Stability of BDF-ADI Discretizations

Thesis by

João Miguel Felício dos Reis

In Partial Fulfillment of the Requirements

For the Degree of

Masters of Science

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

August, 2017
The thesis of João Miguel Felício dos Reis is approved by the examination committee

Committee Chairperson: Professor David Ketcheson
Committee Co-Chair: Professor Lajos Lóczy
Committee Members: Professor David Keyes, Professor Dominik Michels
ABSTRACT

We present new results on absolute stability for BDF-ADI (Backward differentiation formula Alternating Direction Implicit) methods applied to a linear advection and diffusion equations. Unconditional absolute stability of the BDF2-ADI method is proven for advection and diffusion separately, as well as to the BDF3-ADI method for the purely-diffusive case. Conditional absolute stability of the BDF4-ADI is also proven for the purely-diffusive case, and stability regions for BDF3-ADI and BDF4-ADI are given in terms of the PDE coefficients and numerical parameters. Lastly, numerical experiments are presented to support the theoretical results and conjectures. These experiments also suggest future work.
ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisors Prof. David Ketcheson and Prof. Lajos Lóczy for the continuous support on this project, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. Besides my advisers, I would like to thank the rest of my thesis committee: Prof. David Keyes and Prof. Dominik Michels for their insightful comments. I would not have known KAUST and the many opportunities that this university has if not for Prof. Diogo Gomes. I would like to express my sincere appreciation for showing me a new window of possibilities. I thank my dear colleague Yiannis Hadjimichael for the stimulating discussions, and also for answering without hesitation my numerous questions for the last 20 months. Also, I thank my friends within the many departments of KAUST for the precious support over the many days of helpless research. In particular, I am grateful to Francisco Berkemeier and Mariana Prazeres for all the fun and loyalty. Last but not the least, I would like to thank my family: my parents and sister for supporting me spiritually throughout writing this thesis and my life in general.
# TABLE OF CONTENTS

Examination Committee Page 2

Copyright 3

Abstract 4

Acknowledgements 5

List of Figures 8

List of Tables 10

1 Introduction 11

2 Stability of Numerical Methods 17
   2.1 Schur-Cohn polynomial reduction ................................. 20

3 BDF-ADI Spectral Collocation Method 24
   3.1 Spectral Collocation Methods ........................................ 26
      3.1.1 Fourier analysis in one dimension .............................. 27
      3.1.2 Fourier analysis in two dimensions ............................ 30
      3.1.3 A spatial discretization ....................................... 31
      3.1.4 Spectrum of a spectral differentiation matrix based on Fourier collocation ........................................................... 33
   3.2 BDF Methods ............................................................ 34
      3.2.1 Backward Differentiation Formula .............................. 35
      3.2.2 Stability Region .................................................. 37
   3.3 BDF-ADI Spectral Collocation Methods ............................... 40

4 Main Results 43
   4.1 Stability for BDF2-ADI ............................................... 44
      4.1.1 Absolute stability of BDF2-ADI for the parabolic case with mixed term ............................................................... 45
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Regions of stability (in the complex-valued plane) of BDF methods for different orders.</td>
</tr>
<tr>
<td>3.2</td>
<td>The real-valued stability regions of BDF methods for the hyperbolic equation. This is a two-dimensional representation of the imaginary axis of regions in Figure 3.1. The BDF1 and BDF2 methods are unconditionally stable.</td>
</tr>
<tr>
<td>4.1</td>
<td>Stability region for ((A_1, B_1)) of BDF3-ADI for the hyperbolic problem, together with the lattice of scaled eigenvalues.</td>
</tr>
<tr>
<td>4.2</td>
<td>Illustration of conditional stability of BDF4-ADI for the pure parabolic problem.</td>
</tr>
<tr>
<td>4.3</td>
<td>Conditional stability of BDF4-ADI for the pure parabolic problem.</td>
</tr>
<tr>
<td>4.4</td>
<td>Stability region for ((A_2, B_2, F)) of BDF3-ADI for the problem (4.2). This region is symmetric with respect to the plane (A_2 = B_2). However, it is not symmetric with respect to (F = 0).</td>
</tr>
<tr>
<td>4.5</td>
<td>Lattice for the parabolic problem. This lattice corresponds to the scaled eigenvalues (A_2, B_2) and (F) with (\alpha_2 = \beta_2 = \gamma = 1) and (\Delta t = 0.001). Each direction has 64 collocation points.</td>
</tr>
<tr>
<td>4.6</td>
<td>This is a good indication that this choice of parameters yields a stable solution for (\Delta t \leq 0.001).</td>
</tr>
<tr>
<td>4.7</td>
<td>Slices of the stability region of BDF3-ADI for the parabolic problem (4.13). This scheme yields an unstable solution for (\Delta t = 0.002).</td>
</tr>
<tr>
<td>4.8</td>
<td>Stability region for ((A_2, B_2, F)) of BDF4-ADI for problem (4.2).</td>
</tr>
<tr>
<td>4.9</td>
<td>Stable solution of the BDF4-ADI applied to the parabolic problem (4.14).</td>
</tr>
<tr>
<td>4.10</td>
<td>The solution of BDF4-ADI for (4.14) becomes unstable if instead we consider (\Delta t = 0.0012).</td>
</tr>
<tr>
<td>5.1</td>
<td>Convergence test of BDF2-ADI applied to problem (5.1).</td>
</tr>
<tr>
<td>5.2</td>
<td>Numerical solution of BDF3-ADI for problem (5.2).</td>
</tr>
<tr>
<td>5.3</td>
<td>Convergence test of BDF3-ADI for the problem (5.2). The final time is (T = 0.5).</td>
</tr>
</tbody>
</table>
5.4 Stable numerical solution of BDF4-ADI for problem (5.2) ........................................ 63
5.5 Convergence test of BDF4-ADI for the problem (5.2). The final time is $T = 0.5$ .................. 64
5.6 Unstable numerical solution of BDF4-ADI for problem (5.2). The numerical solution at $T = 1.5$ corresponding to $\Delta t = 0.001$ has an error of $10^{-13}$ .................................................. 64
5.7 Stable numerical solution of BDF4-ADI for the parabolic problem (4.12) ....................... 65
5.8 Convergence test of BDF3-ADI for the parabolic problem (4.12). The final time is $T = 0.5$ .................. 65
5.9 Unstable numerical solution of BDF3-ADI for the parabolic problem (4.13) ................. 66
5.10 Numerical solutions for problem (4.14). A slight change of the time step can change the behaviour of the numerical solution drastically. The numerical solution is stable for $\Delta t = 0.001$ and unstable for $\Delta t = 0.0012$. The numerical solution at $T = 3$ corresponding to $\Delta t = 0.001$ has an error of $10^{-16}$ .................. 67
5.11 Convergence of BDF4-ADI for the parabolic equation with the mixed term. This example refers to (4.14). For $\Delta t < 10^{-4}$ the floating point errors originate instabilities .................. 67
6.1 We can conjecture conditional stability of BDF3-ADI and BDF4-ADI for the mixed problem .................. 69
LIST OF TABLES

3.1 Coefficients for BDF methods of order $s$, with $s = 1, \ldots, 6$ \hspace{1cm} 35
Chapter 1

Introduction

A finite difference method is one technique among many to find a numerical solution for a partial differential equation (PDE). In this thesis, we will study a particular finite difference method, the so-called backward difference formula - alternating direction implicit method (BDF-ADI method).

In order to find a numerical solution using a finite difference method, a finite difference equation (FDE) must be solved. This FDE is generated by discretizing the PDE in space and in time. As an example, consider the heat equation \( u_t = u_{xx} + u_{yy} \) and the FDE given by applying a Backward Euler method in time and a second order centred approximation is space,

\[
\frac{U^{n+1} - U^n}{\Delta t} = D_{xx} U^{n+1} + D_{yy} U^{n+1}.
\]

To solve the previous implicit FDE, we need to compute the matrix \( (I - \Delta t D_{xx}^2 - \Delta t D_{yy}^2)^{-1} \). Although each tridiagonal matrix \( (I - \Delta t D_{xx}^2) \) and \( (I - \Delta t D_{yy}^2) \) can be inverted easily, \( (I - \Delta t D_{xx}^2 - \Delta t D_{yy}^2)^{-1} \) does not enjoy the same properties. Therefore, we would like to solve the previous FDE using the easily computable matrices \( (I - \Delta t D_{xx}^2)^{-1} \) and \( (I - \Delta t D_{yy}^2)^{-1} \), instead of \( (I - \Delta t D_{xx}^2 - \Delta t D_{yy}^2)^{-1} \). This is where splitting methods come into play.

Let \( D \) denote a coefficient matrix for a general spatial discretization. The goal of splitting methods is to split \( D \) into other coefficient matrices \( D_1, \ldots, D_r \), each with a simpler form than the original. Going back to the previous example, a splitting
method splits $D = D_{xx}^2 + D_{yy}^2$ into two coefficient matrices. This yields a system of FDEs in the form,

$$
\begin{align*}
\frac{U^* - U^n}{\Delta t} &= D_{xx}^2 U^* + D_{yy}^2 U^n \\
\frac{U^{n+1} - U^*}{\Delta t} &= D_{xx}^2 U^{n+1} + D_{yy}^2 U^*.
\end{align*}
$$

where $D_{xx}$ and $D_{yy}$ are spatial discretization operators for the second-order derivatives in each dimension separately. The numerical solution $U^*$ is an intermediate step between the solutions at time $t_n$ and $t_{n+1}$, having no physical meaning. Although this split FDE is not equivalent to the original FDE, the solutions to both FDE enjoy the same convergence properties. Whereas the original FDE is implicit in both $x$ and $y$ directions, each equation from the split FDE is implicit in just one direction. This splitting makes the split FDE easier to solve than the original FDE. For discretizations such as $D_{xx}$ and $D_{yy}$, both the memory and computational complexity are linear in the number of grid points in each dimension, whereas the complexities of the unsplit inversion grow superlinearly if a direct linear algebraic method is chosen.

The, alternating direction implicit methods (ADI methods) split each coefficient matrix into the same number of matrices as the number of dimensions of the problem. Each new coefficient matrix corresponds to a single directional derivative. Each FDE from this ADI system is easier to solve because the corresponding coefficient matrix has a simpler form. The solution of the ADI system is again a numerical solution of the problem. Usually, it is different from the one obtained by the original FDE, since the ADI system and the original FDE (before the splitting) are, generally, not equivalent.

The kind of ADI splitting approach that we will use for the BDF-ADI method is called Douglas-Gunn splitting. For this particular type of splitting, the numerical solution obtained from the ADI system enjoys the same convergence properties as the one obtained from the original FDE. There are many ways to obtain the ADI
system from the original FDE, and vice versa. We will refer to the difference between the original FDE and the ADI system as the *error term*. As we will see, the presence of the error term can lead to different stability properties for the split method.

The backward differentiation formula (BDF) methods are a family of ODE discretizations commonly used to discretize stiff ODEs such as those arising from diffusion PDEs. One can use BDF methods to discretize the PDE in time, and some other convenient method to discretize it in space. Then, an ADI approach might be used to solve the resulting FDE easily. This a BDF-ADI discretization. In the motivating paper [1], numerical experiments are performed using BDF-ADI methods to solve the two-dimensional pressure-free momentum equation. These numerical experiments suggest that the scheme enjoys some stability properties. In this thesis, we investigate these results by studying the absolute stability of BDF-ADI discretizations.

In this thesis we focus on the model

\[ u_t = \alpha_1 u_x + \beta_1 u_y + \alpha_2 u_{xx} + \beta_2 u_{yy} + \gamma u_{xy} \quad (1.1) \]

where \( \alpha_1, \beta_1 \in \mathbb{R}, \alpha_2, \beta_2 > 0 \) and \( \gamma^2 \leq 4\alpha_2\beta_2 \). The goal of this project is to study the absolute stability the BDF-ADI scheme for (1.1). This analysis will be done by investigating the hyperbolic and the parabolic equations separately. Since the analysis of the mixed problem is a difficult problem, the results for both these equations will help us to understand the mixed problem better. The study of the stability of BDF-ADI methods for a scalar linear PDE works as a building block towards similar investigations for variable-coefficient and nonlinear problems. Moreover, a Fourier spectral discretization in space is used to verify the claimed stability results.

Among the vast literature on ADI methods, we would like to introduce four references which go back to the introduction of the ADI methods, [2, 3, 4, 5]. The first paper [2] presents an alternating implicit method that does not require any error term for the splitting. The Crank-Nicolson-ADI method is applied in detail to the
case of the two-dimensional parabolic equation in [3]. Therein is proven unconditional stability and convergence of this method. The ADI method as it is presented in this manuscript, i.e., with the same error term, is finally introduced in [4], using the BDF1 scheme, also known as the Backward Euler method. These references study consistency, stability, and convergence using Fourier analysis (see Section 9.6 of [6]). The most widely used example is the heat equation. The method to prove the stability is combining the superposition principle with the separation of variables technique and solving the same difference equation for the truncation errors. This approach leads to a solution that can be written in the form of a Fourier series, with Fourier coefficients $c_k$. By proving that $|\frac{c_{k+1}}{c_k}| < 1$ in some norm, then the truncation error goes to zero as the time steps decrease, and convergence is established. This is not always doable since the space operator can be very complex. This also gives cumbersome expressions for the eigenfunctions (or no expressions at all), which will make much more difficult the evaluation of $|\frac{c_{k+1}}{c_k}| < 1$. The present work presents a different approach to the proof of stability, that does not just make the proof simpler for BDF1 but can also be applied to prove the same results for higher-order BDF methods. This approach is called the Schur-Cohn polynomial reduction.

The notion of stability in this thesis differs from the stability concept used in the motivating paper [1]. To define this notion of stability, consider a linear equation in the matrix form $u' = D_h u$, where $D_h$ is a coefficient matrix of some spatial discretization on a constant grid with size $h$. Consider arbitrary and constant time step $\Delta t$. Let $U^0, ..., U^n$ be the initial data. A method is unconditionally stable in the sense of [1] if the numerical solution can be bounded by the product of the sum of the initial data, and a constant that depends on the final time, i.e., $|U^n| < C_T \sum_{j=0}^n |U^{n-k}|$, for some norm $|.|$. In the same sense, a method is quasi-unconditionally stable if the previous bound holds for any $\Delta t < M_t$ and $h < M_h$. On the other hand, the concept of stability considered in this thesis is the so-called absolute stability. This notion is
based on the linear scalar model \( u' = \lambda u \). A method is \textit{absolutely stable} for this ODE if the one-step errors do not grow in future times, \([6]\). For the case of stability in the sense of \([1]\), although the growth of the numerical errors is controlled by the previous bound, it might propagate in future times if \( C_T \) is large enough. For this reason, absolute stability is a stronger notion of stability than the one considered in \([1]\).

It is common to talk about the \textit{region of absolute stability} as a complex region in the \( z \) plane where \( z = \lambda \Delta t \). This region is the set of values \( z \) for which the method is absolutely stable. For a linear PDE, the parameter \( \lambda \) represents an eigenvalue of a discrete operator in space, which may be a coefficient matrix for some spatial discretization. In the nonlinear case, the PDE is typically linearized, and the eigenvalues of the Jacobian are considered. It is therefore important to determine the eigenvalues of this discrete operator to study stability. It is common to consider a linear problem to study the stability of a nonlinear model. Although the final application of BDF-ADI discretizations is on nonlinear problems, stability for the linear problem is at least a necessary condition for good behaviour for nonlinear problems.

The motivating paper \([1]\) presents proofs of unconditional stability of BDF2-ADI schemes for the periodic hyperbolic problem, as well as both for periodic and non-periodic parabolic cases. Moreover, quasi-unconditional stability of un-split BDF methods of orders \( 2 \leq s \leq 6 \) is also proven.

In this thesis, we extended the results of BDF2-ADI to absolute stability. Proofs of these results are presented for periodic hyperbolic and periodic parabolic cases. Furthermore, regions of absolute stability are given for BDF3-ADI and BDF4-ADI, which have not been studied in \([1]\).

There is a handy tool to study the absolute stability of a scheme, and it is directly related to its FDE: the \textit{stability polynomial}. The way the stability polynomial is useful will be discussed in detail later in this thesis, but for now, we would like to stress the importance of the roots of this polynomial. It is important to find the values of \( z \) for
which the stability polynomial has roots inside the unit disk. The problem is that the degree of these polynomials equals the order of the BDF method that is being used. This yields cumbersome polynomials with degrees of order greater than 3. The Schur-Cohn polynomial reduction gives an algorithm to find the region of parameters for which the roots of these polynomials are inside the unit disk.

This thesis is organised as follows. In Chapter 2 we review the stability and convergence theory, as well as a detailed explanation of the Schur-Cohn polynomial reduction algorithm. Furthermore, in Chapter 3 we give an introduction to the BDF-ADI spectral collocation method. In Chapter 4 we present the main stability results along with some conjectures. Most of the theoretical results are proven using the software Mathematica. The results in Chapter 4 are supported with numerical implementations presented in Chapter 5. Finally, in the Concluding Remarks we present suggestions for future work.
Chapter 2

Stability of Numerical Methods

This thesis is a stability analysis of a numerical scheme. To understand the results, we need to know what kind of stability we are concerned with and what numerical scheme we want to study. This chapter concerns the former.

The family of BDF methods is a member of the family of linear multistep methods (LMM). The fundamental theorem of numerical analysis implies that a consistent LMM applied to the test equation \( u' = \lambda u \), is convergent if and only if it is stable. Firstly, the three fundamental notions of consistency, stability and convergence will be presented. After this, we give the definition of LMM. Most of the stability results are obtained via the stability polynomial, whose definition is given after that. Finally, the two essential concepts of stability will be presented; namely, zero-stability and absolute stability. The reader may find these definitions in numerous textbooks. In this case, we follow [7].

Consider the linear test problem,

\[ u' = \lambda u \tag{2.1} \]

where \( \lambda \in \mathbb{C} \). Let \( U^N \) denote the numerical solution at time \( T \) computed using \( N \) steps, each with length \( \Delta t \). To start a multistep method, one needs \( r \) starting values \( U^\nu, \nu = 0, \ldots, r - 1 \). Assume that each of these starting values converges to the value of the solution at \( t_0 \), as \( \Delta t \to 0 \), i.e.,
\[
\lim_{\Delta t \to 0} U^{\nu} = u(t_0) \quad \text{for } \nu = 0, 1, ..., r - 1. \tag{2.2}
\]

Convergence of a numerical method means that
\[
\lim_{\Delta t \to 0} U^N = u(T) \quad \text{with } N\Delta t = T \tag{2.3}
\]
for all problems in a reasonably large class of problems. In summary, a method for (2.1) is convergent if (2.3) holds whenever the initial values satisfy (2.2).

To introduce consistency, we need to introduce the local truncation error (LTE).

Fix a spatial grid with size \( h \) and let \( D_h \) be a coefficient matrix of a numerical scheme. If \( W^n_h \) is the approximation of \( u'(t_n) \) on the spatial grid, then \( D_h U^n_h = W^n_h \) holds for all fixed times \( t_n \). Consistency says that, at each time step, \( W^n_h \) converges to \( u'(t_n) \) as the spatial grid is refined. We define the LTE as
\[
\tau_h := W^n_h - u'(t_n) \tag{2.4}
\]
on the points of the spatial grid. Hence, a method is consistent if \( \lim_{h \to 0} \tau_h = 0 \).

The general form of a \( s \)-step LMM applied to (2.1) is
\[
\sum_{k=0}^{s} c_k U^{n+1-k} = \lambda \Delta t \sum_{k=0}^{s} d_k U^{n+1-k}. \tag{2.5}
\]
In particular, a BDF method is a LMM with \( d_k = 0 \) for \( k > 0 \). We can talk about two different kinds of stability for a LMM. These are zero-stability and absolute stability. Both these notions are closely related to the notion of first characteristic polynomial and stability polynomial, respectively, and root condition.

Define the scaled eigenvalue \( z := \lambda \Delta t \). The stability polynomial of a LMM is defined as

\[ P(\zeta; z) := \sum_{k=0}^{s} (c_k - zd_k)\zeta^{s-k}. \] (2.6)

Another important polynomial is the first characteristic polynomial, defined as \( \rho(\zeta) := P(\zeta; 0) \). In the case of BDF2, the stability polynomial can be written as \( P(\zeta; z) := (1 - z)\zeta^2 - 4/3\zeta + 1/3 \). We shall see later how these methods are formed and other ways that we can write these and other polynomials. The notions of zero stability and absolute stability are related to the roots of these polynomials. A polynomial is said to satisfy the root condition if the roots \( \zeta_j \) satisfy the following conditions,

\[ |\zeta_j| \leq 1 \quad \forall j; \] (2.7)

If \( \zeta_j \) is a repeated root, then \( |\zeta_j| < 1 \).

A LMM is said to be zero-stable if its corresponding first characteristic polynomial satisfies the root condition \((2.7)\). By the definition of \( \rho \), zero-stability is a notion of stability for \( \Delta t = 0 \). To determine whether a method can give reasonable results with a given time step \( \Delta t > 0 \) we need a notion of stability different from zero-stability. This new concept is called absolute stability. The set of points \( z \) in the complex plane for which the stability polynomial satisfies the root condition is called the absolute stability region, denoted by \( \mathcal{R} \). A method is absolutely stable if its stability polynomial \( P(\zeta; z) \) satisfies the root condition \((2.7)\). A method is unstable otherwise.

Although the BDF-ADI is not an LMM, its absolute stability can be characterised in a similar way. However, we will see that its stability polynomial does not depend on just one parameter \( z \), but instead, on several parameters \( z_1, z_2, ..., z_n \). This dependency makes the stability analysis harder since the absolute stability region is now a higher-dimensional complex space. As an example, in the ADI context is useful to
look at the test function $u' = \lambda u$ in the form $u' = \lambda_1 u + \lambda_2 u$, where $\lambda_1 + \lambda_2 = \lambda$. Instead of just one parameter $z$, the latter ODE yields two parameters $z_1 := \lambda_1 \Delta t$ and $z_2 := \lambda_2 \Delta t$. This gives a four-dimensional absolute stability region (two-dimensional for the imaginary part of $z_i$ and another two for their real part). However, if we consider $\lambda_i$ to be purely imaginary (or real) the four-dimensional absolute stability region reduces to a two-dimensional one. In the next chapter, we shall analyse the stability regions in the complex $z$ plane for BDF methods and compare them with the corresponding stability regions in the real $(z_1, z_2)$ plane, for particular assumptions on $\lambda_1$ and $\lambda_2$. This transition will be useful to understand the analogous stability regions in the real $(z_1, z_2)$ plane for BDF-ADI methods. Just like the stability regions change with the introduction of more parameters, the stability polynomial will also depend on more parameters, i.e., $P(\zeta; z)$ becomes $P(\zeta; z_1, ..., z_n)$. Henceforth, we will call an absolutely stable method simply as a stable method.

As we have seen, by the fundamental theorem of numerical analysis, it is important to understand whether a method is consistent and stable. It is relatively easy to compute the LTE, and thus, deduce whether a method is consistent or not. It is not so easy to understand whether a stability polynomial satisfies the root condition, especially if it depends on many parameters. A polynomial that satisfies the root condition is also called a simple von Neumann polynomial. In the next section, we present an algorithm that allows us to infer whether a polynomial is simple von Neumann.

### 2.1 Schur-Cohn polynomial reduction

This section introduces an algorithm to determine the location of the zeros of a polynomial relative to the unit disk. This algorithm is known as the Schur-Cohn polynomial reduction and it is described in [8, 9]. The first references to this algorithm go back to the early twentieth century, with the independent work by Schur, in [10, 11],
and Cohn in [12].

After introducing the concept of simple von Neumann polynomial, we define the reduced polynomial of a given polynomial. The latter is a polynomial of degree less than the original polynomial. Finally, we introduce necessary and sufficient conditions for a polynomial to be simple von Neumann.

Let $\mathcal{D} := \{ \zeta \in \mathbb{C} : |\zeta| < 1 \}$ be the open disk and $\mathcal{S} := \{ \zeta \in \mathbb{C} : |\zeta| = 1 \}$ its boundary. Consider the complex polynomial of the form,

$$P(\zeta) := a_0 + a_1 \zeta + ... + a_s \zeta^s$$  \tag{2.8}

with degree $n$ ($a_s \neq 0$) and non-zero at the origin ($a_0 \neq 0$). A polynomial is simple von Neumann if it has zeros in $\mathcal{D}$ or simple zeros on $\mathcal{S}$. Moreover, consider the reversed polynomial version of $P$

$$P^*(\zeta) := a_s + a_{s-1} \zeta + a_{s-2} \zeta^2 + ... + a_0 \zeta^s.$$

The reduced polynomial of $P$ is defined as

$$P_1(\zeta) := \frac{P^*(0)P(\zeta) - P(0)P^*(\zeta)}{\zeta}.$$  \tag{2.9}

The following result determines whether $P$ is simple von Neumann, given the roots of $P_1$.

$P$ is a simple von Neumann polynomial if and only if:

- either $|a_s| > |a_0|$ and $P_1$ is simple von Neumann,

- or $P_1 \equiv 0$ and all roots of the derivative $P_1'$ are in $\mathcal{D}$.

This result is an inclusive disjunction, therefore, we can focus on just one argument. For simplicity, we will refrain from checking the second condition, and focus
solely on the first one. The first condition of this result yields an algorithm. The main point of this algorithm is that the reduced polynomial has degree strictly smaller than the original polynomial. Iterating this procedure, we arrive at a degree-one polynomial, whose root is easy to find.

This algorithm makes the investigation of zeros of the stability polynomial much easier to handle. Indeed, this creates an algorithm where, at first, one checks whether \( |a_s| > |a_0| \). Then, if this holds, applies the previous result to the reduced polynomial at step \( k \), \( P_1^{(k)} \). The degree of \( P_1^{(k)} \) decreases by one unit in each new iteration, until \( P_1^{(s-1)} \) has degree one. The root of \( P_1^{(s-1)} \) is simple to find explicitly. \( P \) is a simple von Neumann if this root is in \( \mathcal{D} \). A pseudocode of the Schur-Cohn polynomial reduction is shown below, and a Mathematica code is presented in Appendix A.

---

**Algorithm 1:** The Schur-Cohn polynomial reduction.

**Data:** \( s \)-degree stability polynomial \( P(\zeta; z) := a_0 + \ldots + a_s \zeta^s \)

**Result:** Absolute stability region \( \mathcal{R} \)

1. Set \( \mathcal{R} := \{ z \in \mathbb{R} : |a_s(z)| > |a_0(z)| \} \);

2. for \( k=1,\ldots,s-1 \) do

   3. Compute \( P_1^{(k)} \), the reduced polynomial of \( P \);

   4. Define \( a_s^{(k)}(z) \) as the leading coefficient of \( P_1^{(k)} \);

   5. Define \( a_0^{(k)}(z) \) as the constant coefficient of \( P_1^{(k)} \);

   6. Compute \( \mathcal{R}^{(k)} := \{ z \in \mathbb{R} : |a_s^{(k)}(z)| > |a_0^{(k)}(z)| \} \);

   7. Update \( \mathcal{R} := \mathcal{R} \cap \mathcal{R}^{(k)} \);

   8. Update \( P = P_1^{(k)} \);

9. end

10. return \( \mathcal{R} \)

---

The previous pseudocode illustrates the Schur-Cohn polynomial reduction algo-
rithm for a polynomial that depends on a single parameter $z$. As we have seen, we are interested in finding the location of the roots of polynomials that depend on several parameters. If a polynomial of the form $P(\zeta; z_1, \ldots, z_n)$ is considered, regions $\mathcal{R}^{(k)} := \{(z_1, \ldots, z_n) \in \mathbb{R}^n : |a_j^{(k)}(z_1, \ldots, z_n)| > |a_0^{(k)}(z_1, \ldots, z_n)|\}$ must be computed. Such regions and the corresponding intersections are obtained using the command Reduce from Mathematica. Computing each of these regions is not an easy task, and it is a major limitation of the algorithm. The reason is twofold:

- The expressions for the coefficients $a_j^{(k)}$ are more complex as $k$ increases;
- As the number of parameters $(z_1, ..., z_n)$ increases, the more difficult it is to compute $\mathcal{R}$.

As an example of the limitations of the algorithm, consider the polynomial obtained by applying BDF2-ADI to the advection-diffusion equation (1.1). This is a polynomial of degree 2 with five parameters. Using the previous algorithm, we need to compute two regions $\mathcal{R}$ and $\mathcal{R}^{(1)}$ that depend on the parameters $(z_1, ..., z_5)$, lines 1 and 6 of Algorithm 1. After that, we need to compute the intersection of this pair of regions, line 7. Both these calculations are done using the command Reduce, similarly to the way the region $\text{regPar4M}$ is computed in Appendix A. However, in this case, the algorithm produces no output.
Chapter 3

BDF-ADI Spectral Collocation Method

In this chapter, we introduce a numerical scheme to solve the PDE (1.1). Spectral collocation methods and BDF methods are used as space and time discretizations, respectively. We introduce both these discretizations separately and finally assemble them into a single numerical scheme for (1.1), the so-called BDF-ADI spectral collocation method.

The primary goal of this thesis is to understand how the stability properties change from BDF to BDF-ADI methods. Together with some appropriate spatial discretization, these schemes are used to solve a problem of the form (1.1).

The stability analysis of the BDF and BDF-ADI methods is done using the linear ODE (2.1). The question is: which parameter $\lambda$ shall be used? A possible answer is: In theory, all parameters from the infinite set of eigenvalues $\Lambda := \{\lambda \in \mathbb{R} : \alpha_1 u_x + \beta_1 u_y + \alpha_2 u_{xx} + \beta_2 u_{yy} + \gamma u_{xy} = \lambda u\}$, where $u$ is a non-zero solution of (1.1). Hence, the stability analysis of a time discretization for (1.1), is given by studying the problems $u' = \lambda u$ for $\lambda \in \Lambda$. In practice, we cannot use all parameters $\lambda \in \Lambda$, because this is an infinite set. Therefore, we need to choose a finite subset of parameters in $\Lambda$. Let $D$ be the discrete operator obtained by discretizing the function $f(u_x, u_y, u_{xx}, u_{yy}, u_{xy}) = \alpha_1 u_x + \beta_1 u_y + \alpha_2 u_{xx} + \beta_2 u_{yy} + \gamma u_{xy}$ with a spectral method. The finite set $\Lambda^*$ of eigenvalues of $D$ is a subset of $\Lambda$. Therefore, if we find the eigenvalues of a spectral operator based on Fourier collocation of $f$, we find a discrete subset of elements of $\Lambda$. Hence, we can analyse the stability of BDF and BDF-ADI schemes for (1.1) by applying these schemes to the finite number of problems $u' = \lambda u$, ...
for $\lambda \in \Lambda^*$.

The solution for the PDE (1.1) is a smooth function $u(t, x, y) \in \mathbb{R}$ defined on $(t, x, y) \in [0, +\infty) \times \mathbb{R}^2$. In order to find a solution we need to know the values of this function at the initial time, i.e., at $t = 0$. The initial condition will be represented by $u_0(x, y) := u(t_0 = 0, x, y)$. After setting the initial condition we need to build a grid in space, i.e., a set of points in space $\{(x_j, y_k) \in \mathbb{R}^2\}$ on which $u$ will be approximated. The stability of the BDF-ADI discretization may depend on the boundary conditions of the problem. In this work we consider the simplest type of boundary conditions; namely, periodic boundaries. Stability for periodic boundaries is often a necessary condition for stability under other boundary conditions. In this thesis we use the spatial domain $\Omega := [-L, L] \times [-L, L] \subset \mathbb{R}^2$. The solution $u(t, x, y)$ is periodic in $\Omega$. The discrete version of $\Omega$, i.e., the spatial grid, will be denoted by $\Omega_h$, where $h$ is the distance between each consecutive grid points, $h := |x_j - x_{j-1}| = |y_k - y_{k-1}|$. Let $m$ denote the number of space points in each direction, also called collocation points. Formally, the spatial grid is defined as $\Omega_h := \{(x_j, y_k) \in \Omega : j, k = 0, 1, ..., m-1\}$. Moreover, set a time step $\Delta t$ and the final time $T$ at which the solution will be computed. The number of steps needed to reach the solution at time $T$ is denoted by $N$. Indeed, $N := T/\Delta t$. Finally, we can define the numerical solution $U^n \in \mathbb{R}^{m \times m}$. This is an approximation of the true solution at time $t = t_n$ for $n = 0, 1, ..., N$ on the grid $\Omega_h$, i.e., $U^n \approx u(t_n, x, y)$ on $\Omega_h$. Let $u^n_{j,k} \in \mathbb{R}$ denote the approximation of the true solution at a particular point $(j, k)$, i.e., $u^n_{j,k} \approx u(t_n, x_j, y_k)$ for some $(j, k)$. In particular, $(U^n)_{j,k} = u^n_{j,k}$.

This section starts by introducing the spectral collocation methods, also known as spectral methods. Afterwards, the BDF methods are presented, along with their corresponding stability regions. The last section shows and describes the BDF-ADI spectral collocation method.
3.1 Spectral Collocation Methods

The values of the numerical solution at time $t = 0$ are exactly the values of the initial solution, that is $U^0 = u(0, x, y)$ on $\Omega_h$. From this square grid, we march in time towards time $T$, creating $N$ other subsequent square grids throughout the process. Assume the numerical method is consistent and stable with the original boundary value problem, and therefore, the numerical solution $U^n$ converges to $u(T, x, y)$ on $\Omega_h$.

In order to properly introduce the error of the numerical solution obtained by solving (1.1) using BDF-ADI spectral collocation methods, we need to introduce the semi-discrete form of (1.1). Consider a spatial discretization of (1.1). Since this equation is in a two-dimensional setting, the corresponding unknown will be an array $u_h(t) \in \mathbb{R}^{m \times m}$. Let $v_h(t) \in \mathbb{R}^{m^2}$ be a vector that represents $u_h(t)$ in some appropriate ordering. Let $D \in \mathbb{R}^{m^2 \times m^2}$ denote a coefficient matrix. The semi-discrete form of (1.1) is the system of ODE given by,

$$v'_h(t) = Dv_h(t).$$

Let $\epsilon = \epsilon_{\Delta t} + \epsilon_h$ denote the difference between $U^N$ and $u(T, x, y)$ on $\Omega_h$, introduced by the BDF-ADI spectral collocation method. The value $\epsilon$ is the sum of two independent values. The value $\epsilon_h$ is the difference between the exact solution $u(T, x, y)$ on $\Omega_h$ and the exact solution $u_h(T)$ of the semi-discrete form. The difference between these two is the error given by the spatial discretization. Now, consider a time discretization of the previous semi-discrete form. This yields a fully-discretized system of scalar FDE, whose solution at time $T$ is exactly $U^N$. The value $\epsilon_{\Delta t}$ is the difference between $u_h(T)$ and $U^N$. The difference between these two is the error given by the time discretization. The BDF-ADI method converges at a polynomial rate, which means $\epsilon_{\Delta t} = \mathcal{O}(\Delta t^p)$ for some $p > 0$. On the other hand, spectral methods converge at a rate faster than
polynomial. This means that, for $\Delta t$ and $h$ with relatively same magnitudes, the value of $\epsilon$ will be dominated by $\epsilon_{\Delta t}$. Therefore, under these assumptions on $\Delta t$ and $h$, the order of convergence of the BDF-ADI spectral collocation method is given by the order of convergence of the BDF-ADI discretization, i.e., $U^N = u(N, x, y) + O(\Delta t^p)$ for some $p > 0$.

3.1.1 Fourier analysis in one dimension

In this section, we consider a one-dimensional setting. The goal is to approximate the derivative $u'$ of a smooth function $u$ on the partition points $x_j$, given a set of data $u(x_j)$. The basic principle of this method is the following:

1. Compute an interpolating polynomial of $u$, denoted by $p$;
2. Find $p'$;
3. Consider $p'$ as an approximation of $u'$.

We are free to choose $p$, and given the periodic domain and smooth properties of $u$, the natural choice is a trigonometric polynomial. This type of spectral methods is usually called Fourier spectral methods.

The following description is mostly based on the first four chapters of [13]. A less exhausting description of spectral methods with interesting applications can be found in [14].

The set of functions $\{e^{i\xi x} : \xi \in \mathbb{R}\}$ defines an orthonormal basis in $L^2(\mathbb{R})$. We start by assuming that $u \in L^2(\mathbb{R})$. In fact, for the purpose of this thesis, we just need $u \in L^2([-L/2, L/2])$ and periodic. This weaker assumption on $u$ simplifies the theory considerably. In this subsection, we will describe why this simplification is important and how it is done.

Let $u \in L^2(\mathbb{R})$. We can write $u$ as an integral expansion of this basis,
\[ u(x) = \frac{1}{L} \int_{-\infty}^{\infty} \hat{u}(\xi)e^{i\xi x} d\xi. \]  (3.1)

The function that gives the coefficients of this integral \( \hat{u}(\xi) \) is called the Fourier transform of \( u \). The process of decomposing a function using this approach is called Fourier analysis. The Fourier transform can also be obtained from the function \( u \)

\[ \hat{u}(\xi) := \int_{-\infty}^{\infty} u(x)e^{-i\xi x} dx. \]  (3.2)

We call \( u \) the inverse Fourier transform of \( \hat{u} \). The variable \( x \) is called the physical variable, and the variable \( \xi \) is called the wavenumber or frequency variable. Formulas (3.1) and (3.2) are the inverses of each other.

We go back to our primary goal: to compute an interpolating polynomial of \( u \), whose derivative will approximate \( u' \). That polynomial will arise from the right-hand-side of (3.1). We only need to assume one thing: the spatial domain is discrete and unbounded, i.e., \( x \in h\mathbb{Z} \). This assumption yields a bounded and continuous Fourier space interval. To understand how such interval comes into play, one needs to understand the phenomenon of aliasing.

Fix \( \xi \in \mathbb{R}\setminus\{0\} \) and consider the complex-valued exponentials \( f_1(x) = e^{i\xi_1 x} \) and \( f_2(x) = e^{i\xi_2 x} \). The value of both these functions is a point that just “spins” around the unit circle as the product \( \xi x \) increases. The two functions have the same value at a given point \( x_j = jh \) whenever \( (\xi_1 - \xi_2)jh = 2\pi n \), for some integer \( n \). In other words, whenever \( \xi_1 - \xi_2 \) is a multiple of \( \frac{2\pi}{h} \). Hence, we just need to consider wavenumbers \( \xi \) within an interval of length \( \frac{2\pi}{h} \). For symmetry reasons, we consider \([ -\frac{\pi}{h}, \frac{\pi}{h} ] \). Hence,

\[
\frac{1}{L} \int_{-\infty}^{\infty} \hat{u}(x_j)e^{i\xi x_j} d\xi = \lim_{h \to 0} \frac{1}{L} \int_{-\frac{\pi}{h}}^{\frac{\pi}{h}} \hat{u}(x_j)e^{i\xi x_j} d\xi, \quad j \in \mathbb{Z}.
\]

Note that the aliasing argument for the wavenumber \( k \) only holds because we are assuming the discrete domain of \( x \) is unbounded. Since our data lives in a discrete
and unbounded domain, we can approximate (3.2) using the trapezoidal rule,

$$
\hat{u}(\xi) = \lim_{h \to 0} \frac{h}{j=\infty} \sum_{j=-\infty}^{j=\infty} e^{-i\xi x_j} u(x_j), \quad \xi \in \left[-\frac{\pi}{h}, \frac{\pi}{h}\right].
$$

The two previous formulas have special names. The *inverse semidiscrete Fourier transform* is given by

$$
\mathcal{F}^{-1}(x_j) := \frac{1}{L} \int_{-\frac{\pi}{h}}^{\frac{\pi}{h}} \hat{u}(\xi) e^{i\xi x_j} d\xi, \quad j \in \mathbb{Z}
$$

(3.3)

and the *semidiscrete Fourier transform* is given by

$$
\mathcal{F}(\xi) := h \sum_{j=-\infty}^{j=\infty} u(x_j) e^{-i\xi x_j}, \quad \xi \in \left[-\frac{\pi}{h}, \frac{\pi}{h}\right].
$$

(3.4)

Finally, we can express the interpolating polynomial,

$$
p(x) = \frac{1}{L} \int_{-\frac{\pi}{h}}^{\frac{\pi}{h}} \hat{u}(x) e^{i\xi x} d\xi, \quad x \in \mathbb{R},
$$

(3.5)

which is called the *band-limited interpolant of u*. This will be the polynomial that interpolates the points given in (3.3). Recall that, to approximate $\hat{u}$, we use (3.4).

The practical point of these last three formulas is very limited, as it is impossible to have an unbounded discrete set of data. Therefore we need further assumptions on our physical space to narrow it to a bounded domain. This additional assumption is a periodic grid. Our basic grid will have $m$ even collocation points on $[-L^2, L^2]$.

We are interested in computing (3.5) numerically, using a discrete set of data $\{\xi_j\}$. We will choose $m$ wavenumbers evenly spaced within $[-\frac{\pi}{h}, \frac{\pi}{h}]$. Now notice that $h = \frac{L}{m}$, which yields

$$
\frac{\pi}{h} = \frac{\pi m}{L}.
$$

Therefore, instead of considering the interval $[-\frac{\pi}{h}, \frac{\pi}{h}]$ we can pick $m$ wavenumbers evenly spaced in $[-\frac{\pi m}{L}, \frac{\pi m}{L}]$. 
Nothing continuous or infinite is left in the problem, and we can finally compute numerically the band-limited interpolant of \( u \). To do so, we use the trapezoidal rule to approximate (3.1). Let \( u_j \approx u(x_j) \) and \( \hat{u}_l \approx \hat{u}(\xi_l) \), then

\[
    u_j := \frac{1}{L} \sum_{l=0}^{m-1} \hat{u}_l e^{i\xi_l x_j}, \quad x_j = -\frac{L(m-2j)}{2}, \quad j = 0, \ldots, m-1
\]

where the discrete Fourier transform (DFT) \( \hat{u}_l \) is given by

\[
    \hat{u}_l := h \sum_{j=0}^{m-1} u_j e^{-i\xi_l x_j}, \quad \xi_l = -\frac{\pi(m-2l)}{L}, \quad l = 0, \ldots, m-1
\]

Formula (3.6) is called the inverse discrete Fourier transform. Both formulas converge to \( u \) and \( \hat{u} \), respectively, as \( h \to 0 \) and \( m \to +\infty \). Interpolating each point from (3.6) we get the band-limited interpolant,

\[
    p(x) = \frac{1}{L} \sum_{l=0}^{m-1} \hat{u}_l e^{i\xi_l x}, \quad x \in \mathbb{R}.
\]

The reader might be a little bit confused with the abuse of notation between formulas (3.5) and (3.8). In fact, our only concern is to find an interpolating polynomial \( p \). Obviously, formulas (3.5) and (3.8) are not identical, but the latter is numerically computable, unlike the former. Therefore, the final interpolating polynomial will be given by formula (3.8).

After computing \( p \) we need to compute \( p' \). The polynomial \( p' \) is the interpolating polynomial through points \( w_j := p'(x_j) \).

### 3.1.2 Fourier analysis in two dimensions

Recall that we are primarily concerned in a two-dimensional setting. Furthermore, the solution for (1.1) is periodic in \( \Omega = [-\frac{L}{2}, \frac{L}{2}]^2 \) and also depends on time. The DFT formulas in two dimensions are similar to the ones above. The inverse discrete Fourier transform is given by
\[ u_{j,k}(t) := \frac{1}{L^2} \sum_{l,p=0}^{m-1} \hat{u}_{l,p}(t) e^{i\xi_l x_j} e^{i\xi_p y_k}, \quad j, k = 0, \ldots, m - 1, \quad t > 0. \tag{3.9} \]

The two-dimensional DFT is given by

\[ \hat{u}_{j,k}(t) := h^2 \sum_{l,p=0}^{m-1} u_{l,p}(t) e^{-i\xi_l x_j} e^{-i\xi_p y_k}, \quad j, k = 0, \ldots, m - 1, \quad t > 0. \tag{3.10} \]

### 3.1.3 A spatial discretization

In the previous section, we have seen how we can approximate derivatives of functions using spectral methods. Spectral collocation methods are used as a spatial discretization of the equation \((1.1)\). In this subsection, we will study how the above theory can be used towards this goal. We return to the two-dimensional setting.

Equation \((3.9)\) is a finite sum of \(m^2\) terms of the form \(\hat{u}(t)e^{i\xi_x x}e^{i\xi_y y}\), called Fourier modes, that depend on the pair \((\xi_x, \xi_y)\). Fix two wavenumbers \(\xi_x\) and \(\xi_y\) and consider a single Fourier mode. The function \(\tilde{u}(t) := \hat{u}_{\xi_x,\xi_y}(t)e^{i\xi_x x}e^{i\xi_y y}\) is smooth and solves \((1.1)\).

Now evaluate equation \((1.1)\) at \((x_j, y_k)\). Since \(\tilde{u}(t, x_j, y_k) = i\xi_x \tilde{u}(t, x_j, y_k)\) (similarly to the other spatial terms), then

\[ \tilde{u}_t = i\xi_x \tilde{u} + i\xi_y \tilde{u} - \xi_x^2 \tilde{u} - \xi_y^2 \tilde{u} - \xi_x \xi_y \tilde{u}, \]

which still depends on \(x\) and \(y\). Multiply both sides of the equation by \(e^{-i\xi_x x}e^{-i\xi_y y}\). This yields the mixed equation in the frequency space,

\[ \hat{u}' = i\xi_x \hat{u}_{\xi_x,\xi_y} + i\xi_y \hat{u}_{\xi_x,\xi_y} - \xi_x^2 \hat{u}_{\xi_x,\xi_y} - \xi_y^2 \hat{u}_{\xi_x,\xi_y} - \xi_x \xi_y \hat{u}_{\xi_x,\xi_y}. \tag{3.11} \]

The process from \((1.1)\) to \((3.11)\) is called spectral discretization. Finally, we can use a BDF method to find \(\hat{u}_{\xi_x,\xi_y}^N \approx \hat{u}_{\xi_x,\xi_y}(T)\). The previous process is a simplified
procedure because only one pair of wavenumbers is being considered. The initial
condition $u_0(x, y)$ written in the form (3.9) depends on $m^2$ pairs of wavenumbers.
Thus, the stability analysis of the numerical solution corresponding to this initial
condition is done by considering $m^2$ equations of the form (3.11) corresponding to $m^2$
pairs of eigenvalues. If the numerical solution of one of these equations is unstable,
the solution of the PDE will be unstable. The reason for that is because the sum
(3.9) will blow up if one of its coefficients goes off. We make use of formulas (3.9) and
(3.10) to derive an efficient algorithm to find a solution for (1.1) by spectral methods.

\begin{algorithm}
\textbf{Algorithm 2:} Spectral collocation method to find a solution for equation (1.1).
\begin{enumerate}
\item Define $U^0 \in \mathbb{R}^{m \times m}$ such that $u_0^{j,k} := (U^0)_{j,k}$ are the exact values of $u_0(x, y)$ on
$\Omega_h$;
\item Find $\hat{U}_j^0$ by applying formula (3.10) to $U^0_{j,k}$;
\item Initiate $\hat{U}_N \in \mathbb{R}^{m \times m}$ with null entries;
\item for $j,k = 0,\ldots, m-1$ do
\item Find $\hat{u}_j^N$ by solving (3.11) for the point $(j, k)$ with $N$ steps;
\item Update $(\hat{U}_j^N)_{j,k} = \hat{u}_j^N$;
\item end
\item Find $U_{j,k}^N$ by applying formula (3.9) to $\hat{U}_{j,k}^N$;
\item return $U_N$, the numerical solution for (1.1) at time $T$
\end{enumerate}
\end{algorithm}

There are several well-known algorithms to compute $\hat{U}$, one of which is the \textit{Fast Fourier Transform (FFT) algorithm}. We say $\hat{U}$ is the FFT of $U$, if (3.10) is computed
using the Fast Fourier Transform algorithm. Formula (3.9) can be computed by the
inverse FFT algorithm. Another way of computing (3.10) is by using a differentiation
matrix. See [13] for details on how to find these matrices.

3.1.4 Spectrum of a spectral differentiation matrix based on Fourier collocation

A continuous differential operator has infinitely many eigenvalues. One of the most important properties of a spectral differentiation matrix based on Fourier collocation is that its eigenvalues converge very fast to the eigenvalues of the corresponding differential operator.

As an example, consider the second derivative differential operator \( \frac{d^2}{dx^2} \). Let \( u \) denote a periodic function defined on the interval \([0, 2\pi]\). Consider the eigenvalue problem \( \frac{d^2}{dx^2} u = \lambda u \) with Dirichlet boundary conditions \( u(0) = u(2\pi) = 0 \). The eigenvalues of this operator are given by \( \lambda_k \in \Lambda := \{ \lambda_k = -k^2/4, k = 0, 1, 2, ... \} \), which is an infinite set. Furthermore, let \( D_m \in \mathbb{R}^{m \times m} \) denote a periodic spectral differentiation matrix. Examples of such matrices are given in Chapter 3 of [13]. Let \( U = [u_j]_{j=1}^m \) denote the discrete approximation of \( u \) on a grid with even number of \( m \) points in \([0, 2\pi]\). Each coordinate \( u_j \) can be written in the form of (3.6). The matrix-vector product \( D_m U \) gives a discrete approximation of the function \( u''(x) = \frac{d^2}{dx^2} u \) on the same grid. Therefore, each entry of this product is an approximation of the second derivative of the right-hand-side of (3.6) at the point \( x_j \). Hence, \( D_m U \approx -\xi^2 U \). Thus, the solution \( \lambda^* \) to the eigenvalue problem \( D_m U = \lambda^* U \) converges to an element in the set \( \Lambda_m := \{ \lambda_{k'} \in \mathbb{R} : \lambda_{k'} = -(\frac{m}{2} + k')^2, k' = 0, ..., \frac{m}{2} - 1 \} \) as \( m \) goes to \( \infty \).

As the previous example illustrates, each eigenvalue \( \lambda^* \) of the differentiation matrix \( D_m \) converges to some \( \lambda_k \in \Lambda_m \) as the number of collocation points \( m \) increases. The rate of convergence is very quickly, as the following example illustrates. This example was taken from Chapter 4 of [13]. Consider the eigenvalue problem,

\[
\left( -\frac{d^2}{dx^2} + x^2 I \right) u = \lambda u
\]
where $Iv = vI = v$. This problem is equivalent to $-u'' + x^2u = \lambda u$. The rapid decay of the solution allows us to consider Dirichlet boundary conditions in a periodic domain $[-L, L]$, for $L$ sufficiently large. Indeed, the eigenvalues of $\left(-\frac{d^2}{dx^2} + x^2I\right)$ are $\lambda = 1, 3, 5, \ldots$. Using a second order periodic spectral differentiation matrix, see [13], it is possible to approximate the first four eigenvalues of this harmonic oscillator by 13 digits with just 36 collocation points. This accuracy is due to the fast convergence of formulas (3.6) and (3.7) to their corresponding formulas (3.1) and (3.2). For the type of functions $u$ that we are working with, these formulas converge at a rate of $O(m^{-p})$ for every $p > 0$, [13]. Such behaviour is known as spectral accuracy.

We close this subsection addressing the cost of computing the DFT of a function. Spectral derivatives can be computed by differentiation matrices in $O(m^2)$ or by the FFT in $O(m \log(m))$ floating point operations.

### 3.2 BDF Methods

Consider once again the initial value problem (IVP) (1.1) with initial condition $u(0, x, y) = u_0(x, y)$. From the previous section, we know how to discretize this IVP in space and obtain a new IVP in the frequency space given by

$$
\begin{align*}
\hat{u}' &= \lambda \hat{u} \\
\hat{u}_{\xi_x, \xi_y}(0) &= \hat{u}_{0, \xi_x, \xi_y}
\end{align*}
$$

where $\lambda = i\alpha_1 \xi_x + i\alpha_2 \xi_y - \beta_1 \xi_x^2 - \beta_2 \xi_y^2 - \gamma \xi_x \xi_y$.

In this section, we will describe how BDF methods can be applied to (3.12). The common complex-valued regions of absolute stability will be introduced. One-dimensional segments of these complex-valued regions will be represented in two-dimensional real-valued regions. This representation is relevant for the stability results of BDF-ADI presented in the next chapter. The BDF methods enjoy nice properties for a variety of problems. One such advantage of BDF methods is that
these can perform well for stiff problems.

### 3.2.1 Backward Differentiation Formula

The *backward difference formula* (BDF) is a family of implicit schemes to solve the IVP (3.12). These schemes were first introduced by Curtiss and Hirschfelder in [15]. This subsection follows Chapter 8 of [7].

Let $\hat{u}_{j,k}^n \in \mathbb{R}$ denote the approximation of $\hat{u}(t_n, \xi_j, \xi_k)$ at a particular point $(\xi_j, \xi_k)$. Define $\hat{U} \in \mathbb{R}^{m \times m}$ as $(\hat{U})_{j,k}^n = \hat{u}_{j,k}^n$. For the sake of simplifying notation, we will usually represent $\hat{u}_{j,k}^n$ as $\hat{u}^n$. The simplest BDF method is the Backward Euler method, henceforth called as BDF1. This is a first order method given by,

$$
\hat{u}^{n+1} - \hat{u}^n = \Delta t \lambda \hat{u}^{n+1}.
$$

(3.13)

The BDF family goes from order 1 to 6, and it is given by

$$
\hat{u}^{n+1} - \sum_{k=0}^{s-1} a_k \hat{u}^{n-k} = \Delta t b \lambda \hat{u}^{n+1}
$$

(3.14)

where $s$ is the order of the method, and $a_k$ and $b$ are the *BDF coefficients* in Table 3.2.1

<table>
<thead>
<tr>
<th>$s$</th>
<th>$a_0$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{4}{3}$</td>
<td>$-\frac{1}{3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\frac{2}{3}$</td>
</tr>
<tr>
<td>3</td>
<td>$\frac{18}{11}$</td>
<td>$-\frac{9}{11}$</td>
<td>$\frac{2}{11}$</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{6}{11}$</td>
</tr>
<tr>
<td>4</td>
<td>$\frac{48}{25}$</td>
<td>$-\frac{36}{25}$</td>
<td>$\frac{16}{25}$</td>
<td>$-\frac{3}{25}$</td>
<td></td>
<td></td>
<td>$\frac{12}{25}$</td>
</tr>
<tr>
<td>5</td>
<td>$\frac{300}{137}$</td>
<td>$-\frac{300}{137}$</td>
<td>$\frac{200}{137}$</td>
<td>$-\frac{75}{137}$</td>
<td>$\frac{12}{137}$</td>
<td></td>
<td>$\frac{60}{137}$</td>
</tr>
<tr>
<td>6</td>
<td>$\frac{360}{147}$</td>
<td>$-\frac{450}{147}$</td>
<td>$\frac{400}{147}$</td>
<td>$-\frac{225}{147}$</td>
<td>$\frac{72}{147}$</td>
<td>$-\frac{10}{147}$</td>
<td>$\frac{60}{147}$</td>
</tr>
</tbody>
</table>

Table 3.1: Coefficients for BDF methods of order $s$, with $s = 1, \ldots, 6$

The stability polynomials are defined below,
\( s = 1: P_1(\zeta) := (1 - \Delta t\lambda)\zeta - 1, \)

\( s = 2: P_2(\zeta) := (1 - \frac{2}{3}\Delta t\lambda) \zeta^2 - \frac{4}{3}\zeta + \frac{1}{3}, \)

\( s = 3: P_3(\zeta) := (1 - \Delta t\lambda\frac{6}{11}) \zeta^3 - \frac{18}{11}\zeta^2 + \frac{9}{11}\zeta - \frac{2}{11}, \)

\( s = 4: P_4(\zeta) := (1 - \frac{12}{25}\Delta t\lambda) \zeta^4 - \frac{48}{25}\zeta^3 + \frac{36}{25}\zeta^2 - \frac{16}{25}\zeta + \frac{3}{25}, \)

\( s = 5: P_5(\zeta) := (1 - \frac{60}{137}\Delta t\lambda) \zeta^5 - \frac{300}{137}\zeta^4 + \frac{300}{137}\zeta^3 - \frac{200}{137}\zeta^2 + \frac{75}{137}\zeta - \frac{12}{137}, \)

\( s = 6: P_6(\zeta) := (1 - \frac{60}{147}\Delta t\lambda) \zeta^6 - \frac{120}{147}\zeta^5 + \frac{150}{147}\zeta^4 - \frac{400}{147}\zeta^3 + \frac{75}{147}\zeta^2 - \frac{24}{147}\zeta + \frac{10}{147}. \)

All the above polynomials have complex roots because they depend on \( \lambda \in \mathbb{C}. \)

The regions of stability for the product \( \Delta t\lambda \) in the complex plane can be found in any textbook, such as [7]. However, these regions can be seen from a different approach, when we are considering particular problem types from (3.12). The three types of problems are hyperbolic, parabolic and mixed. For the purpose of illustration, we will focus on the hyperbolic problem. The remaining types of PDEs have similar regions of stability but in higher dimensions.

Consider the hyperbolic problem in two dimensions. A spectral discretization for a fixed pair of wavenumbers \((\xi_j, \xi_k)\) yields the ODE,

\[ \hat{u}' = i\alpha_1\xi_j\hat{u} + i\beta_1\xi_k\hat{u}. \]  

(3.15)

Define \( A_1 := \Delta t\alpha_1\xi_j \) and \( B_1 := \Delta t\beta_1\xi_k \), where both of these parameters are real valued. Once again, we can write the corresponding stability polynomials as we did before,

\( s = 1: P_1(\zeta; A_1, B_1) := (1 - iA_1 - iB_1)\zeta - 1, \)

\( s = 2: P_2(\zeta; A_1, B_1) := (1 - i\frac{2}{3}A_1 - i\frac{2}{3}B_1) \zeta^2 - \frac{4}{3}\zeta + \frac{1}{3}, \)

\( s = 3: P_3(\zeta; A_1, B_1) := (1 - i\frac{6}{11}A_1 - i\frac{6}{11}B_1) \zeta^3 - \frac{18}{11}\zeta^2 + \frac{9}{11}\zeta - \frac{2}{11}. \)
• $s = 4 : P_4(\zeta; A_1, B_1) := \left(1 - i \frac{12}{25} A_1 - i \frac{12}{25} B_1\right) \zeta^4 - \frac{48}{25} \zeta^3 + \frac{36}{25} \zeta^2 - \frac{16}{25} \zeta + \frac{3}{25},$

• $s = 5 : P_5(\zeta; A_1, B_1) := \left(1 - i \frac{60}{137} A_1 - i \frac{60}{137} B_1\right) \zeta^5 - \frac{300}{137} \zeta^4 + \frac{300}{137} \zeta^3 - \frac{200}{137} \zeta^2 + \frac{75}{137} \zeta - \frac{12}{137},$

• $s = 6 : P_6(\zeta; A_1, B_1) := \left(1 - i \frac{60}{147} A_1 - i \frac{60}{147} B_1\right) \zeta^6 - \frac{120}{49} \zeta^5 + \frac{150}{49} \zeta^4 - \frac{400}{147} \zeta^3 + \frac{75}{49} \zeta^2 - \frac{24}{49} \zeta + \frac{10}{147}.$

Note that both lists represent the same polynomials when applied to (3.15). We shall focus on the last list of polynomials.

### 3.2.2 Stability Region

A BDF method is stable for the hyperbolic problem if the corresponding stability polynomial satisfies the root condition.

Figure 3.1 reveals the stability regions of the different BDF methods in the complex-valued plane, [7]. A two-dimensional real-valued representation of the imaginary axis of each one of these stability regions is given in Figure 3.2. A possible way to relate both regions is to consider the sum $A_1 + B_1$. The value of $i(A_1 + B_1)$ is a point along the imaginary axis in Figure 3.2. If the point $(A_1, B_1)$ in 3.2 is inside the stability region, then the point $i(A_1 + B_1)$ is stable on the imaginary axis of Figure 3.2.

It is clear from Figures 3.2(a) and 3.2(b) that the imaginary axis is inside the stability regions of BDF1 and BDF2. This means that the corresponding real-valued stability region will be $\mathbb{R}^2$. The term _unconditional stability_ will be used for such cases. This notion can be extended to higher-dimensional real-valued regions, corresponding to stability polynomials that depend on more than two parameters. For example, a method applied to the advection-diffusion equation (1.1) is unconditionally stable if the corresponding real-valued stability region is $\mathbb{R}^5$. 
Figure 3.1: Regions of stability (in the complex-valued plane) of BDF methods for different orders.
Figure 3.2: The real-valued stability regions of BDF methods for the hyperbolic equation. This is a two-dimensional representation of the imaginary axis of regions in Figure 3.1. The BDF1 and BDF2 methods are unconditionally stable.
3.3 BDF-ADI Spectral Collocation Methods

Using the theory of the last section, we finally have the last tool to implement Algorithm 2. Use some BDF method to perform line 5. In this section, we will describe the BDF-ADI method and see how it can be used to perform line 5 of Algorithm 2 instead.

In general, ADI methods differ by their corresponding error term. In this thesis, the particular error term gives rise to the so-called Douglas-Gunn splitting. This splitting form is introduced in [4] and it is named due to the extensive analysis in [16] by the authors.

The Douglas-Gunn splitting is an ADI approach of at most second order. Even if higher than second order BDF methods are used, the error term reduces the overall accuracy to $O(\Delta t^2)$. The authors of the motivating paper [17] present an extension of this splitting approach that yields higher-order accurate methods. In this thesis, we follow this approach.

Consider a pair of wavenumbers $(\xi_j, \xi_k)$ for fixed $j$ and $k$. As in previous chapters, let $A_1 := \xi_j \alpha_1$, $B_1 := \xi_k \beta_1$, $A_2 := -\xi_j^2 \alpha_2$, $B_2 := -\xi_k^2 \beta_2$ and $F := -\xi_j \xi_k \gamma$ denote the scaled eigenvalues. By expressing each eigenvalue, equation (3.14) has the form

$$
\hat{u}^{n+1} - \sum_{k=0}^{s-1} \hat{u}^{n-k} = iA_1 \hat{u}^{n+1} + iB_1 \hat{u}^{n+1} + A_2 \hat{u}^{n+1} + B_2 \hat{u}^{n+1} + F \hat{u}^{n+1}. \quad (3.16)
$$

Moreover, let $P_s(\hat{u}^n)$ be the interpolating polynomial though $\hat{u}^n$. The first modification to (3.16) is the extrapolation of the mixed term $F \hat{u}^{n+1}$ to $FP_s(\hat{u}^{n+1})$. Moreover, consider the error term given by,

$$
\eta_{\Delta t} P_{s-1}(\hat{u}^{n+1}) \quad (3.17)
$$
where $\eta_{\Delta t}$ is defined as the combination of the scaled eigenvalues

$$
\eta_{\Delta t} := -(b\Delta t)^2 A_1 B_1 + i(b\Delta t)^2 A_1 B_2 + i(b\Delta t)^2 A_1 A_2 + i(b\Delta t)^2 B_1 B_2 + (b\Delta t)^2 B_2 A_2 - (b\Delta t)^3 A_1 A_2 B_2 - (b\Delta t)^3 A_1 B_1 B_2 + i(b\Delta t)^3 A_2 B_1 B_2 - (b\Delta t)^4 A_1 A_2 B_2.
$$

In particular, the second-order error term is defined as $\eta_{\Delta t} (\hat{u}^{n+1} - \hat{u}^n)$. This is the term that gives rise to the Douglas-Gunn splitting, [16]. Adding (3.17) to (3.16) yields

$$
\frac{\hat{u}^{n+1} - \sum_{k=0}^{s-1} a_k \hat{u}^{n-k}}{\Delta t} = iA_1 \hat{u}^{n+1} + iA_2 \hat{u}^{n+1} + B_1 \hat{u}^{n+1} + B_2 \hat{u}^{n+1} + bFP_{s-1}(\hat{u}^{n+1}) + \eta_{\Delta t} P_{s-1}(\hat{u}^{n+1}).
$$

After performing a few algebraic manipulations, we can get a split FDE from the original FDE,

$$
(1 - ib\Delta t A_1) \hat{u}^* = \sum_{k=0}^{s-1} a_k \hat{u}^{n-k} + bFP_{s-1}(\hat{u}^{n+1})
$$

$$
(1 - b\Delta t A_2) \hat{u}^{**} = \hat{u}^*
$$

$$
(1 - ib\Delta t B_1) \hat{u}^{***} = \hat{u}^{**}
$$

$$
(1 - b\Delta t B_2) \hat{u}^{n+1} = \hat{u}^{***}.
$$

We use the split FDE (3.19) to perform line 5 of algorithm 2. This gives rise to the BDF-ADI spectral collocation method. An implementation in Python of this method can be found in Appendix B.

The split FDE (3.19) is equivalent to the FDE (3.18). Therefore, both expressions have the same stability properties. The stability analysis of both FDE is done using the stability polynomial.
\( P(\zeta; A_1, B_1, A_2, B_2, F) := (1 - i\delta t A_1)(1 - i\delta t B_1)(1 - \delta t A_2)(1 - \delta t B_2)\zeta^s = \)
\[
\sum_{k=0}^{s-1} a_k \zeta^{n-k} + F P_s(\zeta^s) + \eta_{\delta t,s} P_{s-1}(\zeta^s)
\]
(3.20)
Chapter 4

Main Results

The topic of this thesis was motivated by the theoretical results and numerical experiments done in [17, 1]. These references study a kind of stability weaker than absolute stability. A detailed description of this notion may be found in [9]. We will refer to this type of stability as stability in the weak sense. Indeed, if $h$ is the grid size, we say a method is unconditionally stable in the weak sense within some region $\Lambda := \{(h, \Delta t) : h > 0, \Delta t > 0\}$ if for any final time $T$ and all $(h, \Delta t) \in \Lambda$ the following estimate holds,

$$|U^{(n)}| \leq C_T \sum_{j=0}^{s-1} |U^{(j)}|$$

(4.1)

for all $n\Delta t = T$, where $C_T$ is a constant that depends only in the final time $T$ and $s$ is the number of initial conditions. The method can be called quasi-unconditionally stable in the weak sense if the bound (4.1) holds only for all $h < M_h$ and all $\Delta t < M_t$, where $M_h$ and $M_t$ are positive constants.

In these two references, proofs of conditional and unconditional stability in the weak sense are given for the hyperbolic equation and parabolic equation with the mixed term. Firstly, reference [1] considers the BDF2-ADI scheme. Unconditional stability in the weak sense is proven for these two equations, on a periodic setting. Furthermore, this reference also studies the non-periodic setting. A proof of unconditional stability is given for the parabolic case. However, this result cannot be extended to the hyperbolic equation. Higher order BDF methods (non-ADI) are con-
sidered after that. Proofs of quasi-unconditional stability are given for the periodic advection-diffusion equation, for orders of accuracy $s = 2, 3, 4, 5, 6$. In the final part of [1], some numerical experiments suggest that the BDF-ADI scheme enjoys quasi-unconditional stability for the compressible Navier-Stokes equation. This was the primary motivation to investigate the absolute stability of this scheme.

In this thesis, we extend the BDF-ADI stability analysis to absolute stability. The results obtained support the theory that BDF-ADI methods enjoy some stability properties for nonlinear problems. We prove the unconditional stability of BDF2-ADI for hyperbolic and parabolic cases, which directly improve the results in [1]. No stability analysis for higher order BDF-ADI schemes is done in this reference. In this thesis, we present absolute stability regions for BDF3-ADI and BDF4-ADI for hyperbolic and parabolic cases. Due to similar issues to the ones described at the end of chapter 2, we have not been able to produce any such stability regions for the advection-diffusion equation, and for BDF5-ADI and BDF6-ADI methods.

Most of the proofs would take a long time to write down by hand. Nevertheless, this section starts by showing the details of the proof of the unconditional stability of BDF2-ADI for the parabolic equation.

### 4.1 Stability for BDF2-ADI

In this section, we present the stability results for the BDF2-ADI method. The first result to be proven is unconditional stability for the parabolic case with a mixed term. Afterwards, we use the Schur-Cohn polynomial reduction Algorithm [1] described at the end of Section 2.1 to prove unconditional stability for the hyperbolic problem. These two results yield a conjecture of unconditional stability for the mixed problem (1.1).
4.1.1 Absolute stability of BDF2-ADI for the parabolic case with mixed term

Consider the constant coefficient parabolic equation

\[ u_t = \alpha_2 u_{xx} + \beta_2 u_{yy} + \gamma u_{xy} \quad (4.2) \]

in the parabolic region \( \alpha_2 \geq 0, \beta_2 \geq 0 \) and \( \gamma^2 < 4\alpha_2\beta_2 \). For the remaining scope of this thesis, we consider each direction with the same number of wavenumbers. Recall the definition of the scaled eigenvalues

\[
A_2 = -\Delta t \alpha_2 \xi_j^2 \leq 0, \quad B_2 = -\Delta t \beta_2 \xi_k^2 \leq 0, \quad F = -\Delta t \gamma \xi_j \xi_k. \quad (4.3)
\]

By (3.20), the stability polynomial of BDF2-ADI applied to the problem (4.2) is given by

\[
P(\zeta; A_2, B_2, F) := \left( \frac{4A_2B_2}{9} - \frac{2A_2}{3} - \frac{2B_2}{3} + 1 \right) \zeta^2 + \left( -\frac{4F}{3} - \frac{4A_2B_2}{9} - \frac{4}{3} \right) \zeta + \frac{1}{3} + \frac{2F}{3}.
\quad (4.4)
\]

**Theorem 1.** Define \( A_2, B_2, F \) as in (4.3). The stability region for the BDF2-ADI method applied to (4.2) is given by

\[
\mathcal{R} := \{(A_2, B_2, F) \in \mathbb{R}^3 : A_2, B_2 \leq 0 \text{ and } F^2 < 4A_2B_2\}.
\]

**Proof.** As described in Chapter 2, the Schur-Cohn polynomial reduction algorithm assumes non-zero constant coefficients. Therefore, the first part of the proof concerns the particular case when \( F = -\frac{1}{2} \). In this case, the stability polynomial takes the
the form
\[ P(\zeta) = \left( \frac{4A_2B_2}{9} - \frac{2A_2}{3} - \frac{2B_2}{3} + 1 \right) \zeta^2 + \left( -\frac{4F}{3} - \frac{4A_2B_2}{9} - \frac{4}{3} \right) \zeta, \]

whose roots are given by \( \zeta_1 = 0 \) and \( \zeta_2 = \frac{6 + 4A_2B_2}{(-3 + 2A_2)(-3 + 2B_2)}. \) The reader can verify this is inside the unit disk for \( A_2, B_2 \leq 0 \) and \( 1/4 < 4A_2B_2. \)

Assume \( F \neq -\frac{1}{2}. \) Note that \(-2A_2 - 2B_2 - \frac{4\sqrt{A_2B_2}}{3} \geq 0\) holds for every \( A_2, B_2 \leq 0. \)

Using \( F < 2\sqrt{A_2B_2}, \) and equivalently \(-2\sqrt{A_2B_2} < -F, \) we have

\[ a_2 - a_0 = \frac{4A_2B_2}{9} - \frac{2A_2}{3} - \frac{2B_2}{3} + 1 - \frac{1}{3} - \frac{2F}{3} > F^2 + \frac{2}{3} + 2\sqrt{A_2B_2} > 0. \]

Moreover, consider the reduced polynomial given by the formula in (2.9)

\[ P_1(\zeta) = \left( -\frac{4F^2}{9} - \frac{4F}{9} + \frac{16A_2^2B_2^2}{81} + \frac{16A_2^2B_2^2}{27} + \frac{4B_2^2}{9} + \frac{16A_2^2B_2^2}{9} - \frac{4}{3} \right) \zeta + \left( + \frac{8F^2}{9} + \frac{8FA_2B_2}{27} + \frac{8FA_2B_2}{9} + \frac{8FB_2}{9} + \frac{16A_2^2B_2^2}{81} + \frac{8A_2^2B_2^2}{27} \right) \]

\[ + \left( + \frac{8A_2B_2}{27} + \frac{8A_2B_2}{9} + \frac{8A_2}{9} + \frac{8B_2}{9} - \frac{8}{9} \right). \]

(4.5)

which has a root given by \( \zeta_0 = \frac{6 + 6F + 2A_2B_2}{6 + 3F - 3A_2 - 3B_2 + 2A_2B_2}. \) Once again, note that \(-A_2 - B_2 \geq 2\sqrt{A_2B_2} \) holds for every \( A_2, B_2 \leq 0. \) Therefore, \(-A_2 - B_2 \geq 2\sqrt{A_2B_2} \geq F. \) Hence, \( |\zeta_0| \leq 1. \)

By the Schur-Cohn polynomial reduction algorithm, the roots of the characteristic polynomial are inside the unit disk.

We would like to point out that proving the previous inequality \(|a_2| < |a_0|\) is
something that can be done in 0.002367 seconds using the `Reduce` command in *Mathematica*. For most humans, it takes longer to do it by hand.

The proof for Theorem 1 is the first as well as the last proof in this thesis that is not a computational proof. This proof describes the main steps of the Schur-Cohn polynomial reduction algorithm explicitly as in Algorithm 1 and Appendix A. For the hyperbolic case and higher order methods, the expressions of an explicit proof such as the one given for Theorem 1 grow in complexity. Therefore, we will refrain from writing these proofs explicitly and instead we shall refer to the corresponding Schur-Cohn polynomial reduction algorithm.

**Corollary 1.** The BDF2-ADI method for solving the parabolic equation (4.2) is absolutely stable for all \( \alpha_2 \) and \( \beta_2 \) non-negative real constants and \( \gamma \) any real constant such that \( \gamma^2 < 4\alpha_2\beta_2 \).

### 4.1.2 Absolute stability of BDF2-ADI for the hyperbolic case

Consider the constant coefficient hyperbolic equation

\[
    u_t = \alpha_1 u_x + \beta_1 u_y,
\]

(4.6)

with \( \alpha_1 \) and \( \beta_1 \) real numbers. Recall the definition of the scaled eigenvalues

\[
    A_1 = \Delta t\alpha_1 \xi_j, \quad B_1 = \Delta t\beta_1 \xi_k.
\]

(4.7)

The form of the stability polynomial (3.20) for the BDF3-ADI method applied to the hyperbolic case is given by,

\[
    P(\zeta; A_1, B_1) := \left( -\frac{4AB}{9} - \frac{2iA}{3} - \frac{2iB}{3} + 1 \right) \zeta^2 + \left( \frac{4AB}{9} - \frac{4}{3} \right) \zeta + \frac{1}{3}.
\]

(4.8)

**Theorem 2.** Define \( A_1 \) and \( B_1 \) as in (4.7). The stability region for the BDF2-ADI
method applied to the hyperbolic problem (4.6) is given by,

\[ R := \{(A_1, B_1) \in \mathbb{R}^2\}. \]

Proof. For the case where \( A_1 \neq -B_1 \) the proof is given by the code in Figure A.1. in Appendix A.

In case \( A_1 = -B_1 \) yields the stability polynomial \( P(\zeta; A_1) = \left( \frac{4A_1^2}{9} + 1 \right) \zeta^2 + \left( -\frac{4A_1^2}{9} - \frac{4}{3} \right) \zeta + \frac{1}{3} \), with roots \( \zeta_0 = 1 \) and \( \zeta_1 = \frac{3}{9+2A_1^2} < 1 \).

Corollary 2. The BDF2-ADI method for solving the hyperbolic equation is absolutely stable for all \( \alpha_1 \) and \( \beta_1 \) real constants.

4.1.3 Absolute stability of BDF2-ADI for the mixed case

The Schur-Cohn polynomial reduction algorithm to prove Theorems 1 and 2 is relatively easy to implement in both previous cases. However, as it has already been noted at the end of Section 2.1 we were not able to find an explicit stability region for the mixed equation. Therefore, we present unconditional stability for the advection-diffusion equation as a conjecture. Numerical experiments in the next chapter support this conjecture.

Conjecture 1. The BDF2-ADI method for solving the mixed equation is absolutely stable for all \( \alpha_1, \alpha_2, \beta_1, \beta_2 \) and \( \gamma \) as in (1.1).

We now move to higher-order methods. Recall that the Douglas-Gunn splitting as in [16] does not refer to any splitting with accuracy greater than three. As described in Chapter 3, the motivating papers [11,17] use the error term (3.17) to achieve higher-order splitting. In the next sections, we follow this approach.
4.2 Stability of BDF3-ADI

We start by studying the pure parabolic problem, i.e., the model (4.2) with $\gamma = 0$. The BDF3-ADI yields the stability polynomial,

$$P(\zeta; A_2, B_2) := (1 - \Delta t A_2)(1 - \Delta t B_2) \zeta^3 + \frac{18}{11} \zeta^2 - \frac{9}{11} \zeta + \frac{2}{11} + \eta \Delta t (2\zeta - 1).$$

(4.9)

The BDF3-ADI method for the pure parabolic problem is unconditionally stable in the sense described in Subsection 3.2.2. The case with the mixed term yields a 3-dimensional region of stability for $(A_2, B_2, F)$ and it is left for the end of this chapter.

**Theorem 3.** Define $A_2, B_2$ as in (4.3). The stability region for the BDF3-ADI method applied to (4.2) with $\gamma = 0$ is given by

$$\mathcal{R} := \{(A_2, B_2) \in \mathbb{R}^2 : A_2, B_2 \leq 0\}.$$  

**Proof.** The proof for this theorem is similar to the one presented in Figure A.1. in Appendix A. The region obtained by the Reduce command is $\mathbb{R}^2$. \qed

**Corollary 3.** The BDF3-ADI method for solving the pure parabolic equation is absolutely stable for all $\alpha_2$ and $\beta_2$ non-negative real constants.

Recall the real-valued stability regions illustrated in Figure 3.2. The equivalent region of stability of BDF3 methods for the pure parabolic problem is the $\mathbb{R}^2$ plane. This is because the left-half real axis is contained within the complex-valued stability region of BDF3. The previous theorem implies the same is verified for BDF3-ADI methods.

Now, consider the hyperbolic problem. Figure 4.1 shows the stability region for this case. The hyperbolic problem is unstable for all possible choices of time steps.
To understand why we will illustrate the concept of conditional stability in the next section.

4.3 Stability of BDF4-ADI

Once again, consider the pure parabolic problem. The BDF4-ADI yields the stability polynomial

\[
P(\zeta; A_2, B_2) := (1 - \Delta t A_2) (1 - \Delta t B_2) \zeta^4 + \frac{48}{25} \zeta^3 - \frac{36}{25} \zeta^2 \\
+ \frac{16}{25} \zeta - \frac{3}{25} + \eta_{\Delta t,3} (3\zeta^3 - 3\zeta^2 + 1)
\]  
(4.10)

Theorem 4. Define \( A_2, B_2 \) as in (4.3). The stability region for the BDF4-ADI method applied to (4.2), with \( \gamma = 0 \), is given by Figure 4.3(b).

Proof. The proof for this theorem is the code given Figure A.1. in Appendix A, with \( F = 0 \).
Corollary 4. The BDF4-ADI method for solving the pure parabolic equation is absolutely stable for all \( \alpha_2 \) and \( \beta_2 \) non-negative real constants and \( \Delta t \) small enough.

The last result differs from Corollary 3. In Corollary 4 we must take \( \Delta t \) small enough to have convergence. This is called conditional stability, and it will be illustrated below.

We have seen in Chapter 3 that the numerical solution is given by (3.9). This means we need to solve \( m^2 \) ODE of the type

\[
\hat{u}'_{\xi_j,\xi_k} = \lambda \hat{u}_{\xi_j,\xi_k},
\]

corresponding to the point \((j, k)\). If the numerical solution of at least one of these ODE is unstable, then the sum (3.9) will blow up. This means that we need to select a set of scaled eigenvalues such that all the stability polynomials corresponding to all such ODE have roots inside the stability region. For illustration, consider the diffusion problem

\[
u_t = u_{xx} + u_{yy},
\]
defined in the periodic interval \([-\pi/2, \pi/2]^2\). Fix a spatial grid of \( m = 64 \) collocation points in each direction. This means, the only parameter in the definition of \( A_1 \) and \( B_1 \) that is not fixed is \( \Delta t \). Figure 4.3 illustrates conditional stability. Indeed, if \( \Delta t = 0.0015 \) there will be coefficients of the (3.9) that will blow up, for instance, the mode corresponding to \((\xi_j, \xi_k) = (64, 64)\) as in figure 4.2(a). However, if the time step is small enough, \( \Delta t = 0.001 \) then all scaled eigenvalues lie inside the stability region, which means the numerical solution will be stable, as in Figure 4.2(b).

The stability region of BDF4-ADI for the hyperbolic problem is given by Figure 4.3(a). Note that the shape of this stability region is similar to the shape of Figure 4.1 however, this region is smaller. We can consider a simple example and plot the
(a) If \( \Delta t = 0.0015 \) the resulting numerical solution is unstable.

(b) If \( \Delta t = 0.001 \) the resulting numerical solution is stable.

Figure 4.2: Illustration of conditional stability of BDF4-ADI for the pure parabolic problem.

(a) Stability region of BDF4-ADI for hyperbolic Problem
(b) Stability region of BDF4-ADI for pure parabolic problem

Figure 4.3: Conditional stability of BDF4-ADI for the pure parabolic problem

lattice of modes over the hyperbolic stability region.

Consider the advection equation \( u_t = u_x + u_y \), on the same grid as the previous example and using \( \Delta t = 0.1 \). The lattice of scaled eigenvalues over the stability region for BDF3-ADI is represented in Figure 4.1. No matter how small \( \Delta t \) is, any numerical solution computed by the BDF3-ADI scheme will always be unstable. The
same lattice can be represented together with the stability region of BDF4-ADI, and the same issue arises. Hence, the BDF4-ADI scheme yields an unstable numerical solution for the hyperbolic equation.

4.4 Stability of BDF3-ADI and BDF4-ADI for the parabolic equation case with mixed term

In this section, we will state and illustrate the results on the conditional stability of BDF3-ADI and BDF4-ADI for the full problem (4.2). Moreover, we shall present absolute stability regions for both schemes.

We start with the BDF3-ADI method. The stability polynomial is given by

\[
P(\zeta) := (1 - \Delta t A_2)(1 - \Delta t B_2)\zeta^3 + \frac{18}{11}\zeta^2 - \frac{9}{11}\zeta + \frac{2}{11} - F(3\zeta^2 - 3\zeta + 1) + \Delta t (2\zeta - 1).
\]  

(4.11)

\[P(\zeta) := (1 - \Delta t A_2)(1 - \Delta t B_2)\zeta^3 + \frac{18}{11}\zeta^2 - \frac{9}{11}\zeta + \frac{2}{11} - F(3\zeta^2 - 3\zeta + 1) + \Delta t (2\zeta - 1).\]

\[
\text{Theorem 5. Define } A_2, B_2 \text{ and } F \text{ as in (4.3). The stability region of the BDF3-ADI method for problem (4.2) is given by Figure 4.4.}
\]

Proof. The proof of this theorem is given by a similar code to the one in Figure A.1. in Appendix A, with the difference that, instead of three, we only need to compute two reduced polynomials.

Because we are considering three parameters, the stability region will be three-dimensional. This makes it nontrivial to find the maximum stable time step size. However, we will present a method to find this critical time step, by investigating a particular two-dimensional slice of this three-dimensional stability region. It is important to note that this method may fail, and such cases will be discussed. Nevertheless, this approach has been found to be an efficient way to find a stable set of scaled eigenvalues for (4.2).
Figure 4.4: Stability region for \((A_2, B_2, F)\) of BDF3-ADI for the problem (4.2). This region is symmetric with respect to the plane \(A_2 = B_2\). However, it is not symmetric with respect to \(F = 0\).

Similarly to the two-dimensional case, we need to fit a lattice of scaled eigenvalues inside this stability region to have a stable numerical solution for the parabolic equation. This lattice, of course, depends on the problem to be considered. As an example of this process, we will consider the problem

\[ u_t = u_{xx} + u_{yy} + u_{xy} \]  

(4.12)

with the same spatial grid specifications and initial condition as in the examples of the previous section. A lattice of modes for this problem is illustrated in Figure 4.5. A possible approach to verify whether this time step will give a stable numerical solution is to investigate whether each two-dimensional parabola is contained inside the corresponding slice of the stability region. For 64 collocation points, this would yield an algorithm that verifies this inclusion for 64 parabolas. This process would be a very tedious one, and we suggest a different approach. The technique is based on the simple observation: The amplitude of the lattice can be controlled by one of the
Figure 4.5: Lattice for the parabolic problem. This lattice corresponds to the scaled eigenvalues $A_2, B_2$ and $F$ with $\alpha_2 = \beta_2 = \gamma = 1$ and $\Delta t = 0.001$. Each direction has 64 collocation points.

outer parabolas, say, the one represented in purple. Indeed, if this particular parabola fits inside the corresponding slice of the stability region, then it is likely that all other parabolas will be included in the stability region. This technique may fail, and one such situation will be discussed. Nonetheless, if the outer parabola is well inside the corresponding slice of the stability region, this will be less likely to happen.

Figure 4.6(a) shows that the full three-dimensional lattice seems to be inside the stability region. We investigate this stability region more carefully in Figure 4.6(b). This parabola is well inside the corresponding slice of the stability region. We shall see numerical experiments that show this is indeed a stable numerical solution.

As stated previously, this technique might fail for some cases. Consider the following problem

$$u_t = 1.25u_{xx} + 1.375u_{yy} - 2u_{xy} \tag{4.13}$$

with $\Delta t = 0.002$.

The outer parabola and the corresponding slice of the stability region are represented in Figure 4.7. Looking at Figure 4.7(a), the scheme seems to be stable for the particular problem. However, the numerical experiments show that this is an unstable solution. Figure 4.7(b) shows the parabola and the corresponding slice for
Figure 4.6: This is a good indication that this choice of parameters yields a stable solution for $\Delta t \leq 0.001$.

The eigenvalue with wavenumber $\xi_{21} = 21$. We can verify that the solution is unstable because there are unstable scaled eigenvalues.

Figure 4.7: Slices of the stability region of BDF3-ADI for the parabolic problem (4.13). This scheme yields an unstable solution for $\Delta t = 0.002$. 
The story for BDF4-ADI is dramatically less stable, however, the analysis of the corresponding stability region is very similar to BDF3-ADI.

**Theorem 6.** Define $A_2, B_2$ and $F$ as in (4.3). The stability region for the BDF4-ADI method applied to (4.2) is given by Figure 4.8.

**Proof.** The proof of this theorem is given in Figure A.1. in Appendix A.

The stability properties of BDF4-ADI are worse than those of BDF3-ADI, as it usually requires a smaller time step to find a stable solution for the same problem. In addition to this, Figure 4.2 takes considerably longer to compute than the one for BDF3-ADI. A *Mathematica* code that calculates this specific region is presented in Appendix A. Similarly to BDF3-ADI method, the purple parabolic technique may be used to find stable examples, however, with a considerable amount of extra labour. Consider the problem,

$$u_t = u_{xx} + u_{yy} + \frac{1}{10}u_{xy}.$$  \hspace{1cm} (4.14)

Choose $\Delta t = 0.001$ and similar plots in Figure 4.6 can be seen in Figure 4.9. This
yields a stable solution. From Figure 4.9(b) we see that the corresponding slice of the stability region contains very little of the positive part of $F$. This implies that a minimal change in $\Delta t$ may yield an unstable solution. Figure 4.10 illustrates this. Numerical implementations of both these situations can be found in the next chapter.

(a) Lattice of points inside BDF4-ADI stability region.  

(b) Slice from the stability region corresponding to the outer parabola. Note that the region corresponding to the positive part of $F$ decreases as $A_2$ increases. This means that a very small time step must be used.

Figure 4.9: Stable solution of the BDF4-ADI applied to the parabolic problem (4.14).

Once again, the only possible way to guarantee stability is to verify that all two-dimensional parabolas lie inside the stability region. Nevertheless, investigating the outer parabola has demonstrated to be an effective technique to study stability properties.
Figure 4.10: The solution of BDF4-ADI for (4.14) becomes unstable if instead we consider $\Delta t = 0.0012$. 
Chapter 5

Numerical Implementations

In this last chapter, we will compare the previous theoretical results with numerical experiments. The previous stability regions were obtained with Mathematica and the numerical experiments mostly with numpy package from Python. Different convergence tests, as well as numerical examples, will be presented for the different problem types and numerical schemes.

Consider the spatial grid given by \((x, y) \in [-\frac{\pi}{2}, \frac{\pi}{2}]^2\). Discretize this grid using \(64 \times 64\) collocation points evenly spaced. For all these numerical experiments we consider the initial condition \(u_0(x, y) = e^{-10(x^2+y^2)}\). The fast decay of this function allows us to consider periodic boundary conditions.

To compute the error of a numerical solution to (1.1) by the BDF-ADI spectral collocation method we need to find the exact solution of this PDE. However, for arbitrary initial data, we cannot find such solution explicitly. Since the problem is linear with constant coefficients, we can easily find a very accurate numerical solution to (1.1) using a spectral collocation discretization in space and exact integration in time. In fact, if \(\Delta t\) and \(h\) have the same order of magnitude, this solution is much more accurate than the one obtained using the BDF-ADI spectral collocation method. Therefore, for these simulations, we will call the numerical solution to (1.1) obtained by exact integration in time the reference solution. We can, therefore, perform convergence tests for all our schemes. We measure the errors in the \(L_2\)-norm

\[
\|e\|_2 = (\Delta x \Delta y \sum_{j,k} (U_{j,k}^N - u(T, x_j, y_k))^2)^{1/2}.
\]


5.1 Unconditional stability

The first numerical experiments to be presented are the cases of unconditional stability. Recall the notion of unconditional stability in Section 3.2.2. We have shown that the BDF2-ADI method is unconditionally stable for the parabolic and hyperbolic cases. These two results lead to a conjecture on unconditional stability for the mixed problem has also been presented. The first experiments support Conjecture 1. Furthermore, Theorem 3 is also illustrated with an example.

Consider the BDF2-ADI spectral collocation method applied to the mixed problem

\[ u_t = u_x + u_y + u_{xx} + u_{yy} + u_{xy}. \]  

Figure 5.1 shows a convergence test for a solution computed at time \( T = 0.5 \). Indeed, this shows second order convergence.

Other choices of time steps or parameters can be taken and BDF2-ADI will still converge. This is by far the most well-behaved scheme.

Now, we move to BDF3-ADI methods. Consider the diffusion problem

\[ u_t = u_{xx} + u_{yy}. \]  

Figure 5.2 shows a numerical solution of the problem (5.2). The third order conver-
gence can be verified in Figure 5.3. Again, other choices of time steps or parameters can be taken and BDF3-ADI will still converge for this particular problem.

(a) Numerical solution of (5.2) at $T = 0.2$ using $\Delta t = 0.001$. The corresponding error is of the order $10^{-8}$.

(b) Numerical solution of (5.2) at $T = 0.9$ using $\Delta t = 0.001$. The corresponding error is $10^{-10}$.

Figure 5.2: Numerical solution of BDF3-ADI for problem (5.2).

Figure 5.3: Convergence test of BDF3-ADI for the problem (5.2). The final time is $T = 0.5$. 
5.2 Conditional Stability

A method applied to a particular problem is conditionally stable if for any \( \Delta t \) small enough, the numerical solution converges to the reference solution. This section concerns the cases for this kind of stability. The first such result is Theorem 4 that states conditional stability of the BDF4-ADI problem applied to the equation (4.2) with \( \gamma = 0 \), also called as diffusion equation. The other two results of conditional stability are Theorem 5 and Theorem 6. These results refer to the BDF3-ADI and BDF4-ADI applied to the parabolic equation (4.2).

5.2.1 BDF4-ADI for the diffusion equation

Consider the BDF4-ADI applied to problem (5.2), as well as its related Figure 4.2. A stable numerical solution is plotted in Figure 5.4.

(a) Numerical solution (5.2) at \( T = 0.2 \) using \( \Delta t = 0.001 \). The corresponding error is \( 10^{-10} \).

(b) Numerical solution of (5.2) at \( T = 0.9 \) using \( \Delta t = 0.001 \). The corresponding error is \( 10^{-13} \).

Figure 5.4: Stable numerical solution of BDF4-ADI for problem (5.2).

The numerical solutions represented here are very close to the ones for BDF3-ADI. Note that, for \( T = 0.9 \) the error against the true solution is smaller for BDF4-ADI, than the one for BDF3-ADI in Figure 5.2(b), as expected. Finally, Figure 5.5 shows a convergence test for the solution at time \( T = 0.5 \). We cannot take as small time steps in this case, like those taken in Figure 5.3. Indeed, as the convergence of BDF4-ADI
Figure 5.5: Convergence test of BDF4-ADI for the problem (5.2). The final time is $T = 0.5$.

is faster than the convergence of BDF3-ADI, smaller time steps yield floating point errors that create instabilities in the solution.

Figure 4.2 illustrates an example of an unstable solution. This solution is plotted in Figure 5.6 for different times. Indeed, we have seen that the mode corresponding to the wavenumber pair $(64, 64)$ is outside the stability region. This means that, as the magnitude of the unstable mode is large, the error starts to increase shortly in time. The solution for time $T = 0.9$ is shown in Figure 5.6(a). The error is still relatively small, of order $10^{-11}$. However, soon the solution is evidently unstable, as Figure 5.6(b) suggests.

(a) Numerical solution (5.2) at $T = 0.9$ using $\Delta t = 0.0015$. The corresponding error is $10^{-11}$.

(b) Numerical solution of (5.2) at $T = 1.5$ using $\Delta t = 0.0015$. The corresponding error is $10^{-5}$.

Figure 5.6: Unstable numerical solution of BDF4-ADI for problem (5.2). The numerical solution at $T = 1.5$ corresponding to $\Delta t = 0.001$ has an error of $10^{-13}$. 
5.2.2 Parabolic case with mixed term

In the remaining part of this chapter, we present the numerical simulations of the results for the parabolic equation (4.2).

We start with BDF3-ADI. Consider the example illustrated in Figure 4.6 and the parabolic problem (4.12). A numerical solution as well as a convergence test are given in the Figures 5.7 and 5.8.

(a) Numerical solution at $T = 0.2$ using $\Delta t = 0.01$. The corresponding error is $10^{-5}$.

(b) Numerical solution at $T = 0.9$ using $\Delta t = 0.01$. The corresponding error is $10^{-7}$.

Figure 5.7: Stable numerical solution of BDF4-ADI for the parabolic problem (4.12).

We close the BDF3-ADI experiments with the unstable solution for the problem (4.13) illustrated in Figure 4.7. Note that, since the modes outside the stability region are relatively large, the instability is seen straight away, Figure 5.9.

Figure 5.8: Convergence test of BDF3-ADI for the parabolic problem (4.12). The final time is $T = 0.5$. 
(a) Numerical solution at $T = 0.2$ using $\Delta t = 0.01$. The corresponding error is $10^{-4}$.

(b) Numerical solution at $T = 0.5$ using $\Delta t = 0.01$. The corresponding error $10^{-2}$.

Figure 5.9: Unstable numerical solution of BDF3-ADI for the parabolic problem (4.13).

Finally, we arrive at the last set of simulations. Consider the BDF4-ADI method applied to the problem (4.14). The numerical solutions are illustrated in Figure 5.10. The top plots show a stable solution, whereas the bottom figures illustrate the unstable time step, and the corresponding unstable numerical solution. The numerical solution for the unstable time step takes some time to blow up. This means the unstable eigenvalues are slightly greater than one, and the corresponding unstable coefficients of (3.9) take some time to go off. Last but not the least, the stable solution converges in fourth order, as can be seen in Figure 5.11.
(a) Stable numerical solution at time $T = 0.2$. The error is $10^{-10}$.

(b) Stable numerical solution at time $T = 0.9$. The error is $10^{-13}$.

(c) Unstable solution at time $T = 0.9$. The error is $10^{-12}$.

(d) Unstable solution at time $T = 3$. The error is $10^{-7}$.

Figure 5.10: Numerical solutions for problem (4.14). A slight change of the time step can change the behaviour of the numerical solution drastically. The numerical solution is stable for $\Delta t = 0.001$ and unstable for $\Delta t = 0.0012$. The numerical solution at $T = 3$ corresponding to $\Delta t = 0.001$ has an error of $10^{-16}$.

Figure 5.11: Convergence of BDF4-ADI for the parabolic equation with the mixed term. This example refers to (4.14). For $\Delta t < 10^{-4}$ the floating point errors originate instabilities.
Chapter 6

Concluding Remarks

The purpose of this thesis is to study the absolute stability of the BDF-ADI scheme as a time discretization for the mixed equation (1.1). In summary, we can expect unconditional stability for BDF2-ADI and, at most, conditional stability for higher-order schemes.

As Figure 3.2 illustrates, the negative part of the real-axis is contained in the stability region of BDF4, i.e., BDF4 is unconditionally stable for the diffusion problem. A surprising result is the conditional stability of BDF4-ADI applied to the diffusion problem. This means, in particular, that there is a maximal stable time step for BDF4-ADI, in contrast to BDF4.

The unconditional stability for BDF2-ADI is conjectured because we could not find any explicit region of stability for it. Again, this is an issue that is extended to higher-order methods. However, the conditional stability for BDF3-ADI and BDF4-ADI can be investigated with a numerical experiment. Since there is no stability region, no precise way was used to generate the next example. The example below was generated by adding two hyperbolic terms to the stable parabolic example (4.14), with $\Delta t = 0.001$.

Consider

$$u_t = u_x + u_y + u_{xx} + u_{yy} + \gamma u_{xy}$$

with $\gamma = 0.1$ and the usual initial condition.
Figure 6.1 shows a numerical solution and a convergence test for both methods. Once again, BDF4-ADI is more accurate, as expected. Moreover, as in the parabolic case, the maximum stable time step size is smaller for BDF4-ADI than the one for BDF3-ADI.

(a) Numerical solution at time $T = 0.2$ using BDF3-ADI. The error is $10^{-8}$.

(b) Numerical solution at time $T = 0.7$ using BDF3-ADI. The error is $10^{-10}$.

(c) Numerical solution at time $T = 0.2$ using BDF4-ADI. The error is $10^{-10}$.

(d) Numerical solution at time $T = 0.7$ using BDF4-ADI. The error is $10^{-12}$.

(e) Convergence test for BDF3-ADI at time $T = 0.5$.

(f) Convergence test for BDF4-ADI at time $T = 0.5$.

Figure 6.1: We can conjecture conditional stability of BDF3-ADI and BDF4-ADI for the mixed problem.

Many ideas for future work follow from this investigation. It would be interest-
ing to do the corresponding analysis for BDF5-ADI and BDF6-ADI, although we should not expect more than conditional stability. The final application to BDF-ADI methods is on nonlinear problems, such as in [1]. Spatial discretizations other than spectral methods may be more suitable for some nonlinear models. Therefore, other spatial discretizations or nonlinear solvers can be studied along with BDF-ADI time schemes. Furthermore, it would also be interesting to do a stability analysis similar to the one done in this thesis but based on the quasi-unconditional stability notion defined in [1].
REFERENCES


APPENDICES

A Schur-Cohn polynomial reduction algorithm

Figures A.1 and A.2 illustrate two Mathematica codes that give stability regions for BDF4-ADI and BDF2-ADI applied to the parabolic and hyperbolic problems, respectively. These implementations were done in Mathematica v.11.1.10.
Figure A.1: Implementation of the Schur-Cohn polynomial reduction algorithm of BDF4-ADI for the parabolic equation (4.2). The function complexReducedPolynomial gives the reduced polynomial of any complex valued polynomial. The variables $G$ and $H$ correspond to $A_2$ and $B_2$, respectively.
Figure A.2: Proof of Theorem 2. Since the coefficients of the stability polynomial for the hyperbolic problem are complex-valued, the command `CoefficientList` does not work in the way it works for the parabolic problem. The stability polynomial for the parabolic problem has real-valued coefficients. The variables $A$ and $B$ correspond to $A_1$ and $B_1$, respectively.
B A Python implementation of BDF-ADI method

The function `bdf_adi` returns a numerical solution and the corresponding error in the Frobenius norm of the mixed problem (1.1) using the BDF-ADI method. The programming language is Python v3.0.

```python
import numpy as np

def bdf_adi(N, T, m, alpha_1, alpha_2, beta_1, beta_2, gamma, order, probType, L, u0):

    # N number of time steps
    # T final time
    # m number of grid points
    # alpha_1 advection coefficient for x–direction
    # alpha_2 diffusion coefficient for the x–direction
    # beta_1 advection coefficient for the y–direction
    # beta_2 diffusion coefficient for y–direction
    # gamma diffusion coefficient for xy–direction
    # order order of the method (2,3,4)
    # probType type of problem (hyp, par, mixed)
    # L half length of the interval
    # u0 initial condition (periodic)
```


# Time step

delta_t = T/N

# Spatial grid

x = np.arange(-m/2, m/2)*(L/m)
y = np.arange(-m/2, m/2)*(L/m)
X, Y = np.meshgrid(x, y)
dx = x[1] - x[0]

# Wavenumbers

xi = np.fft.fftfreq(m)*m/2*np.pi/L
yi = np.fft.fftfreq(m)*m/2*np.pi/L

# Matrix coefficients

X_advc = np.zeros((m, m), dtype=complex)  # Advection
    # in x-direction

Y_advc = np.zeros((m, m), dtype=complex)  # Advection
    # in y-direction

X_diff = np.zeros((m, m), dtype=complex)  # Diffusion
    # in x-direction
Y_diff = np.zeros((m,m), dtype = complex)  # diffusion
    # in y-direction

F    = np.zeros((m,m), dtype = complex)  # diffusion
    # in xy-direction

#Hyperbolic problem
if probType == 'hyp':
    if alpha_2 != 0 or beta_2 != 0:
        raise Exception('not a hyperbolic problem')
    for k in range(m):
        for j in range(m):
            X_advc[k,j] = 1.j*alpha_1*xi[j]
            Y_advc[k,j] = 1.j*beta_1*yi[k]

#Parabolic problem
elif probType == 'par':
    if gamma**2 > 4*alpha_2*beta_2:
        raise Exception('Outside parabolic region')
    if alpha_1 != 0 or beta_1 != 0:
        raise Exception('not a parabolic problem')
    for k in range(m):
        for j in range(m):
            X_diff[k,j] = -alpha_2*xi[j]**2
            Y_diff[k,j] = -beta_2*yi[k]**2
            F[k,j] = -gamma*xi[j]*yi[k]
if $F[k,j]^{*2} > 4 \times X_{\text{diff}}[k,j] \times Y_{\text{diff}}[k,j]$: 

raise Exception('Uppps')

# Mixed problem

elif probType == 'mixed':
    if gamma**2 > 4*alpha_2*beta_2:
        raise Exception('Outside parabolic region')
    for k in range(m):
        for j in range(m):
            X.advc[k,j] = 1.j*alpha_1*xi[j]
            X.diff[k,j] = -alpha_2*xi[j]**2
            F[k,j] = -gamma*xi[j]*yi[k]
            Y.advc[k,j] = 1.j*beta_1*yi[k]
            Y.diff[k,j] = -beta_2*yi[k]**2
    else:
        raise Exception('Problem type can only be hyp, par or mixed')

frames = [u0]  # store solutions

# Beginning of ODE setting (frequency space)

# FFT of initial condition
what_0 = np.fft.fft2(u0)
# coefficient for first initial conditions
Coeff = X_adv + X_diff + Y_adv + Y_diff + F
sol = uhat_0*np.exp(Coeff*T) # exact solution in freq space

errorTerm = lambda b: (b*delta_t)**2*X_adv*Y_adv +
(b*delta_t)**2*X_adv*Y_diff +
(b*delta_t)**2*X_diff*Y_adv +
(b*delta_t)**2*X_diff*Y_diff +
(b*delta_t)**2*X_adv*X_diff +
(b*delta_t)**2*Y_adv*Y_diff -
(b*delta_t)**3*X_adv*X_diff*Y_adv -
(b*delta_t)**3*X_adv*X_diff*Y_diff -
(b*delta_t)**3*X_diff*Y_adv*Y_diff -
(b*delta_t)**3*X_adv*Y_adv*Y_diff +
(b*delta_t)**4*X_diff*X_adv*Y_adv*Y_diff

# BDF2-ADI
if order == 2:
    # LHS BDF2 coefficient
    b = 2./3

    # one more initial condition with Explicit Euler
    uhat_1 = uhat_0 + delta_t*Coeff*uhat_0
#store solution
frames.append(np.real(np.fft.ifft2(uhat_1)))

uhat_nm3, uhat_nm2, uhat_nm1, uhat_n = 0., 0., \n uhat_0, uhat_1

#interpolating polynomial of order 1
P_sm1 = lambda uhat_n, uhat_nm1, uhat_nm2, \n uhat_nm3: uhat_n

#interpolating polynomial of order 2
P_s = lambda uhat_n, uhat_nm1, uhat_nm2, \n uhat_nm3: 2*uhat_n - uhat_nm1

#RHS of BDF2
RHS = lambda uhat_n, uhat_nm1, uhat_nm2, \n uhat_nm3: 4./3*uhat_n - 1./3*uhat_nm1

#BDF3-ADI
elif order == 3:

#LHS BDF3 coefficient
b = 6./11

#two more initial conditions with Explicit Euler
uhat_1 = uhat_0*np.exp(Coeff*delta_t)
uhat_2 = uhat_0*exp(Coeff*2*delta_t)

#store solution
frames.append(np.real(np.fft.ifft2(uhat_2)))
frames.append(np.real(np.fft.ifft2(uhat_1)))

uhat_nm3, uhat_nm2, uhat_nm1, uhat_n = 0., uhat_0, \
 uhat_1, uhat_2

#interpolating polynomial of order 1
P_sm1 = lambda uhat_n, uhat_nm1, uhat_nm2, uhat_nm3: \
 2*uhat_n - uhat_nm1

#interpolating polynomial of order 2
P_s = lambda uhat_n, uhat_nm1, uhat_nm2, uhat_nm3: \
 3*uhat_n -3*uhat_nm1 + uhat_nm2

#RHS of BDF3
RHS = lambda uhat_n, uhat_nm1, uhat_nm2, uhat_nm3: \
 18./11*uhat_n - 9./11*uhat_nm1 + 2./11*uhat_nm2

#BDF4-ADI
elif order == 4:
  #LHS BDF4 coefficient
  b = 12./25
#three more initial conditions with Explicit Euler

\[
\begin{align*}
\hat{u}_{1} & = \hat{u}_{0} \cdot \exp(Coeff \cdot \delta t) \\
\hat{u}_{2} & = \hat{u}_{0} \cdot \exp(2 \cdot Coeff \cdot \delta t) \\
\hat{u}_{3} & = \hat{u}_{0} \cdot \exp(3 \cdot Coeff \cdot \delta t)
\end{align*}
\]

#store solution

frames.append(np.real(np.fft.ifft2( \(\hat{u}_{1}\)))
frames.append(np.real(np.fft.ifft2( \(\hat{u}_{2}\)))
frames.append(np.real(np.fft.ifft2( \(\hat{u}_{3}\))))

\(\hat{u}_{n-3}, \hat{u}_{n-2}, \hat{u}_{n-1}, \hat{u}_{n} = \hat{u}_{0}, \hat{u}_{1}, \hat{u}_{2}, \hat{u}_{3}\)

#interpolating polynomial of order 2

\[
\text{P}_{sm1} = \lambda \text{uhat}\_n, \text{uhat}\_nm1, \text{uhat}\_nm2, \text{uhat}\_nm3: \ \\
3*\text{uhat}\_n - 3*\text{uhat}\_nm1 + \text{uhat}\_nm2
\]

#interpolating polynomial of order 3

\[
\text{P}_{s} = \lambda \text{uhat}\_n, \text{uhat}\_nm1, \text{uhat}\_nm2, \text{uhat}\_nm3: \ \\
4*\text{uhat}\_n - 6*\text{uhat}\_nm1 + 4*\text{uhat}\_nm2 - \text{uhat}\_nm3
\]

#RHS of BDF4

\[
\text{RHS} = \lambda \text{uhat}\_n, \text{uhat}\_nm1, \text{uhat}\_nm2, \text{uhat}\_nm3: \ \\
48./25*\text{uhat}\_n - 36./25*\text{uhat}\_nm1 + 16./25*\text{uhat}\_nm2 - \ \\
3./25*\text{uhat}\_nm3
\]
else:
    raise Exception('order of the method must be 2, 3 or 4')

#BDF–ADI system
for n in range(order-1,N):
    #explicit part of BDF
    rhs = RHS(uhat_n, uhat_nm1, uhat_nm2, uhat_nm3)

    #interpolating poly for error term
    p_sm1 = P_sm1(uhat_n, uhat_nm1, uhat_nm2, uhat_nm3)

    #interpolating poly for mixed term
    p_s = P_s(uhat_n, uhat_nm1, uhat_nm2, uhat_nm3)

    #ADI system
    u_star1 = ( rhs + errorTerm(b)*p_sm1 + \
               b*delta_t*F*p_s )/(1 - b*delta_t*X_advc)
    u_star2 = u_star1/(1 - b*delta_t*X_diff)
    u_star3 = u_star2/(1 - b*delta_t*Y_advc)
    uhat_np1 = u_star3/(1 - b*delta_t*Y_diff)

    #update solutions
    uhat_nm3 = uhat_nm2
    uhat_nm2 = uhat_nm1
    uhat_nm1 = uhat_n
uhat_n  = uhat_np1

#End of ODE setting, back to spatial domain
#

u = np.real(np.fft.ifft2(uhat_n)) #imaginary part is errors
sol = np.real(np.fft.ifft2(sol)) #exact solution
error = dx*np.linalg.norm(u-sol) #Frobenius norm

return u, error