_+} + \frac{1}{h^C_-})\right) + \phi_+ \left(\frac{1}{2} - k\frac{1}{h^C_+}\right).$$

One can note the following decomposition of the matrix:

$$\underbrace{\begin{pmatrix} \ddots & & \\ & \frac{1}{2} - k\frac{1}{h^C_+} & \\ -\frac{1}{2} - k\frac{1}{h^C_-} & k(\frac{1}{h^C_+} + \frac{1}{h^C_-}) & \frac{1}{2} - k\frac{1}{h^C_+} \\ & -\frac{1}{2} - k\frac{1}{h^C_-} & \\ & & \ddots \end{pmatrix}}_{L_J} = \underbrace{\begin{pmatrix} \ddots & & \\ & \frac{1}{2} & \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ & -\frac{1}{2} & \\ & & \ddots \end{pmatrix}}_{\text{skew-symmetric}} + \underbrace{\begin{pmatrix} \ddots & & \\ & -k\frac{1}{h^C_+} & \\ -k\frac{1}{h^C_-} & k(\frac{1}{h^C_+} + \frac{1}{h^C_-}) & -k\frac{1}{h^C_+} \\ & -k\frac{1}{h^C_-} & \\ & & \ddots \end{pmatrix}}_{\text{symmetric}}$$

Just as in the proof concerning the coefficient matrix of Method A being always positive stable, for Theorem 4.1, the matrix L_J can be split into a skew-symmetric and a weakly diagonally dominant symmetric matrix. Therefore, we can conclude that also the coefficient matrix of Jameson's method will be positive-stable, having for all eigenvalues a positive real part and never becoming singular.

5.5.2 Coefficient Matrix of the Precise Cell-Centered Method

The coefficient matrix of the precise cell-centered method, further denoted as L_C is given by the expression:

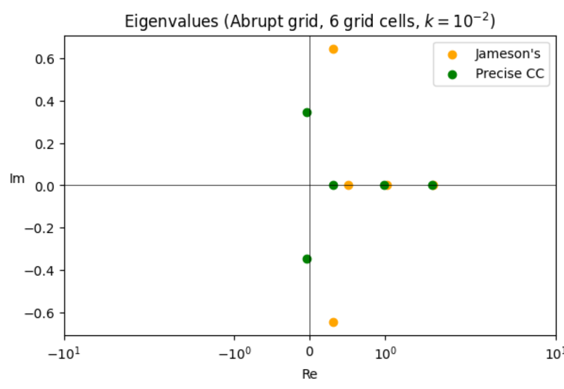
$$\phi_- \left(-\frac{H_0}{2h^C_-} - k\frac{1}{h^C_-}\right) + \phi_0 \left(\frac{H_0}{2h^C_-} - \frac{H_0}{2h^C_+} + k(\frac{1}{h^C_+} + \frac{1}{h^C_-})\right) + \phi_+ \left(\frac{H_0}{2h^C_+} - k\frac{1}{h^C_+}\right).$$

The diagonal term of such a matrix, is then given by:

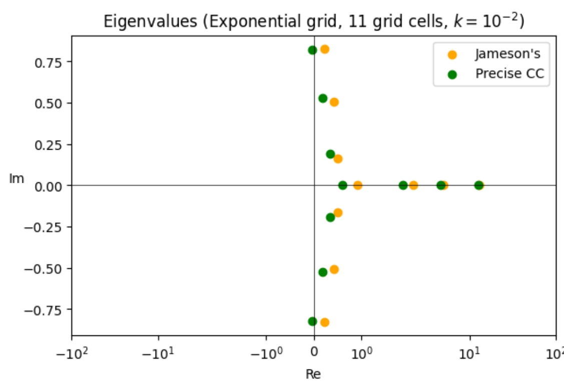
$$\frac{H_0}{2h^C_-} - \frac{H_0}{2h^C_+} + k(\frac{1}{h^C_+} + \frac{1}{h^C_-}) = \frac{H_0(h^C_+ - h^C_-) + 2k(h^C_- + h^C_+)}{2h^C_- h^C_+}, \tag{31}$$

which has the same problem as the coefficient matrix of Method B. When the grid gets coarser or we decrease k, the diagonal weakens, which can lead to smaller eigenvalues. Hence, the global error for the discretization increases.

Similar to the coefficient matrix of Method B, also the coefficient matrix of the precise cell-centered method can have negative eigenvalues. We note this in the following two figures, where we plot the eigenvalues of both the matrix L_J and L_C on the complex plane with k being kept constant at 10^{-2}.



(a) Methods applied on the abrupt grid

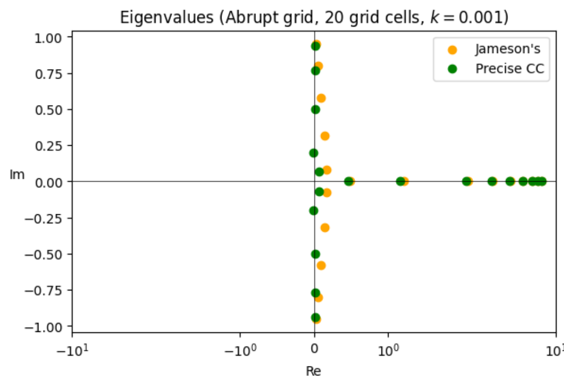


(b) Methods applied on the exponential grid

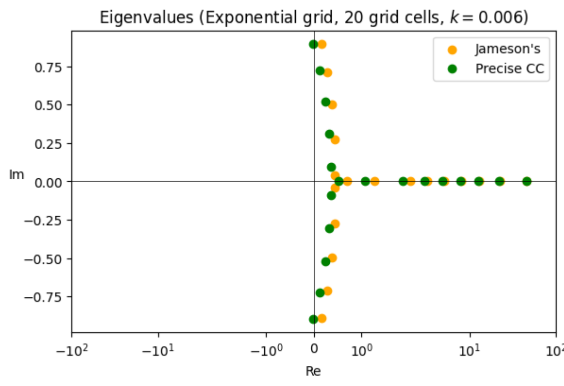
Figure 17: Eigenvalues of precise cell-centered method and Jameson's method for a fixed k.

The plots illustrate that if we were to discretize the convection-diffusion equation, where the diffusive coefficient $k = 10^{-2}$, we would get an unstable system if we were to use 6 grid cells on the abrupt grid or 11 on the exponential grid.

Similar to Section 4.6.3, we note in the following two figures, that if we had to discretize with a computer capable of maintaining only 20 grid cells, time integration would be unstable if we were to drop k to 10^{-3}.



(a) Methods applied on the abrupt grid



(b) Methods applied on the exponential grid

Figure 18: Eigenvalues of precise cell-centered method and Jameson's method a fixed grid cell number.

5.6 Conclusions in the Framework

Once again, the method with the worse local truncation error, Jameson's method is the discretization method worth recommending in the cell-centered finite volume framework. The local truncation error is even larger for Jameson's method than it was for Method A. However, the global results are accurate and the structure of the coefficient matrix has better properties than the matrix for the precise cell-centered method. If one were to compare also Method B against the precise cell-centered method, the latter should be preferred as it has a smaller global error. Nevertheless, the fact remains the same, the precise cell-centered method should be avoided because of the unstable time integration problem.

The following section explains why both Method A and Jameson's method inherit such good coefficient matrices and better numerical results. This will show why Method A and Jameson's method are the right ones to recommend.

6 Conservation of Physical Properties

Until now, the analysis has focused on the theoretical properties of the discretization methods, such as the convergence measured by the global error and the characteristics of the coefficient matrix. However, as we are modeling real-life quantities, one can also evaluate if our discretization methods behave as their continuous counterpart would.

For the compared methods: Method A and Method B, the only underlying difference between them was how the first order derivative was approximated. The same holds true for the finite volume methods, where only the construction of the convective flux differed. We therefore take a look at what only the first order derivative would describe, by introducing the partial differential equation modeling convection in time:

$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} = 0.$$

With any first order derivative discretization in space one would end up with the following linear system of ODEs, where L^{conv} denotes the coefficient matrix stemming from discretizing the first order derivative in space:

$$\begin{aligned} \frac{d\phi}{dt} + L^{\text{conv}}\phi &= 0, \\ \frac{d\phi}{dt} &= -L^{\text{conv}}\phi. \end{aligned} \tag{32}$$

One can talk about conserving certain physical properties when it comes to differential equations describing physical behavior. In [2, Section 1.5.4], the author defines the energy of the solution, which is equal to the weighted $L2$ norm of ϕ, incorporating the local grid cell size with the matrix $H = \text{diag}(h_- + h_+)$:

$$\|\phi\|_h^2 = \phi^* H \phi. \tag{33}$$

With H being a positive-definite matrix, this is a genuine vector norm.

Since convection only moves the quantity ϕ along the flow, without removing or adding more of it, the energy in a convection based system should remain constant. This implies that the time derivative of energy should be equal to 0:

$$\frac{d}{dt}\|\phi\|_h^2 = 0.$$

The following theorem will show that from our four discretization methods, this is only possible for Method A and Jameson's method.

Theorem 6.1 (Conservation of energy). *The solution of* (32) *conserves energy* (33) *if and only if the scaled coefficient matrix HL^{conv} is skew-symmetric.*

Proof. We derive the evolution of energy in time, by using the product rule for the derivative of energy and substituting the derived discretized system (32):

$$\begin{aligned}
\frac{d}{dt}\|\phi\|_h^2 &= \frac{d}{dt}(\phi^* H\phi) \\
&= \frac{d\phi^*}{dt}H\phi + \phi^* H\frac{d\phi}{dt} \\
&= -(L^{\text{conv}}\phi)^* H\phi - \phi H L^{\text{conv}}\phi \\
&= -\phi^*(HL^{\text{conv}} + (HL^{\text{conv}})^*)\phi.
\end{aligned}$$

For $\phi \neq 0$, the right hand side vanishes if and only if $HL^{\text{conv}} + (HL^{\text{conv}})^* = 0$. This occurs if and only if HL^{conv} is a skew-symmetric matrix.

□

One can note the following structure of the discretized convective operator (first derivative approximation) of Method A, L_A^{conv}:

$$\text{Method A:} \quad \phi_x \approx \frac{\phi_+ - \phi_-}{h_+ + h_-}$$

$$\Longrightarrow$$

$$HL_A^{\text{conv}} = H\begin{pmatrix} \ddots & & \\ & \frac{1}{h_+ + h_-} & \\ -\frac{1}{h_+ + h_-} & 0 & \frac{1}{h_+ + h_-} \\ & -\frac{1}{h_+ + h_-} & \\ & & \ddots \end{pmatrix} = \begin{pmatrix} \ddots & & \\ & 1 & \\ -1 & 0 & 1 \\ & -1 & \\ & & \ddots \end{pmatrix}.$$

The scaled coefficient matrix HL_A^{conv} is in fact skew-symmetric as the diagonal entries are all zero and the off-diagonal elements are the negatives of each other. Hence, the discretization method of Method A preserves analytical properties as it conserves energy.

The convective operator of Method B, L_B^{conv} however, does not have the same structure as the approximation includes a non-zero diagonal element:

$$\text{Method B:} \quad \phi_x \approx \frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)}$$

$$\Longrightarrow$$

$$HL_B^{\text{conv}} = H \begin{pmatrix} \ddots & & \\ & \frac{h_-}{h_+(h_++h_-)} & \\ -\frac{h_+}{h_-(h_++h_-)} & \frac{(h_+-h_-)}{h_+h_-} & \frac{h_-}{h_+(h_++h_-)} \\ & -\frac{h_+}{h_-(h_++h_-)} & \\ & & \ddots \end{pmatrix} = \begin{pmatrix} \ddots & & \\ & \frac{h_-}{h_+} & \\ -\frac{h_+}{h_-} & \frac{(h_+^2-h_-^2)}{h_+h_-} & \frac{h_-}{h_+} \\ & -\frac{h_+}{h_-} & \\ & & \ddots \end{pmatrix}$$

As generally $h_+ \neq h_-$ on non-uniform grids, the scaled coefficient matrix HL_B^{conv} will not be skew-symmetric. This implies that with approximating the first order derivative as in Method B, one would end up with an operator that does not follow physical properties. Discretizing convection with Method B does not keep true to the analytical properties of convection conserving energy.

A similar statement can be noted for the finite volume framework. Approximating the first order derivative with Jameson's method conserves energy yet discretizing with the the precise cell-centered method does not. This can be noted in the following approximations of the first order derivative of each method. Only the scaled coefficient matrix of Jameson's method will be skew-symmetric:

$$\text{Jameson's method:} \quad \phi_x \approx \frac{\phi_+ - \phi_-}{2H_0},$$
$$\text{Precise cell-centered method:} \quad \phi_x \approx \frac{H_0}{2h_+^C}(\phi_+ - \phi_0) + \frac{H_0}{2h_-^C}(\phi_0 - \phi_-).$$

In summary, in this section we established the possibility to conserve physical properties also in the structure of a discretization method. We have noted that convection is discretized with embedded energy conserving if and only if the scaled coefficient matrix HL^{conv} is skew-symmetric. This however only holds for the methods that on first glance seem worse because of their local truncation error - Method A and Jameson's method. This further exemplifies that both Method A and Jameson's method should be the preferred methods in their corresponding discretization frameworks. More relevant information about energy preserving discretized operators can be found in [9].

Remark. *Such an approach to conserve properties of analytical equations in the discrete operator is called mimetic discretization [2, Section 1.5.3]. Interestingly, when Antony Jameson constructed Jameson's method [8, Section 2], he had not noticed how such an approach would lead to a skew-symmetric discretized convective operator.*

7 Other Noteworthy Results

7.1 Random Grids

Till now we have only looked at two grid constructions- the abrupt and the exponential. Moreover, not always, when modeling real-life events, do we have the freedom of choice to pick underlying grid constructions. Therefore, to get a complete scientific outlook, in this section we generate discretization results using random grids. Understanding if discretization methods can work with any grid can give further reasons to recommend using a select method.

Moreover, the second order-convergence for both the finite difference and finite volume framework, have been proven on grids where the ratio between the smallest and largest grid cell is bounded. Such a property may no hold true on random grids. Therefore, it is important to test such grids.

We first show second-order convergence results for the random abrupt and the random exponential grid. In these, we still keep some underlying grid structures from before. We will look at the results both for the finite difference methods- Method A and Method B and for finite volume methods - precise cell-centered method and Jameson's method. We then limit-test our methods using a fully random grid only with boundary values staying intact.

We further give the formal definition of each random grid, followed by results. The results are generated using a select Python `np.random.seed`. Remarks concerning results about all randomly generated grids can be found later in the section 7.1.4.

7.1.1 Random Abrupt Grid

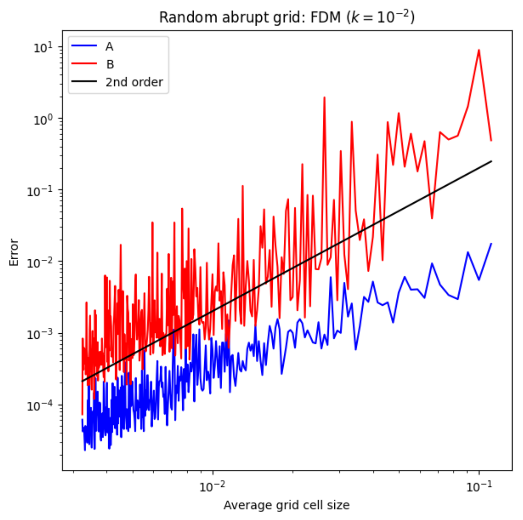
Just like with the uniform abrupt grid, half of the random generated grid cells will be placed in the interval from 0 to Tk and the other half in the boundary interval from Tk to 1. If the number of grid cells is odd, we again allocate an extra cell to the boundary layer. With this, the definition of the random abrupt grid becomes:

Definition (Random Abrupt Grid):

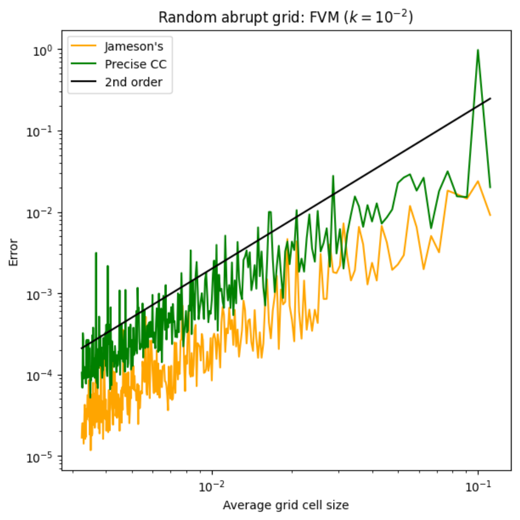
$$\begin{aligned} x_i &= Unif(0, Tk), \text{ for } i = 1, ..., M, \\ x_i &= Unif(Tk, 1), \text{ for } i = M+1, ..., N-1, \end{aligned}$$

$$\begin{gathered} \text{with } x_0 = 0, \quad x_N = 1, \\ \text{and } x_i < x_{i+1} \quad \text{for all } i. \end{gathered}$$

Results:



(a) Finite difference framework



(b) Finite volume framework

Figure 19: Second order convergence on the random abrupt grid.

7.1.2 Random Exponential Grid

We use the definition for the uniform exponential grid (6), however now σ_i gets produced from a random uniform distribution. Again enforcing half of the grid cells to lie in the boundary layer with $s = 2\ln[(1 - Tk)/Tk]$ we get the following:

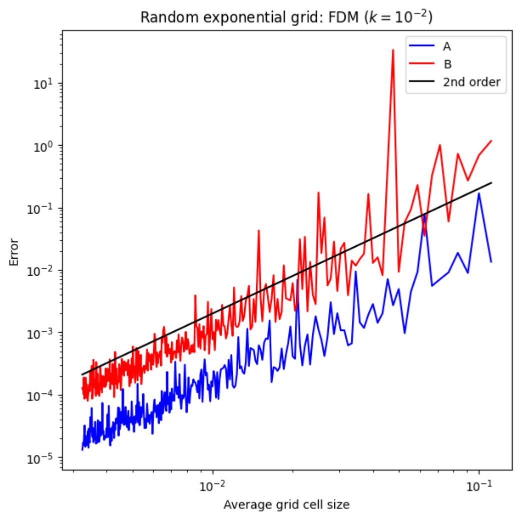
Definition (Random Exponential Grid):

$$x_i = \frac{1 - e^{-s\sigma_i}}{1 - e^{-s}}, \quad \text{where } \sigma_i = Unif(0,1), \quad \text{for } i = 1..., N-1,$$

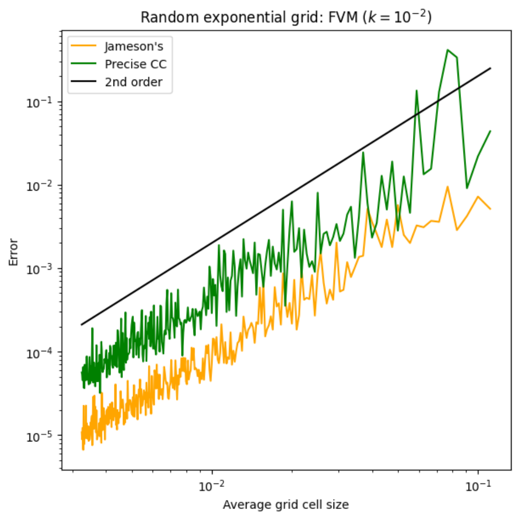
$$\text{with } \sigma_0 = 0, \sigma_N = 1,$$

$$\text{and } \sigma_i < \sigma_{i+1} \text{ for all } i.$$

Results:



(a) Finite difference framework



(b) Finite volume framework

Figure 20: Second order convergence on the random exponential grid.

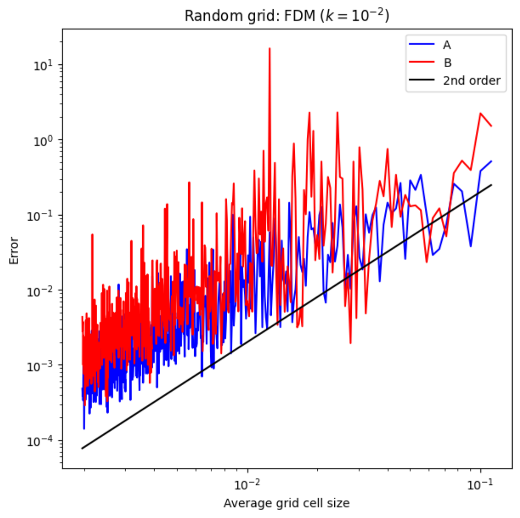
7.1.3 The Random Grid

Lastly, for limit-testing, we introduce the random grid. This will be a fully random grid with no boundary layer separation:

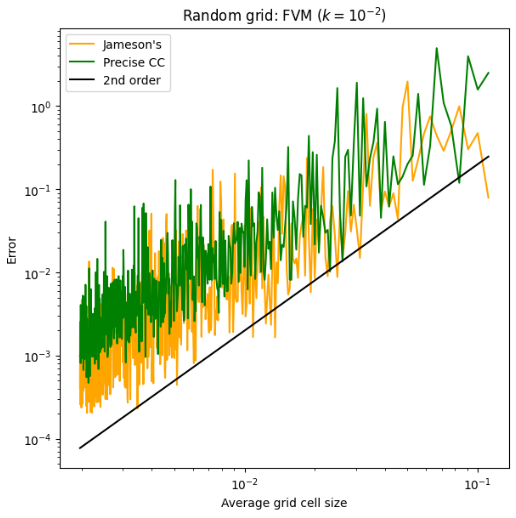
Definition (Random grid):

$$x_i = Unif(0,1), \text{ for } i = 1, ..., N-1,$$
$$\text{with } x_0 = 0, \quad x_N = 1,$$
$$\text{and } x_i < x_{i+1} \quad \text{for all } i.$$

Results:



(a) Finite difference framework



(b) Finite volume framework

Figure 21: Second order convergence on the random grid.

7.1.4 Remarks About Random Grids

In general, concerning the random abrupt and the random exponential grid, we see similar behavior as we saw when using non-random grids. We both see that Method A outperforms Method B and that the Jameson's method a gives better error than the precise cell-centered method. The wiggles of the error, when the grid gets finer, calm down, introducing again two lines parallel to the second-order line. As the grid gets coarser we again see more rapid increases in the error for both Method B and the precise cell-centered method.

One might notice that there are some select grid constructions on which the "worse" method either Method B or the precise cell-centered method performs better. Further research might be recommended on this, yet the fact remains the same - both Method A and Jameson's method preserve energy in their discretized convective operator. Moreover, when time is introduced, time integration for Method B and the precise cell-centered method can become unstable. Which continues to discourage the use of such methods.

For the fully random grid, we had to decrease for what average grid cell size we plot the results.

The grid cell number here runs from 10 grid cells to 500. We do this as it is harder to achieve the second-order convergence because the boundary layer is discretized with no extra care. As the grid gets finer and finer it starts to approach an extremely fine uniform grid for which we have the desired convergence results.

7.2 Finite Element Methods

For completeness also the finite element framework was looked at. Although, not to the same extent as the finite difference or finite volume methods.

We computed the discretized solution using the Galerkin approach, for which we took the standard form of a basis function:

$$\psi_i(x_j) = \delta_{ij} = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

The derivations in the finite element framework have been described in the Appendix B.3. Moreover, the code for the computed discretization method has been added in Appendix C. Turns out, the finite element method with such a basis function gives the same solution as Method A did in the finite difference framework. Further research of this phenomenon is recommended.

8 Conclusions

8.1 Summary

In this report, we have exemplified the need for coarse non-uniform grids as discussed in Section 3.1. Such grids play a large role in numerical mathematics, specifically computational fluid dynamics. In their applications it is common that functions both display smooth, constant intervals and boundary layers, where the function experiences a sharp, sudden increase. Non-uniform grids allow us to save computational time. By discretizing constant intervals with a few large cells and by dividing the boundary layers with many fine grid cells, we reduce the dimensions of the discretized linear system and at the same time maintain accuracy.

As the involvement of non-uniform grids makes method selection more advanced, we provided the convection-diffusion equation as a test case for various discretization methods in Section 2. Highlighting, how the convection-diffusion equation plays a role in computational fluid dynamics and why it serves as a good function for testing non-uniform grid theory.

As there are infinitely many ways one could define non-uniform grids, this paper, in Section 3.2 also introduces some specific grid layouts keeping in mind the underlying structure of discretizing the boundary layer with more care. We introduced the abrupt and the exponential grid, which give accurate results, provided the applied discretization method is well-constructed.

Both the finite difference and the finite volume framework, Section 4 and Section 5 respectively, were analyzed with care. We gave the underlying construction of all discretization methods and analyzed their respective local truncation errors. It was then exemplified by numerical results that the local truncation error is not a sufficient indicator of estimating the performance of discretization methods. In both frameworks, the method with a larger truncation error gave better global results. We included an explanation of the numerical results through a thorough analysis of the coefficient matrices.

Lastly, we investigated the physical properties of the partial differential equation modeling convection in Section 6. We proved that it is possible to maintain analytical properties of convection. The conservation of energy, an inherent characteristic of convection, can only be preserved in the discrete operator if the operator is skew-symmetric. Such a structure was embedded in Method A from the finite difference framework and Jameson's method from the finite volume framework.

8.2 Key findings

Comparison of Coarse and Fine Grids

Throughout this paper, we repeatedly exemplified that even though a discretization might work well on fine grids, it may give poor results coarse grids. Because of this, careful method selection is needed, as one might find methods that work well on both fine and coarse grids. There is no need to waste computational power, if one can discretize with a method that works with fewer grid cells. Such methods do in fact exist as was noted repeatedly.

The Misleading Nature of Local Truncation Error

The numerical results concerning the global error revealed that the local truncation error can be misleading when evaluating a discretization method. We saw that even though Method B has a lower local truncation error, it introduced a larger global error than Method A. Same can be noted in the finite volume framework. Mathematicians working in the field of computational mathematics should pay close attention to this. Spending resources on developing a worse discretization method because it displayed better first glance properties should not be the norm.

Discretization Methods Should Maintain Physical Properties

The superior performance of Method A and Jameson's method can be explained because they inherit physical properties. When one models real-life quantities, close attention should be paid to ensuring that the discretization methods also follow properties of analytical events. In this report this was exemplified by the superior methods of each respective framework, conserving energy in their discretized convective operator. Hence modeling convection properly as it indeed conserves energy.

This is the core finding of this report, as not only better global error is achieved with mimetic discretization but also one can place trust in the discretized results when employing them for practical applications.

8.3 Limitations and Further Research

The central limitation of this paper includes discretizing only the 1D convection-diffusion equation. Not only have we restricted the research to just one type of PDE, we have also further simplified it by setting the convective coefficient $u = 1$. Moreover, the notion of the start of the boundary layer should be better defined. For k values such as 10^{-2} or 10^{-3} the start of the boundary layer is described properly with $1 - 5k$, as stated by [4, Section 3.2], [1, Section 3]. However, this description becomes less accurate for smaller values of k.

Even if we have limited our research with only the convection-diffusion equation, there is still much more to explore that has not been covered in this report. One could discretize the convection-diffusion equation in more dimensions. Moreover, one might implement and compare discretization methods of higher order. For example, in [2, Section 1.7], a fourth-order discretization method has been introduced which also maintains energy conserving properties.

For further research, the study of discretizing the Navier-Stokes equations should also be noted. The close relationship between the convection-diffusion equation and the Navier-Stokes equations is undeniable. Both in some form, model convection and diffusion. Moreover, the distinct structure of constant and jump regions can be noted. This further illustrates why after studying how non-uniform grids behave on the convection-diffusion a logical next step is to implement non-uniform grids when discretizing the Navier-Stokes equations. As noted by [2, Section 2.3], the energy conserving quality embedded in skew-symmetric operators is also heavily used in Navier-Stokes discretization. Hence, even if the Navier-Stokes equations introduce non-linearity, underlying ideas about conserving physical properties should remain the same.

References

[1] A. E. P. Veldman and R. W. C. P. Verstappen, "Energy-preserving discretization on poorly-resolving grids," 2025, unpublished manuscript.

[2] A. E. P. Veldman, "Computational Fluid Dynamics," Lecture Notes, 2016.

[3] M. W. Dewhirst and T. W. Secomb, "Transport of drugs from blood vessels to tumour tissue," *Nature Reviews Cancer*, vol. 17, no. 12, pp. 738–750, Dec 2017.

[4] A. E. P. Veldman and K. Rinzema, "Playing with nonuniform grids," *Journal of Engineering Mathematics*, vol. 26, no. 1, pp. 119–130, 1992.

[5] E. K. de Rivas, "On the use of nonuniform grids in finite-difference equations," *Journal of Computational Physics*, vol. 10, no. 2, pp. 202–210, 1972.

[6] T. A. Manteuffel and A. B. White, Jr., "The numerical solution of second-order boundary value problems on nonuniform meshes," *Mathematics of Computation*, vol. 47, no. 176, pp. 511–535, 1986.

[7] H. J. Crowder and C. Dalton, "Errors in the use of nonuniform mesh systems," *Journal of Computational Physics*, vol. 7, no. 1, pp. 32–45, 1971.

[8] A. Jameson, W. Schmidt, and E. Turkel, "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes," in *14th Fluid and Plasma Dynamics Conference*, ser. AIAA Paper, no. 81-1259. Palo Alto, California: AIAA, 1981.

[9] A. E. P. Veldman, "Supraconservative Finite-Volume Methods for the Euler Equations of Subsonic Compressible Flow," *SIAM Review*, vol. 63, no. 4, pp. 756–779, 2021.

[10] ——, "Who's Afraid of Non-Symmetric Matrices? A Discussion of Elementary Iterative Methods," Delft University of Technology, Tech. Rep. TW1 88-49, 1988.

[11] A. S. Householder, *The Theory of Matrices in Numerical Analysis.* New York: Blaisdell Publishing Company, 1964.

A Needed Linear Algebra Theory

In this appendix section, we note definitions, theorems and lemmas relevant for linear algebra used in this report.

Definition A.1 (Skew-Symmetric Matrix). *A real square matrix A is said to be skew-symmetric if the following property holds:*

$$A^T = -A.$$

Such an expression can hold true if and only if the diagonal entries of A are 0 and

$$\forall i, j \quad a_{ij} = -aji.$$

Moreover, skew-symmetric matrices have purely imaginary eigenvalues.

Definition A.2 (Symmetric Matrix). *A real square matrix A is said to be symmetric if the following property holds:*

$$A^T = A.$$

Such an expression can hold true if and only if

$$\forall i, j \quad a_{ij} = aji.$$

Symmetric matrices have only real eigenvalues.

Definition A.3 (Weak Diagonal Dominance). *A matrix is defined as weakly diagonally dominant if the following inequality holds true for all rows of the matrix:*

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \forall i.$$

Lemma A.4 (Positive scaling). *Let A be a positive-stable matrix (i.e. all eigenvalues have a positive real part). Then for any positive definite matrix Q, also the matrix QA is positive-stable.*

Proof. See [10]. □

Theorem A.5 (Bendixson's Theorem). *Let $A = R + S$ be a general real matrix, where $R = \frac{1}{2}(A + A^T)$ is its symmetric part and $S = \frac{1}{2}(A - A^T)$ its skew-symmetric part. Let ρ_m and ρ_M be the smallest and largest eigenvalues of R, respectively; similarly σ_m and σ_M with respect to S. Then the eigenvalues of A are located in the rectangle defined by the lower-left and upper-right corners (ρ_m, σ_m) and (ρ_M, σ_M).*

Proof. Let x be a normed eigenvector (i.e. $x^*x = 1$) corresponding to an eigenvalue $\lambda = a + ib$ of A. Then

$$a + ib = \lambda = \lambda x^*x = x^*Ax = x^*Rx + ix^*Sx,$$

hence $a = x^*Rx$ and $b = x^*Sx$. Since both R and S are Hermitian matrices their eigenvalues are real and one has

$$\rho_m \leq x^*Rx \leq \rho_M;$$

a similar relation holds for S. This finishes the proof displayed in [11].

□

B Resulting Numerical Systems

To not over-saturate the report with mathematical expressions, the derivations of all the discretization methods are given here. The resulting linear equations describing the coefficient matrices and the right-hand side vector are then used in the code, Appendix C.

B.1 Finite Difference Methods

B.1.1 Method A

As we have the approximations for both ϕ_x and ϕ_{xx}, given in (11) and (14) respectively, we discretize the time-independent convection-diffusion equation in (2). We substitute the generated approximations into the convection-diffusion equation to get a solvable linear system for Method A:

$$\text{for } i = 2, ..., N-2 :$$
$$\frac{\phi_+ - \phi_-}{h_+ + h_-} - k\frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = 2, ..., N-2 :$$
$$\phi_- \left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)}\right) + \phi_0 \left(k\frac{1}{\frac{1}{2}h_+h_-}\right) +$$
$$+\phi_+ \left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)}\right) = 0$$

For the elements in the first and last rows of the coefficient matrix, this formula is also valid but it includes the boundary conditions. With $\phi(0) = \phi_0 = 0$ we have:

$$\text{for } i = 1 :$$
$$\phi_0 \left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_1 \left(k\frac{1}{\frac{1}{2}h_+h_-} \right) +$$
$$\phi_2 \left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right) = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = 1 :$$
$$\phi_1 \left(k\frac{1}{\frac{1}{2}h_+h_-} \right) + \phi_2 \left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right) = 0.$$

And from the right boundary $\phi(1) = \phi_N = 1$ we get contribution to the RHS:

$$\text{for } i = N - 1$$
$$\phi_{N-2} \left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_{N-1} \left(k\frac{1}{\frac{1}{2}h_+h_-} \right) +$$
$$+\phi_N \left(\frac{1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right) = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = N - 1$$
$$\phi_{N-2} \left(\frac{-1}{h_+ + h_-} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_{N-1} \left(k\frac{1}{\frac{1}{2}h_+h_-} \right) =$$
$$= k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} - \frac{1}{h_+ + h_-}$$

B.1.2 Method B

We get the following approximation for Method B:

$$\text{for } i = 2, ..., N - 2 :$$
$$\frac{h_-^2\phi_+ + (h_+^2 - h_-^2)\phi_0 - h_+^2\phi_-}{h_+h_-(h_+ + h_-)} - k\frac{h_-\phi_+ - (h_+ + h_-)\phi_0 + h_+\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = 2, ..., N - 2 :$$
$$\phi_- \left(\frac{-h_+}{h_-(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_-(h_+ + h_-)} \right) + \phi_0 \left(\frac{h_+ - h_-}{h_+h_-} + k\frac{1}{\frac{1}{2}h_+h_-} \right) +$$
$$+\phi_+ \left(\frac{h_-}{h_+(h_+ + h_-)} - k\frac{1}{\frac{1}{2}h_+(h_+ + h_-)} \right) = 0$$

Just as before, we also calculate the equations for the first and the last rows of the coefficients matrix:

$$\text{for } i = 1:$$
$$\phi_1 \left(\frac{h_+ - h_-}{h_+ h_-} + k \frac{1}{\frac{1}{2} h_+ h_-} \right) + \phi_2 \left(\frac{h_-}{h_+(h_+ + h_-)} - k \frac{1}{\frac{1}{2} h_+(h_+ + h_-)} \right) = 0$$

$$\text{for } i = N-1:$$
$$\phi_{N-2} \left(\frac{-h_+}{h_-(h_+ + h_-)} - k \frac{1}{\frac{1}{2} h_-(h_+ + h_-)} \right) + \phi_{N-1} \left(\frac{h_+ - h_-}{h_+ h_-} + k \frac{1}{\frac{1}{2} h_+ h_-} \right) =$$
$$= k \frac{1}{\frac{1}{2} h_+(h_+ + h_-)} - \frac{h_-}{h_+(h_+ + h_-)}$$

B.2 Finite Volume Methods

B.2.1 Vertex-Centered Method

After applying the conservation law (22), we get the following equations governing the coefficient matrix for the vertex-centered method without the terms containing boundary conditions:

$$\text{for } i = 2, ..., N-2:$$
$$\frac{1}{2}(\phi_+ - \phi_-) - k \left(\frac{\phi_+ - \phi_0}{h_+} - \frac{\phi_0 - \phi_-}{h_-} \right) = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = 2, ..., N-2:$$
$$\phi_- \left(-\frac{1}{2} - k \frac{1}{h_-} \right) + \phi_0 \left(k \left(\frac{1}{h_+} + \frac{1}{h_-} \right) \right) + \phi_+ \left(\frac{1}{2} - k \frac{1}{h_+} \right) = 0 \tag{34}$$

By having the cell-faces lie at the middle of grid cells everywhere, also for the boundaries this formula is valid:

$$\text{for } i = 1: \quad \phi_1 \left(k \left(\frac{1}{h_+} + \frac{1}{h_-} \right) \right) + \phi_2 \left(\frac{1}{2} - k \frac{1}{h_+} \right) = 0$$
$$\text{for } i = N-1: \quad \phi_{N-2} \left(-\frac{1}{2} - k \frac{1}{h_-} \right) + \phi_{N-1} \left(k \left(\frac{1}{h_+} + \frac{1}{h_-} \right) \right) = k \frac{1}{h_+} - \frac{1}{2}$$

B.2.2 Precise Cell-Centered Method

The equations governing the coefficient matrix for the precise cell-centered method are the following:

$$\text{for } i = 2, ..., N-1:$$
$$\frac{H_0}{2h_+^C}(\phi_+ - \phi_0) - k\frac{\phi_+ - \phi_0}{h_+^C} + \frac{H_0}{2h_-^C}(\phi_0 - \phi_-) + k\frac{\phi_0 - \phi_-}{h_-^C} = 0$$
$$\iff$$
$$\text{for } i = 2, ..., N-1:$$
$$\phi_-\left(-\frac{H_0}{2h_-^C} - k\frac{1}{h_-^C}\right) + \phi_0\left(\frac{H_0}{2h_-^C} - \frac{H_0}{2h_+^C} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_+\left(\frac{H_0}{2h_+^C} - k\frac{1}{h_+^C}\right) = 0 \tag{35}$$

Note that, for the boundaries, we will have that the cell sizes match half of the length of the cell-volumes. Therefore:

$$\text{for } i = 1: \quad \frac{1}{2}H_0 = h_-^C \tag{36}$$
and
$$\text{for } i = N: \quad \frac{1}{2}H_0 = h_+^C \tag{37}$$

We can use the same expression (35) for the boundaries, however here we use (36) for the left boundary:

$$\text{for } i = 1:$$
$$\phi_1\left(\frac{H_0}{2h_-^C} - \frac{H_0}{2h_+^C} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_2\left(\frac{H_0}{2h_+^C} - k\frac{1}{h_+^C}\right) = 0$$
$$\iff$$
$$\text{for } i = 1:$$
$$\phi_1\left(1 - \frac{H_0}{2h_+^C} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_2\left(\frac{H_0}{2h_+^C} - k\frac{1}{h_+^C}\right) = 0$$

For the right boundary we use (37):

$$\text{for } i = N:$$
$$\phi_{N-1}\left(-\frac{H_0}{2h_-^C} - k\frac{1}{h_-^C}\right) + \phi_N\left(\frac{H_0}{2h_-^C} - \frac{H_0}{2h_+^C} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) = k\frac{1}{h_+^C} - \frac{H_0}{2h_+^C}$$
$$\iff$$
$$\text{for } i = N:$$
$$\phi_{N-1}\left(-\frac{H_0}{2h_-^C} - k\frac{1}{h_-^C}\right) + \phi_N\left(\frac{H_0}{2h_-^C} - 1 + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) = k\frac{1}{h_+^C} - 1$$

B.2.3 Jameson's Method

The construction of Jameson's method follows the vertex-centered method. Using the vertex-centered derivations (34) with the cell-centered grid definitions (24), and with the fact that the cell-centered methods solve for one extra variable, we get the following:

$$\text{for } i = 2, ..., N-1:$$
$$\phi_-\left(-\frac{1}{2} - k\frac{1}{h_-^C}\right) + \phi_0\left(k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_+\left(\frac{1}{2} - k\frac{1}{h_+^C}\right) = 0,$$

Here, however, the definitions for the first and the last row of the coefficient matrix differ as we have the following. For the first grid point, as the left cell-face lies on the left boundary, the flux there will be given by the formula:

$$\text{for } i = 1: q_{(1)-\frac{1}{2}} = \phi_0 - k\frac{\phi_1 - \phi_0}{h_-^C}.$$

The flux at the right cell-face of the first grid point does not differ from before, therefore with $\phi_0 = 0$ and the conservation law (22), we get the following formula for the first row of the coefficient matrix:

$$\text{for } i = 1:$$
$$\frac{\phi_1 + \phi_2}{2} - k\frac{\phi_2 - \phi_1}{h_+^C} - \phi_0 + k\frac{\phi_1 - \phi_0}{h_-^C} = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = 1:$$
$$\phi_1\left(\frac{1}{2} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) + \phi_2\left(\frac{1}{2} - k\frac{1}{h_+^C}\right) = 0$$

Similarly, for the last grid point, we will have that the right cell-face differs. With $\phi_{N+1} = 1$, we will have that:

$$\text{for } i = N:$$
$$q_{(N)+\frac{1}{2}} = \phi_{N+1} - k\frac{\phi_{N+1} - \phi_N}{h_+^C}$$
$$\Longrightarrow$$
$$\text{for } i = N:$$
$$\phi_{N+1} - k\frac{\phi_{N+1} - \phi_N}{h_+^C} - \frac{\phi_N + \phi_{N-1}}{2} + k\frac{\phi_N - \phi_{N-1}}{h_-^C} = 0$$
$$\Longleftrightarrow$$
$$\text{for } i = N:$$
$$\phi_{N-1}\left(-\frac{1}{2} - k\frac{1}{h_-^C}\right) + \phi_N\left(-\frac{1}{2} + k(\frac{1}{h_+^C} + \frac{1}{h_-^C})\right) = k\frac{1}{h_+^C} - 1$$

B.3 Finite Element Methods

One can reduce our boundary value problem in (2) to have two zero boundary conditions:

take $\phi(x) = \bar{\phi}(x) + \tilde{\phi}(x)$,
choose a linear function $\bar{\phi}(x) = x$ s.t. $\bar{\phi}(0) = 0$ and $\bar{\phi}(1) = 1$,
$\tilde{\phi}(x)$ unknown function with $\tilde{\phi}(0) = 0$ and $\tilde{\phi}(1) = 0$.

This reduces the problem we solve for:

$$\begin{gathered}\phi_x - k\phi_{xx} = 0 \\ \Longrightarrow \\ \frac{d(x+\tilde{\phi}(x))}{dx} - k\frac{d^2(x+\tilde{\phi}(x))}{dx^2} = 0 \\ \Longrightarrow \\ \tilde{\phi}_x - k\tilde{\phi}_{xx} = -1,\end{gathered}$$

with boundary conditions $\tilde{\phi}(0) = 0$ and $\tilde{\phi}(1) = 0$. After finding $\tilde{\phi}$ we get the resulting solution by

$$\phi(x) = \tilde{\phi}(x) + \bar{\phi}(x) = \tilde{\phi}(x) + x \tag{38}$$

We solve the problem

$$\begin{gathered}\tilde{\phi}_x - k\tilde{\phi}_{xx} = -1 \quad \text{on } \mathcal{W}, \\ \text{where } \mathcal{W} = \{\tilde{\phi} \in \mathcal{C}^2[0,1] | \quad \tilde{\phi}(0) = 0,\, \tilde{\phi}(1) = 0\}, \quad \mathcal{V} = \mathcal{W}.\end{gathered}$$

Galerkin approach then states:

$$\begin{gathered}\text{Find } \tilde{\phi}_v \text{ s.t. } (r(\tilde{\phi}_v), v) = 0 \quad \forall v \in \mathcal{V}, \\ \text{where } r(\tilde{\phi}_v) = 1 + \frac{d\tilde{\phi}_v}{dx} - k\frac{d^2\tilde{\phi}_v}{dx^2}.\end{gathered}$$

This results in the following:

$$\begin{gathered}(r(\tilde{\phi}_v), v) = 0 \\ \Longrightarrow \\ \int_0^1 v(1 + \frac{d\tilde{\phi}_v}{dx} - k\frac{d^2\tilde{\phi}_v}{dx^2})dx = 0.\end{gathered}$$

Such an expression then gives the following linear and bi-linear form:

$$F(v) = -\int_0^1 v dx, \tag{39}$$

$$a(v, \tilde{\phi}_v) = \int_0^1 v \frac{d\tilde{\phi}_v}{dx} dx + k \int_0^1 \frac{dv}{dx} \frac{d\tilde{\phi}_v}{dx} dx \tag{40}$$

With our chosen basis function

$$\psi_i(x_j) = \delta_{ij} = \begin{cases} 0, & i \neq j, \\ 1, & i = j, \end{cases}$$

we construct the mass and the stiffness matrix:

$$A_{ij} = a(\psi_i, \psi_j), \quad b_i = F(\psi_i).$$

With the linear and bi-linear form definitions given in 39 and 40 respectively, one gets the following discretized equation describing the coefficient matrix and the right hand side:

$$\tilde{\phi}_- \left(-k\frac{1}{h_-} - \frac{1}{2} \right) + \tilde{\phi}_0 \left(k(\frac{1}{h_-} + \frac{1}{h_+}) \right) + \tilde{\phi}_+ \left(-k\frac{1}{h_+} + \frac{1}{2} \right) = -\frac{h_- + h_+}{2}$$

The discretized solution $\tilde{\phi}$ then gives the solution with $\phi(x) = \tilde{\phi} + x$ as noted in (38). Such a solution fully matches the solution of Method A, as stated in Section 7.2.

C Code

The numerical results where generated using the following code definitions. For coding implications the programming language Pyhton was used. In this appendix one can find both the code definitions of grids and methods. Moreover, other definitions that ease programming exercises, like the function to compute the error of the discretization or the analytical solution are included.

C.1 Arbitrary definitions

```
# Analyitical solution


def analytical_sol(k):
    phi = lambda x: (np.exp(x/k - 1/k)-np.exp(-1/k)) / (1 - np.exp(-1/k))
    return phi



# Calculating grid cell sizes h
```

```
def grid_dif(grid):
    K = len(grid) - 1
    h_array = np.zeros(K)
    for l in range(K):
        h_array[l] = grid[l + 1] - grid[l]

    return h_array


# Check if the grid is incorrecly structured

def grid_check(grid):

    check_value = grid[0]
    for i in range(len(grid) - 1):
        if grid[i+1] <= check_value:
            break
        else:
            check_value = grid[i+1]

    if i != len(grid) - 2:
        print("Incorrect grid")

    if grid[-1] != 1 or grid[0] != 0:
        print("Incorrect endpoints")


# Check if the grid is coarse

def T_check(T, k):
    if T*k > 0.5:
        print("Not a coarse grid")
        print(f"T = {T}, k = {k}")


# Calculate error

def calculate_error(grid, sol, approx):
    e = approx - sol(grid)
    error = np.sqrt(np.trapezoid(e ** 2, grid))

    return error


# List of methods

def method_list():
    methods = ["A", "B", "FVM_CC", "Jameson", "FEM"]

    return methods


# List of grids

def grid_list():
    grids = ["abrupt", "exp", "random abrupt", "random exp", "random"]

    return grids


# Constructing grid

def choose_grid(k, grid_points, grid_type, T=5):
    if grid_type == "abrupt":
        grid = grid_abrupt(grid_points, T, k)
    elif grid_type == "exp":
        grid = grid_exp(grid_points, T, k)
    elif grid_type == "random abrupt":
        grid = grid_rand_abrupt(grid_points, T, k)
    elif grid_type == "random exp":
        grid = grid_rand_exp(grid_points, T, k)
    elif grid_type == "random":
        grid = grid_random(grid_points)
    else:
        print("Incorrect grid name")

    return grid
```

```
# Getting the underlying grid, approximation and error

def grid_method_error(k, grid_points, grid_type, method_type, T=5):

    if grid_type == "abrupt":
        grid = grid_abrupt(grid_points, T, k)
    elif grid_type == "exp":
        grid = grid_exp(grid_points, T, k)
    elif grid_type == "random abrupt":
        grid = grid_rand_abrupt(grid_points, T, k)
    elif grid_type == "random exp":
        grid = grid_rand_exp(grid_points, T, k)
    elif grid_type == "random":
        grid = grid_random(grid_points)
    else:
        print("Incorrect grid name")

    if method_type == "A":
        method = method_A(grid, k)
    elif method_type == "B":
        method = method_B(grid, k)
    elif method_type == "Precise CC":
        method, grid = method_FVM_CC(grid, k)
    elif method_type == "Jameson's":
        method, grid = method_jameson(grid, k)
    elif method_type == "FEM":
        method = method_FEM(grid, k)
    else:
        print("Incorrect method name")

    sol = analytical_sol(k)

    error = calculate_error(grid, sol, method)

    return grid, method, error
```

C.2 Grids

```
def grid_abrupt(nr_of_points, T, k):
    T_check(T, k)
    node_array = np.zeros(nr_of_points)

    half_1 = int((nr_of_points - 1)/2)
    half_2 = nr_of_points - 1 - half_1

    for i in range(half_1):
        node_array[i] = i * (1 - T * k) / half_1
    for j in range(half_2 + 1):
        node_array[- j - 1] = 1 - (j * T * k) / half_2

    grid_check(node_array)

    return node_array


def grid_rand_abrupt(nr_of_points, T, k):
    T_check(T, k)

    boundary_threshold = 1 - T * k

    half_points = int((nr_of_points - 2) / 2)
    node_array = np.zeros(1)

    a1 = 0
    b1 = boundary_threshold
    a2 = boundary_threshold
    b2 = 1

    rand_array_1 = (b1 - a1) * np.random.random_sample((1, half_points)) + a1
    rand_array_1 = np.sort(rand_array_1)
    node_array = np.append(node_array, rand_array_1)

    rand_array_2 = (b2 - a2) * np.random.random_sample((1, half_points + 1)) \
    + a2
    rand_array_2 = np.sort(rand_array_2)
    node_array = np.append(node_array, rand_array_2)

    node_array = np.append(node_array, 1)

    return node_array
```

```
def grid_rand_exp(nr_of_points, T, k):
    T_check(T, k)

    s = 2 * np.log((1-T*k)/(T*k))

    sigma = np.random.uniform(0, 1, size=(1, nr_of_points - 2))
    sigma = np.sort(sigma)

    sigma = np.append(0, sigma)
    sigma = np.append(sigma, 1)

    node_array = np.zeros(len(sigma))

    for l in range(len(sigma)):
        node_array[l] = (1 - np.exp(-sigma[l] * s)) / (1 - np.exp(-s))

    return node_array


def grid_exp(nr_of_points, T, k):
    T_check(T, k)

    s = 2 * np.log((1-T*k)/(T*k))

    sigma = np.linspace(0, 1, nr_of_points)

    node_array = np.zeros(len(sigma))

    for l in range(len(sigma)):
        node_array[l] = (1 - np.exp(-sigma[l] * s)) / (1 - np.exp(-s))

    return node_array


def grid_random(nr_of_points):
    node_array = np.random.random_sample((1, nr_of_points-2))
    node_array = np.sort(node_array)
    node_array = np.append(0, node_array)
    node_array = np.append(node_array, 1)

    return node_array
```

C.3 Methods

```
def method_A(grid, k):

    h_array = grid_dif(grid)

    N = len(grid) - 2

    lower_diag = np.zeros(N - 1)
    upper_diag = np.zeros(N - 1)
    main_diag = np.zeros(N)

    # Lower diag
    for j in range(N - 1):
        h_p = h_array[j+2]
        h_m = h_array[j+1]

        sum_h = h_p + h_m
        prod_h = h_p * h_m

        lower_diag[j] = (- 1) / sum_h - k * (h_p / (0.5 * prod_h * sum_h))

    # Upper_diag
    for m in range(N - 1):
        h_p = h_array[m+1]
        h_m = h_array[m]

        sum_h = h_p + h_m
        prod_h = h_p * h_m

        upper_diag[m] = 1 / sum_h - k * (h_m / (0.5 * prod_h * sum_h))

    # Main diag
    for p in range(N):
        h_p = h_array[p + 1]
        h_m = h_array[p]
        prod_h = h_p * h_m

        main_diag[p] = k / (0.5 * prod_h)

    A = np.diag(main_diag) + np.diag(upper_diag, k=1) \
    + np.diag(lower_diag, k=-1)

    # RHS
    b = np.zeros(N)
    sum_h = h_array[-1] + h_array[-2]
    prod_h = h_array[-1] * h_array[-2]
    h_m = h_array[-2]

    b[-1] = - 1 / (sum_h) + (k * h_m) / (0.5 * prod_h * sum_h)

    approx = np.linalg.solve(A, b)

    approx = np.append(approx, 1)
    approx = np.append(0, approx)

    return approx


def method_B(grid, k):
    h_array = grid_dif(grid)

    N = len(grid) - 2

    lower_diag = np.zeros(N - 1)
    upper_diag = np.zeros(N - 1)
    main_diag = np.zeros(N)

    # Lower diag
    for j in range(N - 1):
        h_p = h_array[j + 2]
        h_m = h_array[j + 1]

        sum_h = h_p + h_m
        prod_h = h_p * h_m
        lower_diag[j] = (- h_p ** 2) / (prod_h * sum_h) \
        - k * (h_p / (0.5 * prod_h * sum_h))

    # Upper_diag
    for m in range(N - 1):
        h_p = h_array[m+1]
        h_m = h_array[m]

        sum_h = h_p + h_m
        prod_h = h_p * h_m

        upper_diag[m] = (h_m ** 2) / (prod_h * sum_h) \
        - k * (h_m / (0.5 * prod_h * sum_h))

    # Main diag
    for p in range(N):
        h_p = h_array[p + 1]
        h_m = h_array[p]

        sum_h = h_p + h_m
        prod_h = h_p * h_m

        main_diag[p] = (h_p**2 - h_m**2)/ (prod_h * sum_h) \
        + k / (0.5 * prod_h)

    A = np.diag(main_diag) + np.diag(upper_diag, k=1) \
    + np.diag(lower_diag, k=-1)

    # RHS
    b = np.zeros(N)
    sum_h = h_array[-1] + h_array[-2]
    prod_h = h_array[-1] * h_array[-2]
    h_m = h_array[-2]

    b[-1] = - (h_m ** 2) / (prod_h * sum_h) \
    + (k * h_m) / (0.5 * prod_h * sum_h)

    approx = np.linalg.solve(A, b)
```

```
    approx = np.append(approx, 1)
    approx = np.append(0, approx)

    return approx


def method_FVM_CC(grid, k):
    face_grid = grid
    H_array = grid_dif(face_grid)

    grid = np.zeros(len(grid) - 1)

    for i in range(len(grid)):
        grid[i] = 0.5 * (face_grid[i] + face_grid[i+1])

    grid = np.append(grid, 1)
    grid = np.append(0, grid)

    h_array = grid_dif(grid)

    N = len(grid) - 2

    lower_diag = np.zeros(N - 1)
    upper_diag = np.zeros(N - 1)
    main_diag = np.zeros(N)

    # Lower diag
    for j in range(N - 1):
        h_m = h_array[j + 1]
        H = H_array[j + 1]

        lower_diag[j] = - H / (2 * h_m) - k / h_m

    # Upper diag
    for l in range(N - 1):
        h_p = h_array[l + 1]
        H = H_array[l]

        upper_diag[l] = H / (2 * h_p) - k / h_p
    # Main diag
    for m in range(N):
        h_p = h_array[m + 1]
        h_m = h_array[m]
        H = H_array[m]

        main_diag[m] = H / (2 * h_m) - H / (2 * h_p) + k / h_p + k / h_m

    A = np.diag(main_diag) + np.diag(upper_diag, k=1) \
    + np.diag(lower_diag, k=-1)

    # RHS
    b = np.zeros(N)
    h_p = h_array[-1]
    H = H_array[-1]
    b[-1] = k / h_p - H / (2 * h_p)

    approx = np.linalg.solve(A, b)

    approx = np.append(approx, 1)
    approx = np.append(0, approx)

    return approx, grid


def method_jameson(grid, k):
    face_grid = grid

    grid = np.zeros(len(grid) - 1)

    for i in range(len(grid)):
        grid[i] = 0.5 * (face_grid[i] + face_grid[i+1])

    grid = np.append(grid, 1)
    grid = np.append(0, grid)

    h_array = grid_dif(grid)

    N = len(grid) - 2
```

```
lower_diag = np.zeros(N - 1)
upper_diag = np.zeros(N - 1)
main_diag = np.zeros(N)

# Lower diag
for j in range(N - 1):
    h_m = h_array[j+1]

    lower_diag[j] = - 0.5 - k / h_m

# Upper_diag
for m in range(N - 1):
    h_p = h_array[m+1]

    upper_diag[m] = 0.5 - k / h_p

# Main diag
for p in range(N):
    h_p = h_array[p + 1]
    h_m = h_array[p]


    main_diag[p] = k * (1 / h_p + 1/ h_m)

    if p == 0:
        main_diag[0] += 0.5
    elif p == N-1:
        main_diag[-1] -= 0.5



A = np.diag(main_diag) + np.diag(upper_diag, k=1) \
+ np.diag(lower_diag, k=-1)

# RHS
b = np.zeros(N)
h_p = h_array[-1]

b[-1] = -1 + k / h_p

approx = np.linalg.solve(A, b)
```

```
    approx = np.append(approx, 1)
    approx = np.append(0, approx)

    return approx, grid, face_grid



def method_FEM(grid, k):
    # Calculate cell sizes
    h_array = grid_dif(grid)

    N = len(grid) - 2

    lower_diag = np.zeros(N - 1)
    upper_diag = np.zeros(N - 1)
    main_diag = np.zeros(N)

    # Main diag
    for i in range(N):
        h_m = h_array[i]
        h_p = h_array[i+1]

        main_diag[i] = k * (1/ h_p + 1/h_m)

    # Lower diag
    for j in range(N-1):
        h_p = h_array[j+1]

        lower_diag[j] = -k / h_p - 0.5

    # Upper diag
    for m in range(N-1):
        h_p = h_array[m+1]

        upper_diag[m] = -k / h_p + 0.5

    A = np.diag(main_diag) + np.diag(upper_diag, k=1) \
    + np.diag(lower_diag, k=-1)

    # RHS
```

```
b = np.zeros(N)

for l in range(N):
    h_m = h_array[l]
    h_p = h_array[l+1]

    b[l] = - (h_p + h_m) / 2

approx = np.linalg.solve(A, b)
# u(x) = \tilde{u} + x
approx = approx + grid[1:N+1]

approx = np.append(approx, 1)
approx = np.append(0, approx)

return approx
```