THE UNIVERSITY OF CALGARY

An adaptive Lagrange-Galerkin method for the numerical solution of the Navier-Stokes equations

by

Andrew Taylor

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

January, 2005

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "An adaptive Lagrange-Galerkin method for the numerical solution of the Navier-Stokes equations" submitted by Andrew Taylor in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Dr. Tony Ware
Department of Mathematics and Statistics

Dr. Len Bos
Department of Mathematics and Statistics

Dr. Edward Krebes
Department of Geology and Geophysics

21/1/05
Date

Abstract

We shall examine the numerical solutions of two dimensional incompressible fluid flow problems via spectral methods. The focus will be on Galerkin methods with the aim of constructing a Lagrange-Galerkin method which is adaptive in time. A comparison will also be made between the Lagrange-Galerkin method in both its adaptive and unadaptive forms with the pseudospectral method. The pseudospectral method has been chosen since it is a widely accepted standard method for solving periodic fluid flow problems.

Since the Navier-Stokes equations, which govern the motion of incompressible fluids are nonlinear, there are often difficulties in computing the numerical solution. The main difficulty with the pseudospectral method is that it is only conditionally stable. This is the motivation for using the Lagrange-Galerkin method instead since it is unconditionally stable. So the only consideration that needs to be made for the step size is determined by how accurate the solution needs to be. The motivation for the adaptive Lagrange-Galerkin method is that it is a faster unconditionally stable method as opposed to the pseudospectral method.

Acknowledgements

I would first like to acknowledge Prof. Tony Ware for his patience and perseverence over the past number of years, and also for the little MATLAB tidbits that made programming less of a torment and more of an enjoyment.

It is my humble opinion that friends are a necessary distraction in order to complete a thesis. So thanks also go out friends, near and far, for movies, Ultimate, camping trips, games, BBQ's and most importantly, just doing what friends do.

Finally I would like to thank my family for their support, and especially my mother, as I know she would have liked to have seen my finished work.

Table of Contents

Approval Page ii

Abstract iii

Acknowledgements iv

Table of Contents v

1 Introduction 1
- 1.1 The Navier-Stokes equations . . . 1
- 1.2 Overview of spectral methods . . . 6
- 1.3 Overview of important references . . . 9
- 1.4 Open problems and challenges . . . 10
  - 1.4.1 Compressible flow . . . 11
  - 1.4.2 The onset of turbulence . . . 12
  - 1.4.3 Existence and smoothness of the Navier-Stokes equations . . . 13
- 1.5 Summary of thesis . . . 14

2 Pseudospectral method 16
- 2.1 Spatial discretization . . . 16
  - 2.1.1 Aliasing . . . 20
  - 2.1.2 Spectral derivatives . . . 23
- 2.2 The vorticity formulation of the Navier-Stokes equations . . . 24
  - 2.2.1 Obtaining $\mathbf{u}$ from ω . . . 26
- 2.3 Time discretization . . . 28
- 2.4 Higher order . . . 31
  - 2.4.1 Initialization procedure for higher order methods . . . 33
- 2.5 Testing the method . . . 36
  - 2.5.1 Rotating cone problem . . . 36
  - 2.5.2 Steady sine function . . . 38
- 2.6 Stability of the method . . . 40
- 2.7 Variations on the pseudospectral method . . . 44

3 The Lagrange-Galerkin method 47
- 3.1 Characteristic curves . . . 48
- 3.2 Solving for the characteristics . . . 52
  - 3.2.1 Evaluating X . . . 54

3.3 Approximate FT using Taylor polynomials 56
3.4 Divergence free space 58
3.5 Some test problems 60
3.5.1 Rotating cone problem 60
3.5.2 Steady sine function 62
3.5.3 Vortex convergence problem 63
3.5.4 Stream vortex problem 65
3.6 Stability of the method 67
3.7 Variations on the Lagrange-Galerkin method 71

4 Adaptive Lagrange-Galerkin method **73**
4.1 Error estimate 73
4.1.1 A modified backward difference formula 75
4.2 Modifying the step size 78
4.2.1 Halving Δt 78
4.2.2 Doubling Δt 80
4.2.3 Overshooting 80
4.3 Some test problems 81
4.3.1 Vortex convergence problem 81
4.3.2 Stream vortex problem 82
4.4 Variations on the adaptive Lagrange-Galerkin method 84
4.4.1 Runge-Kutta time stepping 84
4.4.2 Variable multistep methods 86

5 Results and comparisons **87**
5.1 Speed vs. accuracy 87
5.2 More comparative tests 88
5.2.1 A random problem 89
5.3 Further possible modifications 92

6 Conclusions **95**

Bibliography **97**

List of Tables

2.1 BDF coefficients 29
2.2 PS errors for the rotating cone problem 38
2.3 PS errors for the standing sine problem 40
2.4 Minimum steps for instability 41
2.5 Newton extrapolation coefficients 42
2.6 Comparison of z and $a\Delta tp$ 43

3.1 LG errors for order versus T 61
3.2 LG errors for order versus T with $\nu = 5 \times 10^{-4}$ 61
3.3 LG errors for order versus N 62
3.4 Comparison of LG errors for order versus N 62
3.5 LG errors for the standing sine problem 63
3.6 LG errors for the vortex problem 64
3.7 LG errors for the stream vortex problem 67

4.1 mBDF coefficients 76
4.2 Local error constants 77
4.3 A-LG errors for the vortex problem 82
4.4 A-LG errors for the stream vortex problem 83

5.1 Computational time 87
5.2 Results for the problem with a random initial condition 89
5.3 Results for a more viscous problem 92

List of Figures

2.1 Aliasing example 21
2.2 Order versus error 31
2.3 Leapfrogging for $s = 3$ 36
2.4 Instability test 39
2.5 Instability example 41
2.6 Stability regions for the linear advection equation 45
2.7 positive imaginary axis for $s = 3$ and 4 46

3.1 Characteristic curves 49
3.2 Predictor-Corrector algorithm 53
3.3 LG solution of the vortex problem 65
3.4 $\log_2 \mathcal{E}$ of the vortex errors 66
3.5 LG solution of the stream vortex problem 68
3.6 $\log_2 \mathcal{E}$ of the stream vortex errors 69

4.1 Old list to new list 80
4.2 Overshooting T 81
4.3 A-LG solution for the vortex problem 82
4.4 Δt vs. t for the vortex problem 83
4.5 A-LG solution to the stream vortex problem 84
4.6 Δt vs. t for the stream vortex problem 85

5.1 LG solution for the random problem 90
5.2 A-LG solution for the random problem 91
5.3 Δt vs. t for the random problem 92
5.4 LG solution for a more viscous problem 93
5.5 A-LG solution for a more viscous problem 94
5.6 Δt vs. t for a more viscous problem 94

Chapter 1

Introduction

Fluids can be encountered in almost every aspect of our daily life; from that hot cup of morning coffee to the very air we breathe. Some other examples include the instability of flow down a pipe, the jetstream in the atmosphere, volcanic lava flow in the Earth, the swimming of bio-organisms, oil reservoir simulation and magnetohydrodynamics [67]. It is no wonder that there is such an interest in not only how fluids behave, such as the propagation of waves in a pool, but in how they interact with their environment, such as how the air flow over a wing provides negative pressure and hence lift. The behavior of fluids can be simple or complex. Some simple cases can be solved directly [1]. The more complex problems however require a numerical solution. These numerical solutions are often quite computationally time consuming, especially in three dimensional flow. So the name of the game becomes how to trim down these computations from the most recent benchmarks, all the while maintaining a desired level of accuracy. This thesis will present and compare three possible methods for the numerical solution of fluid dynamics. An overview of the pertinent fluid equations will be given before these three methods can be introduced.

1.1 The Navier-Stokes equations

First we let

$$\mathbf{x} = (x(t), y(t), z(t))$$

be a Euclidean position vector and then define

$$\mathbf{u} = (u(\mathbf{x}, t), v(\mathbf{x}, t), w(\mathbf{x}, t))$$

to be the velocity for a given point $\mathbf{x}$ at time t. u, v, and w are the velocities in the x, y, and z directions respectively and depend on x, y, z, and t. We can think of $\mathbf{x}$ as the position of a tiny particle suspended in the fluid at a given time t. Then $\mathbf{u}$ becomes the velocity of that particle at that same time.

It is assumed that the fluids discussed herein are ideal. An ideal fluid has a uniform density (ρ is constant), is incompressible ($\nabla \cdot \mathbf{u} = 0$) and the force on a surface element $\mathrm{d}S$ can be defined by $p\mathbf{n}\mathrm{d}S$, where $p(\mathbf{x}, t)$ is the pressure function and $\mathbf{n}$ is the normal to the surface element $\mathrm{d}S$. S refers to the closed surface of a fixed volume V in the domain [1, 12].

A fluid is said to be inviscid if there is no interaction between the particles of the fluid or if these interactions are small enough to be ignored [46]. The basic equations that describe the motion of inviscid fluids are called Euler's equations. Euler's equations of motion for inviscid fluids are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{g} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1.2}$$

The second equation is the incompressibility condition. If the fluid is incompressible then the net flow passing through the entire surface S must be zero [12], in which

case (1.2) can be shown as follows

$$0 = \int_S \mathbf{u} \cdot \mathbf{n} dS = \int_V \nabla \cdot \mathbf{u} dV \Rightarrow \nabla \cdot \mathbf{u} = 0,$$

since this must hold for all bodies V. In (1.1) p is the pressure within the system, which like $\mathbf{u}$ can depend on x, y, z and t. ρ is called the mass density. $\mathbf{g}$ represents any external forces acting on the system. This could be anything from a magnetic force acting on a metal in a liquid state (although this would be by far the simplest case of magnetohydrodynamics [10]) to the ever present force of gravity. With the flow defined as it has been above then there are a few 'specialized' types or conditions in the flow that have very simple representations. Steady flow, for example, is a flow which remains constant and does not change with time and can be represented as

$$\frac{\partial \mathbf{u}}{\partial t} = 0.$$

Irrotational flow, as the name implies, has no points where the flow rotates and can be represented with

$$\nabla \times \mathbf{u} = 0.$$

Otherwise we call ω, defined as

$$\omega = \nabla \times \mathbf{u},$$

the vorticity of the flow.

Euler's equations, given by (1.1) and (1.2), only describe the motion of inviscid fluids. However, since all fluids are viscous to one extent or another, it seems reason-

able to include viscosity into the equations for the motion of the fluid. The viscous counterparts to Euler's equations are known as the Navier-Stokes (N-S) equations. These are given by

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g} \tag{1.3}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1.4}$$

where $\nu = \frac{\mu}{\rho}$ and $\mu =$ *coefficient of viscosity.* These equations are derived from three principles of physical motion applied to a mass of fluid: conservation of mass, balance of momentum and the conservation of energy [12]. One other useful piece of information when discussing fluid flow is the dimensionless Reynolds number given by

$$R = \frac{UL}{\nu}$$

where $U =$ *characteristic velocity* and $L =$ *characteristic length.* The Reynolds number can also be thought of as a ratio of the magnitude of the two dominant terms in the Navier-Stokes equations: the convection term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ and the diffusion term $\nu \nabla^2 \mathbf{u}$ [1]. This ratio can be written as

$$\frac{|(\mathbf{u} \cdot \nabla)\mathbf{u}|}{|\nu \nabla^2 \mathbf{u}|} = \mathcal{O}\left(\frac{U^2/L}{\nu U/L^2}\right) = \mathcal{O}(R).$$

The Reynolds number can be used to determine when two flows are similar [12]. If we looked at a small pebble ($L_1 = 0.1cm$) rolling around in water ($\nu_1 = 0.001\frac{cm^2}{s}$) at a moderate speed ($U_1 = 10\frac{cm}{s}$), and compared that to a bowling ball ($L_2 = 10cm$) rolling around in olive oil ($\nu_2 = 1.0\frac{cm^2}{s}$) at slow speed ($U_2 = 0.1\frac{cm}{s}$), we would find

that the same model could be used to represent both cases, since

$$\frac{U_1 L_1}{\nu_1} = \frac{U_2 L_2}{\nu_2} = 1.$$

There are more practical uses for the Reynolds number, such as determining the thickness of a boundary layer δ using the formula

$$\frac{\delta}{L} = \mathcal{O}(R^{-\frac{1}{2}}).$$

The Reynolds number can also be used to split flows into two broad categories: low Reynolds flow and high Reynolds flow. Although there is no set division as to what constitutes low or high Reynolds flow, a good guideline would be $R \ll 1$ (R much smaller than 1) for low Reynolds flow and $R \gtrsim 1$ (R near or greater than 1) for high Reynolds flow [1]. This distinction is useful since the behaviour of a fluid could change from one case to another. Consider the behaviour of air at low speeds, that is to say situations of low Reynolds flow like driving a car, the flow around an object can be modeled quite accurately by the Navier-Stokes equations. However at higher Reynolds flow, such as the case of a rocket travelling faster than the speed of sound, the simplified form of the Navier-Stokes equations that has been presented thus far may not be sufficient. A newer set of equations would be needed to account for the compressibility of the air at and beyond the speed of sound.

In order to keep things simple we will restrict our attention to the two dimensional incompressible case and use periodic boundary conditions. In two dimensions, the vorticity ω becomes a scalar [22]. With periodic boundary conditions we do not

have to worry about discontinuities near the boundary caused by sharp features. With these two simplifications the high accuracy of the pseudospectral method can be maintained with little effort. We will also work within the region $\Omega = [0, 2\pi]^2$ knowing that any other rectangular area can be accomodated using a change of scale.

1.2 Overview of spectral methods

All of the methods discussed in this thesis can be classified under the broad label of spectral methods. The strength of the spectral methods lies in the transformation into Fourier space. This transformation carries many advantages, the most important of which are speed (via the Fast Fourier Transform (FFT), or the Inverse Fast Fourier Transform (IFFT) where applicable), the ease with which derivatives may be applied, and speed of convergence.

The FFT is possibly one of the most important computational algorithms of the past century. The FFT was a huge leap forward in areas such as signal processing, numerical analysis and, of course, spectral methods [7]. Being able to apply a Discrete Fourier Transform (DFT) in $\mathcal{O}(N \log N)$ time rather than the cumbersome $\mathcal{O}(N^2)$ is what makes these spectral methods practical.

When we wish to apply a derivative to some function in Fourier space the operation is a multiplication by a scalar. For example, given a sufficiently smooth differentiable function, $f(x)$ on $x \in \mathbb{R}$, its Fourier coefficients corresponding to a frequency p can be represented as $\hat{f}(p)$ in the following way

$$\hat{f}(p) = \int_{-\infty}^{\infty} f(x) e^{-ipx} \mathrm{d}x, \quad p \in \mathbb{R}. \tag{1.5}$$

The process of getting $\hat{f}(p)$ from $f(x)$ is referred to as a Fourier transform. To undo this process an inverse Fourier transform can be applied and is defined as

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(p) e^{ipx} \mathrm{d}p, \quad x \in \mathbb{R}. \tag{1.6}$$

Then for a derivative of $f(x)$ in Fourier space, $\frac{\partial f}{\partial x}$ defined as in (1.6) will have coefficients $ip\hat{f}(p)$. If $f(x)$ is a periodic function with a period of 2π then the Fourier transform can be evaluated with an infinite sum rather than an integral. The periodic equivalent of (1.6) is written as

$$f(x) = \sum_{p=-\infty}^{\infty} \hat{a}(p) e^{ipx} \tag{1.7}$$

where $\hat{a}(p)$ is defined as

$$\hat{a}(p) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ipx} \mathrm{d}x, \quad p \in \mathbb{Z}.$$

We refer to (1.7) as the Fourier series expansion of a 2π-periodic function $f(x)$. Since we cannot sample every single point $x \in \mathbb{R}$ then what we use instead is a Discrete Fourier Transform (DFT) where only a discrete number of evenly spaced points, x_j, are sampled [65]. At the discrete grid points $x_j = \frac{2j\pi}{N}$, $f(x_j)$ is represented by

$$f(x_j) = \frac{1}{N} \sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{a}_p e^{ipx_j}, \quad j = 0, \ldots, N-1 \tag{1.8}$$

where

$$\hat{a}_p = \sum_{j=0}^{N-1} f(x_j) e^{-ipx_j}, \quad p = -\frac{N}{2}, \ldots, \frac{N}{2} - 1.$$

The Fourier coefficients of the derivative of $f(x)$ when represented in the form given by (1.8) are $ip\hat{a}_p$. The advantage to dealing with the derivatives in Fourier space is that the task of taking the derivative of a discrete periodic function $f(x)$ is reduced to a matrix-vector multiplication [27, 43].

The area that most spectral methods differ is in how they deal with the non-linear convective term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ in (1.3). Products in real space become convolutions in Fourier space. The convolution of two one dimensional discrete functions $u(n)$ and $v(n)$, periodically extended to $\mathbb{Z}$ for $n = 0, \ldots, N-1$, can be defined as

$$(u * v)(n) = \sum_{m=0}^{N-1} u(m) v(n-m), \quad \text{for } n = 0, \ldots, N-1.$$

To evaluate the convolution in this form would require $\mathcal{O}(N^2)$ operations, quite time consuming when compared to the other operations in Fourier space. The pseudospectral method deals with this problem by making an FFT transformation back into real space ($\mathcal{O}(N \log N)$), then performing the product directly ($\mathcal{O}(N)$), then transforming back into Fourier space again ($\mathcal{O}(N \log N)$) for a total of $\mathcal{O}(N \log N)$ operations [52, 65]. In the two dimensional case that we will be working in, we get a savings of $\mathcal{O}(2N^2 \log N)$ versus $\mathcal{O}(N^4)$. For a grid size of $N = 32$, that is the difference between 10^6 and 10^4.

Semi-Lagrangian methods handle the convolution term by looking along the path the 'particle' velocity follows where the convolution term can be combined into a

single term with the time derivative. In other words, we are looking to change from the old coordinate system $(\mathbf{x}, t)$ to a new coordinate system $(X(\mathbf{x}, t), t)$ where $U(\mathbf{x}, t) = \mathbf{u}(X, t)$. This new coordinate system is chosen such that the material derivative (the left side of (1.3)) becomes a directional derivative in the direction of X. This directional derivative is denoted as U_t and is defined as

$$U_t = \mathbf{u}_t(X, t) + (\mathbf{u}(X, t) \cdot \nabla)\mathbf{u}(X, t).$$

In this new coordinate system (1.3) now looks like

$$U_t = -\frac{1}{\rho}\nabla p + \nu\nabla^2 U + \mathbf{g}.$$

If $\mathbf{g} = 0$ and p is constant then the equation (1.3) is reduced to a pure diffusion equation for U, and the problematic convective term has been assimilated into the term U_t.

1.3 Overview of important references

For comprehensive articles on the subject of spectral methods we need only look as far as Gottlieb and Orszag [31] and Fornberg and Sloan [27]. While the latter tends to focus almost entirely on pseudospectral methods, including a treatment of pesudospectral methods as a special case of finite difference methods, it is none the less a good introduction to the topic. The former is more suited to a reader looking for a broader, more comprehensive view, including aspects of stability, convergence, accuracy, and many explicative examples. Previous papers by Orszag would also

cover the same topics [51, 52]. For a more comprehensive resource one could look to the treatment given by Canuto et al. [8]. This is a very influential work on the subject of spectral methods. For a slightly more mathematical and up to date reference one could look at Peyret [53]. Another article that was heavily referenced in regards to pseudospectral methods was that of Schneider, Kevlahan and Farge [22]. This was also used as the basis for the comparisons done in Sections 3.5.3 and 4.3.1. The basis for the comparisons in Sections 3.5.4 and 4.3.2 are the works done by Ware [66, 67]. These were also greatly referenced in regards to the spectral Lagrange-Galerkin method. The last of the extensively referenced works is treatment of numerical methods given by Iserles [40]. This was a particularly influential work in regards to the time stepping and adaptivity algorithms.

One final work of importance is the FFT algorithm presented by Cooley and Tukey [16], without which spectral methods would simply not be feasible.

1.4 Open problems and challenges

The subject of fluid dynamics is a broad and widely applied field. Many other aspects of the field are widely researched and yet we still do not possess a full understanding of the subject. Other areas that could be discussed are: compressible flows, shocks and waves [12], the conditions and triggers for turbulence [18, 46], non-periodic boundary conditions, high viscosity flow [1], thermal convection [41] and combustible fluids [44]. Full texts can be written, as noted in the previous section, and still not cover all the possible topics.

The aim of this section is to look at a few other topics and open problems in fluid

dynamics. The main focus will be on how they differ from what has been previously discussed and what the potential impact could be.

1.4.1 Compressible flow

It is somewhat unrealistic to restrict our attention to incompressible fluids since, to some extent, all fluids are compressible. However, when the Mach number M is small it can be shown that the N-S equations will provide a good approximation to the behavior of a compressible fluid. The Mach number here refers to another dimensionless number describing the ratio of the velocity of the flow compared to the speed of sound, c. First, assume that the fluid is adiabatic, that ρ, p and $\mathbf{u}$ depend on $\mathbf{x}$ and t and define the constant γ as $\gamma = \frac{C_p}{C_v}$ where $C_p =$ *specific heat at constant pressure* and $C_v =$ *specific heat at constant volume.* An adiabatic system is one in which no energy is lost or gained from outside sources. The Mach number may then be defined as

$$M = \frac{|\mathbf{u}|}{c}$$

where

$$c^2 = \frac{\gamma p}{\rho}.$$

In a more general setting, one way of writing the compressible fluid equations would be

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \mu \nabla^2 \mathbf{u} \tag{1.9}$$

where λ is defined by the relation $\zeta = \lambda + \frac{2}{3}\mu$, where $\zeta =$ *second coefficient of viscosity*

and the enthalpy ε (energy per unit mass) is given by

$$\mathrm{d}\varepsilon = \frac{1}{\rho}\frac{\mathrm{d}p}{\mathrm{d}t}, \tag{1.10}$$

assuming the energy within the system is constant [8, 12]. The equation (1.9) differs from (1.3) only by the term $(\lambda + \mu)\nabla(\nabla \cdot \mathbf{u})$ and the fact that ρ and p depend on $\mathbf{x}$ and t. The dependence on $\mathbf{x}$ and t can be related by (1.10), and for obvious reasons the incompressibility condition (1.4) is not used.

1.4.2 The onset of turbulence

Spectral methods, particularly the semi-Lagrangian methods, have been used to investigate turbulent flows. A question that remains is when this turbulence will occur. Partial answers can be investigated using pseudospectra [4, 11, 48, 52]. Pseudospectra in this case are different from the pseudospectral method referred to in the remainder of this paper. Here we are referring to the spectra of slightly perturbed operators. Another way to describe pseudospectra is as follows: we define $\mathcal{A}$ to be an operator on u then we define the spectrum $\Lambda(\mathcal{A})$ to be

$$\Lambda(\mathcal{A}) = \{\lambda \in \mathbb{C} | \lambda I - \mathcal{A} \text{ is not invertible}\}.$$

The ϵ-pseudospectrum $\Lambda_\epsilon(\mathcal{A})$ is then defined to be the set of all $\lambda \in \mathbb{C}$ such that

$$\left\|(\lambda I - \mathcal{A})^{-1}\right\| \geq \frac{1}{\epsilon}.$$

$\Lambda(\mathcal{A})$ are often used to analyse the operator $\mathcal{A}$. $\Lambda_\epsilon(\mathcal{A})$ can then be used to provide further analysis in a way $\Lambda(\mathcal{A})$ cannot [56, 63].

These ϵ-pseudospectra give a hint as to why turbulence sets in sooner than expected in an experiment where the velocity of the flow is slowly being increased from a stable state. It is clear that eventually turbulence will occur, but the current models based on the spectrum do not yield results that are sufficiently close to the results of the physical experiments. If we extend those models to include the pseudospectrum then the model becomes an improved predictor for the experiments. Although this does provide a vast improvement, the predictions are still not as close as desired to the experimental results. What is also still missing from this topic is a complete understanding as to why the inclusion of the pseudospectra provides improved results [18, 64].

1.4.3 Existence and smoothness of the Navier-Stokes equations

Another open challenge that exists for the N-S equations is that the existence and smoothness of the solutions has not yet been proven for three dimensional flows, for time $[0, \infty)$, given smooth initial conditions. (The existence and smoothness can be shown for two dimensions.) It can also be shown that the existence and smoothness holds for certain restrictions on the initial condition $\mathbf{u}_0$ and the time interval $[0, T)$. The vague restriction on $\mathbf{u}_0$ is that it must be sufficiently small [13]. For the time interval, existence and smoothness can be shown when T depends on the initial data. In addition, it is known that N-S equations satisfying the above conditions will always have a weak solution, but the uniqueness of these weak solutions has not yet been proven.

When fluid equations can be found in such a vast number of fields as mentioned above, yet their existence and smoothness cannot be shown, it is of little surprise to see that proving the existence and smoothness for the N-S equations, became one of the seven Clay Mathematics Institute's Millennium problems [25].

1.5 Summary of thesis

We will begin with a look at the pseudospectral method for two reasons. First, this method will be the method to which the Lagrange-Galerkin and its adaptive variation will be compared. The pseudospectral method has been chosen as this standard since it is widely known to provide highly accurate solutions if given sufficient time steps and for periodic boundary conditions [22]. The main fault with the pseudospectral method is that it is only conditionally stable. As shall be seen in Section 2.6, there is a minimum step size that must be taken so that the method will not diverge or as it is sometimes referred to: blow up. A second reason to start with the pseudospectral method is that it is a fairly straightforward spectral method and will thus provide a good introduction to the spatial discretization of spectral methods that we will see in Section 2.1 and the remainder of the thesis [31].

Having laid down the groundwork for spectral methods, specifically the pseudospectral method in Chapter 2, we then look at the Lagrange-Galerkin method in Chapter 3. The most sensible way to discuss the Lagrange-Galerkin method is to first discuss the method of characteristics as in Sections 3.1 and 3.2, since the Lagrange-Galerkin method can be thought of as a multi-dimensional extension of that method [38, 66]. Having completed the discretization in time we then look at

the tools that will be needed to compute the spatial solution. To accomplish this we use an approximation to an unevenly spaced Fourier transform, as seen in Section 3.3 [68], and then solve the Helmholtz equation.

Once the Lagrange-Galerkin method has been described, then the only element that is needed in order to form an adaptive Lagrange-Galerkin method is some means of measuring the error, which shall be covered in Section 4.1. A way to adjust the step size accordingly as in Section 4.2 will be examined [40]. Each chapter will be finished off with a few examples to illustrate the performance of the various methods.

Then for Chapter 5 there is a comparison of the two forms of the Lagrange-Galerkin method for certain random and some selected initial conditions [36]. The intent of this comparison is to highlight the conditions for which adaptivity in time will be an asset to the computation.

Chapter 2

Pseudospectral method

The pseudospectral (PS) method has a long history of use for numerical simulation of fluid flows with periodic boundary conditions [4, 5, 27, 37, 53, 66]. It is highly accurate and, with the use of the FFT, can be implemented very efficiently. For these reasons it shall be used to determine a reference solution to which the other methods can be compared, as was done in [22]. While this reference solution may not be the exact solution to the given problem, if a very small time step is used, it will be very close. A standard reference for pseudospectral methods is [8] and is heavily referenced in this section.

2.1 Spatial discretization

Given a function $u \in L^2_{\mathbb{C}}([0, 2\pi])$, we can represent the function with its continuous Fourier series representation

$$Su(x) = \sum_{p=-\infty}^{\infty} \hat{u}_p e^{ipx} \tag{2.1}$$

where

$$\hat{u}_p = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ipx} \mathrm{d}x. \tag{2.2}$$

If we define

$$\phi_p(x) = e^{ipx},$$

we can write

$$\hat{u}_p = \langle u, \phi_p \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the usual definition of a weighted inner product on $L^2_{\mathbb{C}}([0, 2\pi])$ [15], defined for $f, g \in L^2_{\mathbb{C}}([0, 2\pi])$ by

$$\langle f, g \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x)\overline{g(x)}\mathrm{d}x. \tag{2.3}$$

It may be noted that the ϕ_p are orthonormal functions, since

$$\int_0^{2\pi} \phi_p(x)\overline{\phi_q(x)}\mathrm{d}x = 2\pi\delta_{pq}$$

where δ_{pq} is the Kroenecker delta function. Su as it stands right now could be described as discrete but not finite-dimensional. Assuming N is even, then define a finite-dimensional approximation via

$$P_N u(x) = \sum_{p=-\frac{N}{2}}^{\frac{N}{2}} \hat{u}_p \phi_p(x).$$

If

$$S_N = \text{span}\left\{ \phi_p(x) \,\middle|\, -\frac{N}{2} \leq p \leq \frac{N}{2} \right\}$$

then

$$\langle P_N u, v \rangle = \langle u, v \rangle, \quad \forall v \in S_N, \tag{2.4}$$

so that P_N is the orthogonal projection in $L^2_{\mathbb{C}}([0, 2\pi])$ onto S_N.

Using $P_N u(x)$ is pointless unless it can be shown that not only does the approxi-

mation converge to $u(x)$ but that it converges rapidly. This can be shown using the properties of the Fourier transform and the following lemma.

Lemma 1 (Riemann-Lebesgue). *If* $u \in L^2_{\mathbb{C}}[0, 2\pi]$, *then* $\langle u(x), \phi_p(x) \rangle \to 0$ *as* $p \to \pm\infty$.

This lemma enables us to establish some estimates on $|\hat{u}|$ for sufficiently-smooth u. We assume that u and its first $r-1$ derivatives are continuous and 2π-periodic and that its r^{th} derivative is integrable. Then if we take equation (2.2) and apply integration by parts we get

$$\hat{u}_p = \frac{1}{2\pi(ip)^r} \int_0^{2\pi} u^{(r)}(x) e^{-ipx} \mathrm{d}x \quad p \in \mathbb{Z}$$

where $u^{(r)}(x)$ is the r^{th} derivative of $u(x)$. Then since we assume $u^{(r)}(x)$ is integrable we can apply the Riemann-Lebesgue lemma to get

$$|\hat{u}_p| \ll \frac{1}{p^r}, \quad p \to \pm\infty.$$

From this it can be shown that if u, extended periodically to $\mathbb{R}$, is in $C^\infty(\mathbb{R})$ then $P_N u(x)$ converges to $u(x)$ faster than any finite power of $\frac{1}{N}$ as $N \to \infty$, $\quad \forall x \in \mathbb{R}$ [31]. In simpler terms, very little accuracy is lost when we truncate $Su(x)$ at a sufficiently large value of N. To clarify, we can say that for any $l < \infty$, and any sufficiently smooth u, there is a constant C such that [8]

$$\|u - P_N u\|_{L^2} \leq C N^{-l} \left\|u^{(l)}\right\|_{L^2}, \quad \forall N > 0.$$

This rapid decay of the error is often refered to as "spectral" accuracy.

In practice, numerical integration must be used to compute $\hat{u}_p$. This leads to the definition of the discrete Fourier transform. With $x_j = 2\pi\frac{j}{N}$, $j \in \mathbb{Z}$, the discrete Fourier coefficients are defined as

$$\tilde{u}_p = \frac{1}{N}\sum_{j=0}^{N-1} u(x_j)e^{-ipx_j} \tag{2.5}$$

$$= \frac{1}{N}\sum_{j=-\frac{N}{2}}^{\frac{N}{2}}{}'' u(x_j)e^{-ipx_j} \quad \text{for } -\frac{N}{2} \le p \le \frac{N}{2}, \tag{2.6}$$

where $\sum_p''$ is defined to be the regular definition of summation except that the first and last terms are halved [65, 67]. This second form of $\tilde{u}_p$ is equivalent because of the periodicity. We can represent the function u discretely at x_j as

$$u(x_j) = \sum_{p=0}^{N-1} \tilde{u}_p e^{ipx_j} \tag{2.7}$$

$$= \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' \tilde{u}_p e^{ipx_j} \quad \text{for } j = 0, 1, \ldots, N-1. \tag{2.8}$$

If

$$I_N u(x) = \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' \tilde{u}_p e^{ipx} \tag{2.9}$$

then, under the condition that u is as above, $I_N u(x_j) = u(x_j)$ for $j = 0, 1, \ldots, N-1$. Thus $I_N u$, the discrete Fourier series of u, interpolates u at the points x_j. Again it can be shown [8] that, for sufficiently smooth u, and for any $l > \infty$, there is a

constant C such that

$$\|u - I_N u\|_{L^2} \leq C N^{-l} \left\|u^{(l)}\right\|_{L^2}, \quad \forall N > 0. \tag{2.10}$$

A better way to think of this is that (2.6) is the numerical quadrature of the integral (2.2), wrapped onto $[-\pi, \pi]$ periodically, via the trapezoidal rule.

2.1.1 Aliasing

With the current notation we can say

$$u(x_j) = Su(x_j) = \sum_{p=-\infty}^{\infty} \hat{u}_p e^{ipx_j} = I_N u(x_j) = \sum_{p=-\frac{N}{2}}^{\frac{N}{2}\,\prime\prime} \tilde{u}_p e^{ipx_j}. \tag{2.11}$$

Looking at the far right side of equation (2.11) we can then write

$$\begin{aligned}
\tilde{u}_p &= \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ipx_j} \quad \text{for } p = -\frac{N}{2}, \ldots, \frac{N}{2} \\
&= \frac{1}{N} \sum_{j=0}^{N-1} \left(\sum_{q=-\infty}^{\infty} \hat{u}_q e^{iqx_j} \right) e^{-ipx_j} \\
&= \frac{1}{N} \sum_{q=-\infty}^{\infty} \hat{u}_q \sum_{j=0}^{N-1} e^{i(q-p)x_j}.
\end{aligned}$$

Discrete orthogonality can be defined as

$$\sum_{j=0}^{N-1} e^{i(q-p)x_j} = \begin{cases} N & \text{if } p - q = mN, \quad m \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases}.$$

Then we can then say

$$\tilde{u}_p = \hat{u}_p + \sum_{m \neq 0} \hat{u}_{p+mN}. \tag{2.12}$$

It is easy to see that at the grid points x_j that $\phi_{p+mN}(x_j) = \phi_p(x_j)$. An example of this phenomenon is shown in Figure 2.1.1. When frequencies differ by multiples of N we say that they are aliased.

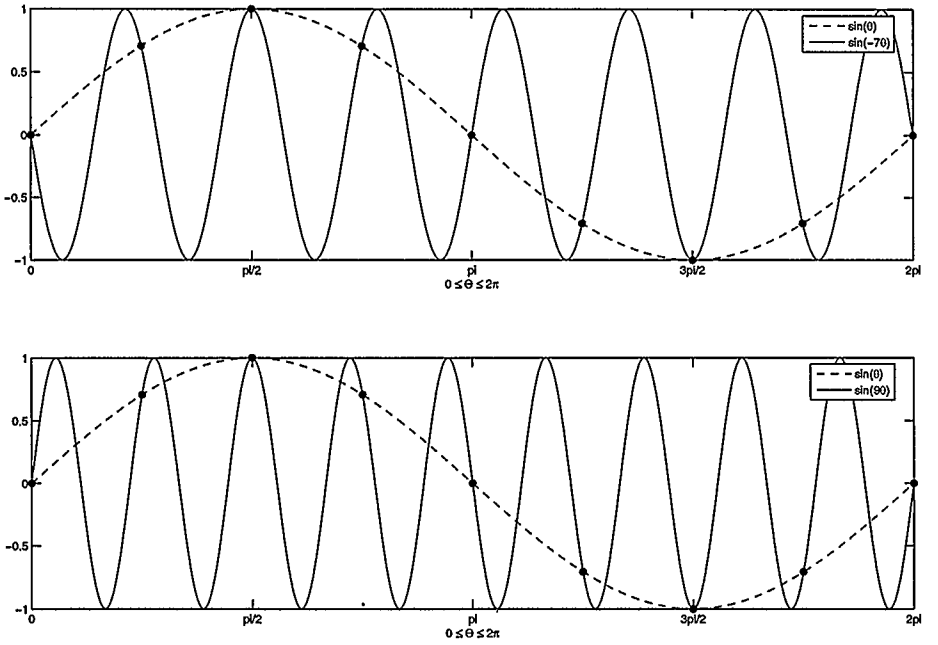

Figure 2.1: **Aliasing example:** In the two examples we can see that when only looking at 8 points on the graph the frequencies of $n = -7, 1$ and 9 will be indistinguishable.

If (2.12) is substituted into (2.9) then we can say

$$I_N u(x) = \sum_{p=-\frac{N}{2}}^{\frac{N}{2}\,\prime\prime} \left(\hat{u}_p + \sum_{|m| \geq 1} \hat{u}_{p+mN} \right) e^{ipx}$$

$$= \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' \hat{u}_p e^{ipx} + \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' \sum_{|m|\geq 1} \hat{u}_{p+mN} e^{ipx}$$

$$= P_N u(x) + \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' \left(\sum_{|m|\geq 1} \hat{u}_{p+mN} \right) e^{ipx}.$$

Define the aliasing error $\mathcal{E}_a$ as

$$\mathcal{E}_a = I_N u(x) - P_N u(x).$$

To show that $\|\mathcal{E}_a\| \leq CN^{-l} \left\|u^{(l)}\right\|_{L^2}$ we proceed as follows

$$\begin{aligned} \|u - I_N u\|_{L^2}^2 &= \|u - P_N u + P_N u - I_N u\|_{L^2}^2 \\ &= \|u - P_N u\|_{L^2}^2 + \|P_N u - I_N u\|_{L^2}^2 \quad \text{(by orthogonality)} \\ &= \|u - P_N u\|_{L^2}^2 + \|\mathcal{E}_a\|_{L^2}^2 . \end{aligned}$$

It follows that

$$\|u - I_N u\|_{L^2} \geq \|\mathcal{E}_a\|_{L^2}$$

and from (2.10) we can say

$$\|\mathcal{E}_a\|_{L^2} \leq CN^{-l} \left\|u^{(l)}\right\|_{L^2} .$$

What this means is that the aliasing error is of the same order as the interpolation error [8, 53].

2.1.2 Spectral derivatives

Knowing $I_N u(x)$ we can differentiate:

$$(I_N u(x))' = \sum_{p=-\frac{N}{2}}^{\frac{N}{2}}{}'' ip\tilde{u}_p e^{ipx}.$$

Hence, to get the derivative of a Fourier series of a function, we simply multiply the Fourier coefficients by ip, a scalar multiple of their respective index numbers p. Now $(I_N u(x))' \neq I_N u'(x)$ but it can be shown [8] that the error $(I_N u(x))' - I_N u'(x)$ is of the same order as the truncation error for u'; i.e., as of

$$\|u' - (I_N u)'\|_{L^p} \leq CN^{1-l} \left\|u^{(l)}\right\|_{L^p}.$$

Now consider the multidimensional version of the Fourier transform and its application to the two dimensional Navier-Stokes equations. If we take $\mathbf{u}(x, y, t)$ to be the velocity of a two dimensional flow, then

$$I_N \mathbf{u}(x, y, t) = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \tilde{\mathbf{u}}_{pq} e^{i(px+qy)}$$

where in a similar fashion to (2.9)

$$\tilde{\mathbf{u}}_{pq} = \frac{1}{NM} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \mathbf{u}(x_j, y_k) e^{-i(px_j+qy_k)}.$$

Then given $\mathbf{u}$ satisfying $\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g}$, such that $\nabla \cdot \mathbf{u} = 0$, we

can define $\mathcal{L}(\mathbf{u})$ to be

$$\mathcal{L}(\mathbf{u}) = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla p + \nu\nabla^2\mathbf{u} + \mathbf{g} \tag{2.13}$$

so that

$$\mathbf{u}_t - \mathcal{L}(\mathbf{u}) = 0. \tag{2.14}$$

Then, projecting:

$$\begin{aligned} 0 &= I_N(\mathbf{u}_t - \mathcal{L}(\mathbf{u})) \\ &= (I_N\mathbf{u}_t) + I_N((\mathbf{u}\cdot\nabla)\mathbf{u}) + I_N\nabla p - \nu I_N\nabla^2\mathbf{u} - I_N\mathbf{g}. \end{aligned}$$

We define $\vec{\mathbf{u}}(\mathbf{x},t)$ such that $\vec{\mathbf{u}}(\cdot,t) \in S_N, \forall t$ and $\vec{\mathbf{u}}$ is sufficiently smooth as a function of t such that

$$I_N(\vec{\mathbf{u}}_t + \vec{\mathbf{u}}\cdot\nabla\vec{\mathbf{u}} + \nabla I_N p - \nu\nabla^2\vec{\mathbf{u}} - \mathbf{g}) = 0. \tag{2.15}$$

We wish to solve for the numerical solution $\vec{\mathbf{u}}$ of the equation (2.15) which approximates equation (1.3)[53].

2.2 The vorticity formulation of the Navier-Stokes equations

We will use vorticity form of the Navier-Stokes equations for the pseudospectral method. Since we are working in two dimensions, $\omega = \nabla\times\mathbf{u} = (\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y})\vec{k}$ is equivalent to a scalar. We will have no external forces, meaning that we will let $\mathbf{g} = 0$. To get from the Navier-Stokes equations (1.3) and (1.4) to the vorticity

formulation of the Navier-Stokes equations:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = \nu \nabla^2 \omega \tag{2.16}$$

we take the curl of both sides of the Navier-Stokes equation. We have two reasons for choosing the vorticity formulation. First, in two dimensions ω becomes one dimensional, so when we are solving for each time step (Section 2.3) we need only do half of the work. It is worth noting that this method would not work in three dimensions since the vorticity would also be three dimensional. Solutions in three dimensions are often mixtures of vorticity and velocity and possibly other variables such as the potential and the pressure of the system. The second reason is that the pressure p no longer has a bearing on the outcome. So we are saved the trouble of having to solve for $p(\mathbf{x}, t)$ at each time step. Evaluating the pressure can be a time consuming aspect of the computation [5, 37].

The curl of most terms in the Navier-Stokes equations is straightforward. Only the convective term requires a little care and the use of the incompressibility condition. The curl of the convective term can be given as

$$\nabla \times (\mathbf{u} \cdot \nabla)\,\mathbf{u} = \frac{\partial}{\partial x}\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) - \frac{\partial}{\partial y}\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right).$$

Its expansion using the product rule becomes

$$\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + u\frac{\partial^2 v}{\partial x^2} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + v\frac{\partial^2 v}{\partial x \partial y} - \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} - u\frac{\partial^2 u}{\partial x \partial y} - \frac{\partial v}{\partial y}\frac{\partial u}{\partial y} - v\frac{\partial^2 u}{\partial y^2}.$$

Now using $\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}$, from (1.4), we get

$$-\frac{\partial v}{\partial y}\frac{\partial v}{\partial x} + u\frac{\partial^2 v}{\partial x^2} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + v\frac{\partial^2 v}{\partial x \partial y} - \frac{\partial u}{\partial y}\frac{\partial u}{\partial x} - u\frac{\partial^2 u}{\partial x \partial y} + \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} - v\frac{\partial^2 u}{\partial y^2}$$

which

$$\begin{aligned}
&= u\frac{\partial^2 v}{\partial x^2} + v\frac{\partial^2 v}{\partial x \partial y} - u\frac{\partial^2 u}{\partial x \partial y} - v\frac{\partial^2 u}{\partial y^2} \\
&= \left(u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y}\right)\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \\
&= \left(u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y}\right)\omega \\
&= \mathbf{u} \cdot \nabla\omega.
\end{aligned}$$

For the pressure term we need only recall that $\nabla \times \nabla r = 0$ for any $r(\mathbf{x}, t) \in \mathbb{C}^n$.

2.2.1 Obtaining u from ω

As we shall see in Section 2.3, when we wish to discretize (2.16), there will come a point that when, given $\omega(\mathbf{x}, t)$, we wish to be able to compute $\vec{\mathbf{u}}(\mathbf{x}, t)$. Once again, to accomplish this we look to Fourier space to end up with the following equation [22]

$$\vec{\mathbf{u}}(x, y) = \sum_{\substack{p=-\frac{M}{2} \\ p^2+q^2\neq 0}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \frac{1}{p^2+q^2} \begin{pmatrix} -iq \\ ip \end{pmatrix} \hat{\omega}(p, q) e^{i(px+qy)}. \tag{2.17}$$

This equation is unique up to a constant value. This can be seen by defining $\underline{\mathbf{u}} = \mathbf{u} + c = (u + c, v + c)$, where c is some constant value and $\nabla \cdot \underline{\mathbf{u}} = 0$, then $\underline{\omega} = \frac{\partial(v+c)}{\partial x} - \frac{\partial(u+c)}{\partial y} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = \omega$.

To get (2.17) we start with

$$\nabla \cdot \mathbf{u} = 0$$

or equivalently

$$\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}.$$

Since we know that

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

then

$$\begin{aligned}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u &= \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 v}{\partial x \partial y} \\ &= \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) \\ &= -\frac{\partial \omega}{\partial y},\end{aligned}$$

and

$$\begin{aligned}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) v &= \frac{\partial^2 v}{\partial x^2} - \frac{\partial^2 u}{\partial x \partial y} \\ &= \frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) \\ &= \frac{\partial \omega}{\partial x}\end{aligned}$$

which gives

$$\nabla^2 \mathbf{u} = \begin{pmatrix} -\frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} \end{pmatrix} \omega. \tag{2.18}$$

Now if we interpolate (2.18) in a similar fashion to (2.9), using the Fourier derivative

[46], we get

$$\sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \begin{pmatrix} -iq \\ ip \end{pmatrix} \hat{\omega}(p,q) e^{i(px+qy)} = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' (p^2+q^2)\hat{\mathbf{u}}(p,q) e^{i(px+qy)}, \quad \forall x, y \in \Omega.$$

We can then rewrite the above, assuming $p^2 + q^2 \neq 0$ as

$$\mathbf{u}(x,y) = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \frac{1}{p^2+q^2} \begin{pmatrix} -iq \\ ip \end{pmatrix} \hat{\omega}(p,q) e^{i(px+qy)}.$$

2.3 Time discretization

We discretize in two stages: first in space using the interpolant $I_N\mathbf{u}(\mathbf{x})$, then in time. It shall be seen later that for the Lagrange-Galerkin method the discretization is done first in time, then in space.

There are many possible time stepping algorithms that could be used for this method. A few such examples include the various Euler methods, the Adams-Bashforth [17], Leapfrogging and the Crank-Nicholson scheme [51, 53]. Newer schemes include the Newmark-β scheme [43] and mixtures of the various schemes for different parts of the problem, such as using a Backward Difference Formula (BDF) for the viscosity terms and an Adams-Bashforth method for the convective terms [35, 55]. Another possible method that seems popular with Finite Element Schemes is the discontinuous Galerkin method [29, 57, 58]. Since the time steps Δt are evenly spaced, (for now) then a good method to use is the BDF since it is $A(\alpha)$-stable for order $s \leq 6$. By $A(\alpha)$-stable, we mean that the infinite wedge $\mathcal{V}_\alpha = \{\rho e^{i\theta} | \rho > 0, |\theta + \pi| < \alpha\} \subseteq \mathbb{C}^-$ lies entirely within the linear stability domain $\mathcal{D}$ of the BDF [40]. Another way of

s	β	$\{\alpha_{n+1}, \ldots, \alpha_{n-s+1}\}$
1	1	$\{1, -1\}$
2	$\frac{2}{3}$	$\{1, \frac{-4}{3}, \frac{1}{3}\}$
3	$\frac{6}{11}$	$\{1, \frac{-18}{11}, \frac{9}{11}, \frac{-2}{11}\}$
4	$\frac{12}{25}$	$\{1, \frac{-48}{25}, \frac{36}{25}, \frac{-16}{25}, \frac{3}{25}\}$
5	$\frac{60}{137}$	$\{1, \frac{-300}{137}, \frac{300}{137}, \frac{-200}{137}, \frac{75}{137}, \frac{-12}{137}\}$
6	$\frac{20}{49}$	$\{1, \frac{-120}{49}, \frac{150}{49}, \frac{-400}{147}, \frac{75}{49}, \frac{-24}{49}, \frac{10}{147}\}$

Table 2.1: BDF coefficients

looking at this is to say that if all of the eigenvalues of the operator, in this case the BDF, lie inside $\mathcal{V}_\alpha$, then any errors that are incurred by the method will only decay, rather than grow.

We can write the general form of the s order BDF for $\mathbf{u}_t = \mathcal{L}(\mathbf{u})$ as follows

$$\mathbf{u}^{n+1} - \beta \Delta t \mathcal{L}(\mathbf{u}^{n+1}) = \sum_{j=n-s+1}^{n} \alpha_j \mathbf{u}^j \tag{2.19}$$

where β and α_j correspond to the values in Table 2.1. Let $\hat{\omega}^j(p,q)$ denote the Fourier coefficients of $\omega^j(x,y)$ then, for 2^{nd} order, given $\hat{\omega}^n$ and $\hat{\omega}^{n-1}$ then we wish to solve for $\hat{\omega}^{n+1}$ from the discretization of (2.16), then

$$(I + \Delta t \beta \nu \nabla^2)\hat{\omega}^{n+1} = -\Delta t \beta \widehat{\mathbf{u}^* \cdot \nabla \omega^*} + \sum_{k=1}^{2} \alpha_k \hat{\omega}^{n-k+1}, \tag{2.20}$$

where I is the 2×2 identity matrix and $\widehat{\mathbf{u}^* \cdot \nabla \omega^*}$ is an approximation to $\mathbf{u}^{n+1} \cdot \nabla \omega^{n+1}$ in Fourier space. From (2.19) we can see that we need the term $\Delta t \beta \mathcal{L}(\mathbf{u}^{n+1})$ to

evaulate ω^{n+1} using the BDF. Part of $\mathcal{L}(\mathbf{u}^{n+1})$ requires the evaluation of $\mathbf{u}^{n+1}\cdot\nabla\omega^{n+1}$. This product could be evaluated in Fourier space in a manner similar to $\nabla^2\omega^{n+1}$, but as seen in Section 1.2, in Fourier space a product becomes the convolution operator. It is more efficient to evaluate the product in real space [52]. $\mathbf{u}^{n+1}$ and ω^{n+1} are needed to evaluate the product. Since ω^{n+1} has not been evaluated yet, we use the approximations $\mathbf{u}^*$ and ω^* respectively, as seen on the right side of (2.20). To accomplish this, we extrapolate the value ω^*, then compute $\mathbf{u}^*$ using (2.17). For $s=2$, a linear extrapolation is used, so $\omega^* = 2\omega^n - \omega^{n-1}$ and $\mathbf{u}^*$ is defined by letting

$$\hat{\mathbf{u}}^*(p,q) = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}\,\prime\prime}\sum_{q=-\frac{N}{2}}^{\frac{N}{2}\,\prime\prime} \frac{1}{p^2+q^2}\begin{pmatrix} -iq \\ ip \end{pmatrix}\hat{\omega}^*(p,q), \quad \text{for } p^2+q^2 \neq 0 \tag{2.21}$$

$$\hat{\mathbf{u}}^*(0,0) = 0. \tag{2.22}$$

The $\mathbf{u}^*$ is obtained from applying the inverse Fourier transform to $\hat{\mathbf{u}}^*$.

The general 2nd order algorithm for each time step can then be summarized as follows. Given $\hat{\omega}^n$ and $\hat{\omega}^{n-1}$ then

Step 1. Extrapolate $\hat{\omega}^*$ using $\hat{\omega}^n, \hat{\omega}^{n-1}$, and evaluate $\hat{\mathbf{u}}^*$ using (2.21).

Step 2. $\hat{\omega}^{n+1} := -\Delta t\beta\widehat{\mathbf{u}^*\cdot\nabla\omega^*}$ where the dot product $\mathbf{u}^*\cdot\nabla\omega^*$ is evaluated in real space.

Step 3. $\hat{\omega}^{n+1} := \hat{\omega}^{n+1} + \sum_{k=1}^{s}\alpha_k\hat{\omega}^{n-k+1}$ (apply the BDF).

Step 4. $\hat{\omega}^{n+1}(p,q) := \dfrac{\hat{\omega}^{n+1}}{1-\Delta t\beta\nu(p^2+q^2)}$, since multiplying the coefficients by $-(p^2+q^2)$ is equivalent to ∇^2 in Fourier space.

2.4 Higher order

The main advantage to having higher orders is that for a small increase in the runtime, and usually in the storage as well, we get a significant increase in the accuracy of the method. An example of this idea is shown in Figure 2.2, where an increase in the order of the error can be seen with each increase in the order of the method. This means that we can often achieve a fixed accuracy using fewer timesteps, since each time step has a higher accuracy.

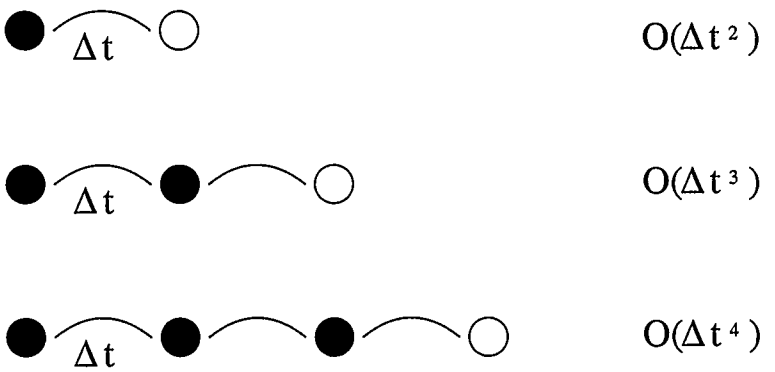

Figure 2.2: **Order versus error:** using the s previous values (filled circles) and an s order method, the local error of the computed value (empty circle) is proportional to Δt^{s+1}. So, for $\Delta t < 1$, the higher the order, the smaller the local error.

The two things that change when using a higher order are the BDF coefficients used in the time stepping and the number of points used for the extrapolation of ω^*. Since the time stepping is done by the BDF which we know to be $A(\alpha)$-stable for order ≤ 6, then all we need to do is update the BDF formula according to Table 2.1.

For order s, if we are given $\hat{\omega}^n, \ldots, \hat{\omega}^{n-s+1}$, then

$$(I + \Delta t \beta \nu \nabla^2)\hat{\omega}^n = -\Delta t \widehat{\mathbf{u}^* \cdot \nabla \omega^*} + \sum_{j=1}^{s} \alpha_j \hat{\omega}^{n-j+1}. \tag{2.23}$$

So now instead of storing only two of the previous results $(\hat{\omega}^n, \hat{\omega}^{n-1})$ we need to store up to six. Since the grid size that we will be using is usually smaller than $N = 2^7 = 128$ then the extra storage is not a problem. If for whatever reason N were larger or we were to work in three dimensions, then the extra storage could become an issue.

The Newton formula for interpolating polynomials [14] is used for the extrapolation of the value ω^*. Given the evenly spaced points in time $t_0, t_1 = t_0 + \Delta t, t_2 = t_0 + 2\Delta t, \ldots, t_{n-1} = t_0 + (n-1)\Delta t$ then we have the corresponding values of $\hat{\omega}^0, \hat{\omega}^1, \hat{\omega}^2, \ldots, \hat{\omega}^{n-1}$. $\hat{\omega}^*$ can then be approximated using

$$\hat{\omega}^* = \sum_{j=0}^{n-1} \hat{\omega}[t_j, \ldots, t_{n-1}] \prod_{k=j+1}^{n-1} (t_n - t_k).$$

The divided-difference $\hat{\omega}[t_j, \ldots, t_{n-1}]$ is defined via

$$\hat{\omega}[t_j, \ldots, t_{j+k}] := \frac{\hat{\omega}[t_{j+1}, \ldots, t_{j+k}] - \hat{\omega}[t_j, \ldots, t_{j+k-1}]}{t_{j+k} - t_j}$$

with $\hat{\omega}[t_j] = \hat{\omega}^j$ for $j = 0, \ldots, n-1$. Since all the elements are equally spaced in time the divided-difference can be simplified to a polynomial of the given order and

represented using the binomial expansion as [14]

$$\hat{\omega}^* = \sum_{j=0}^{n-1} (-1)^{j+1} \binom{s}{j} \hat{\omega}^j.$$

2.4.1 Initialization procedure for higher order methods

When the order s was only two the initialization was not a big problem. Given ω^0 then we simply calculated ω^1 using Δt with a first order (linear) method, then with $\hat{\omega}^0$ and $\hat{\omega}^1$ we can start the main loop of the algorithm. The main loop involved solving for $\hat{\omega}^{n+1}$ using $\hat{\omega}^n$ and $\hat{\omega}^{n-1}$. The error of the linear step is $\mathcal{O}(\Delta t^2)$. Problems arise when this is increased to $s \geq 3$ or higher where, with an exception near the initial steps, it no longer holds true that solving for $\hat{\omega}^n$ using $\hat{\omega}^{n-1}, \hat{\omega}^{n-2}, \ldots, \hat{\omega}^{n-s}$ will be of the desired accuracy. For example, if $s = 3$ then the error is of order Δt^3. So if you were given a starting value of ω^0 and were to then find ω^1 and ω^2 using Δt as the time step, then a linear method would be used to find ω^2. The error for ω^2 is then $\mathcal{O}(\Delta t^2)$. So for $\Delta t < 1$, $\Delta t^2 > \Delta t^3$ so the ensuing values ω^n will be inaccurate since the initial conditions were not accurate to begin with. To fix this problem, we start with $\Delta\sigma = \frac{\Delta t}{2^r}$ where $r \in \mathbb{N}$ is chosen so that $\omega^0, \ldots, \omega^{s-1}$ are accurate to Δt^{s+1}.

Lemma 2. *Given an s-order method then the initial time step $\Delta\sigma$ may be chosen to be $\Delta\sigma = \frac{\Delta t}{2^r}$ for $0 < \Delta t \leq 1$ and $r \in \mathbb{N}$. A sufficient value for r such that the method remains $\mathcal{O}(\Delta t^{s+1})$ is*

$$r = \left\lceil \log_2 \left(\Delta t^{-\left\lceil \frac{s}{2} - 1 \right\rceil} \right) \right\rceil \tag{2.24}$$

where $\lceil \cdot \rceil$ is the ceiling function.

Proof. To show this, first we must realize that the first order step $\mathcal{O}(\Delta\sigma^2)$ must be small enough so that it will be the same level of accuracy as the main step $\mathcal{O}(\Delta t^{s+1})$. So we want

$$\Delta\sigma^2 \leq \Delta t^{s+1} \tag{2.25}$$

but we also know that

$$\Delta\sigma = \frac{\Delta t}{2^r}. \tag{2.26}$$

The rest is as follows:

$$\left(\frac{\Delta t}{2^r}\right)^2 \leq \Delta t^{s+1} \tag{2.27}$$

$$\frac{\Delta t^2}{\Delta t^{s+1}} \leq 2^{2r} \tag{2.28}$$

$$\Delta t^{-(s-1)} \leq 2^{2r} \tag{2.29}$$

$$\log_2(\Delta t^{-\frac{1}{2}(s-1)}) \leq r. \tag{2.30}$$

Now we want $r \in \mathbb{N}$. Since $s \geq 1$ and $s \in \mathbb{N}$ then we can say

$$\frac{s}{2} - \frac{1}{2} = \begin{cases} n - \frac{1}{2} \geq \lfloor n - \frac{1}{2} \rfloor = n - 1 = \lceil n - 1 \rceil & \text{if } s = 2n, \text{for } n \in \mathbb{N} \\ n = \lceil n \rceil \geq \lceil n - \frac{1}{2} \rceil & \text{if } s = 2n+1, \text{for } n \in \mathbb{N} \end{cases} \tag{2.31}$$

then

$$-\frac{1}{2}(s-1) \leq -\left\lceil \frac{s}{2} - 1 \right\rceil. \tag{2.32}$$

Finally we can say

$$\log_2\left(\Delta t^{-\frac{1}{2}(s-1)}\right) \leq \log_2\left(\Delta t^{-\lceil\frac{s}{2}-1\rceil}\right) \leq \left\lceil \log_2\left(\Delta t^{-\lceil\frac{s}{2}-1\rceil}\right)\right\rceil = r \qquad (2.33)$$

and r will be a minimal sufficient value so as to ensure the method remains $\mathcal{O}(\Delta t^{s+1})$.

□

Once a sufficient value of $\Delta\sigma$ has been chosen and we are given ω^0, then we continue with the following steps:

Step 1. Use a first order method to obtain $\omega^1, \ldots, \omega^{s-1}$. A higher order method may be used if sufficient values are available. This brings us to the second line of Figure 2.3.

Step 2. Use the s most recent ω's to find $s-1$ more ω values. So if we were to start with $\omega^{n-s+1}, \ldots, \omega^n$, then after this step we would have $\omega^{n-s+1}, \ldots, \omega^{n+s-1}$. This is the 3rd and 4th lines of Figure 2.3.

Step 3. Take every second ω value starting with ω^0, double the step size and repeat Step 2 until $\Delta\sigma = \Delta t$. This is the last line of Figure 2.3.

Step 4. Begin the main loop of the algorithm.

From the steps above we can see that this start up procedure can be quite costly for higher order methods.

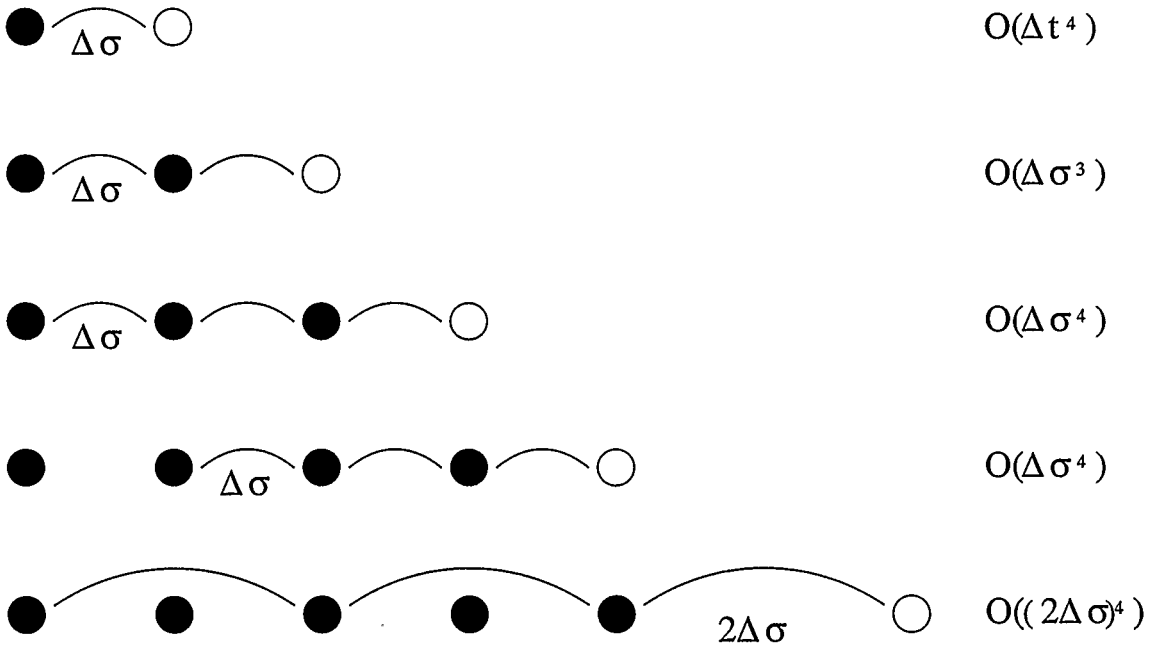

Figure 2.3: **Leapfrogging:** by the third line, there are enough values to use the 3rd order method. The 3rd order method must be used 2 more times before the $\Delta\sigma$ can be doubled.

2.5 Testing the method

Here we apply the pseudospectral method to a few selected problems to illustrate the performance of the algorithm.

2.5.1 Rotating cone problem

The first test that will be used is the rotating cone problem. As its namesake indicates this is simply a conical "spike" in a flat domain which will follow a given trajectory. Since we know or have constructed the trajectory we know the results at any given point and can then see how close our method is to the true solution. We can then use this as a reference point to see how the Lagrange-Galerkin method compares in both its fixed and adaptive forms. We look at the following advection-diffusion problem

(not the same as the N-S equations) for $\mathbf{u}$ [38, 51]:

$$\mathbf{u}_t + a \cdot \nabla \mathbf{u} - \nu \nabla^2 \mathbf{u} = \mathbf{g} \qquad \mathbf{x} \in \Omega, t \in [0, T] \tag{2.34}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0 \qquad \mathbf{x} \in \Omega \tag{2.35}$$

where $\mathbf{g} = 0$,

$$a = \begin{pmatrix} y - \pi \\ \pi - x \end{pmatrix}$$

and

$$\mathbf{u}_0 = \begin{cases} \cos^2 r & \text{for } r \leq \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.36}$$

where $r^2 = (2x - \pi)^2 + (2y - 2\pi)^2$. In the vorticity formulation (2.34) would look like

$$\omega_t + a \cdot \nabla \omega - \nu \nabla^2 \omega = \mathbf{g} \tag{2.37}$$

$$\omega(\mathbf{x}, 0) = \omega_0. \tag{2.38}$$

We use the following conditions: $\nu = 5 \times 10^{-16}, T = 2\pi$ with 20000 time steps to ensure that the method 'is stable'. Now T has been chosen so that the true solution ω is the same as the initial condition ω_0. The error measured is $\mathcal{E}_N = \|\omega - \vec{\omega}\|_{N,\infty}$, the infinity norm, where $\vec{\omega}$ is the computed solution for an N square grid and $\|\cdot\|_{N,\infty}$ is the discrete infinity norm, i.e. the maximum absolute value on the $N \times N$ grid. The results can be seen in Table 2.2. The '$*$' are cases where instability has caused the solution to blow up. The causes for this blow up will be examined below.

Order	1	2	3	4	5	6
$\mathcal{E}_{16}$	0.0218	0.0168	0.0168	0.0168	0.0166	0.0168
$\mathcal{E}_{32}$	0.0215	0.00372	0.00367	0.00367	0.00364	0.00367
$\mathcal{E}_{64}$	*	8.22×10^{-4}	7.95×10^{-4}	7.93×10^{-4}	7.88×10^{-4}	7.94×10^{-4}
$\mathcal{E}_{128}$	*	3.16×10^{-4}	2.31×10^{-4}	2.31×10^{-4}	2.47×10^{-4}	2.31×10^{-4}

Table 2.2: PS errors for the rotating cone problem

We can see that across the orders there is only a small decrease in the error. This is acceptable since the initial condition is only second order continuous. On the plus side we can see that when the grid size is doubled in each direction we obtain $\frac{1}{4}$ of the error from the previous step. This indicates that the dominant error for this problem is due to the discretization in space. So $\frac{1}{4}\mathcal{E}_N = \mathcal{E}_{2N}$ as expected.

Figure 2.4 shows the rotating cone for $N = 32, \nu = 10^{-16}$ and various steps as indicated. For the time steps indicated we are able to see some instabilities forming at the corners. The number of steps is chosen so that the instabilities are just beginning to be visible.

2.5.2 Steady sine function

Since the rotating cone problem in Section 2.5.1 used $(a \cdot \nabla)\mathbf{u}$ for the convective term rather than $(\mathbf{u} \cdot \nabla)\mathbf{u}$ then it is not really a good representation of how the convective term behaves in the Navier-Stokes equations. It is a good idea to look at another problem with a known solution that will test this method using the $(\mathbf{u} \cdot \nabla)\mathbf{u}$ term. This is accomplished by making use of the $\mathbf{g}$ value in (2.16). For each time step we set $\mathbf{g} = (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p - \nu\nabla^2\mathbf{u}$ or $\mathbf{g} = (\mathbf{u} \cdot \nabla)\omega + \nabla p - \nu\nabla^2\omega$. If we choose $\omega = 2\sin(x)\sin(y)$ then $\mathbf{g} = 4\nu\sin(x)\sin(y)$ which is what we get from (2.16) after

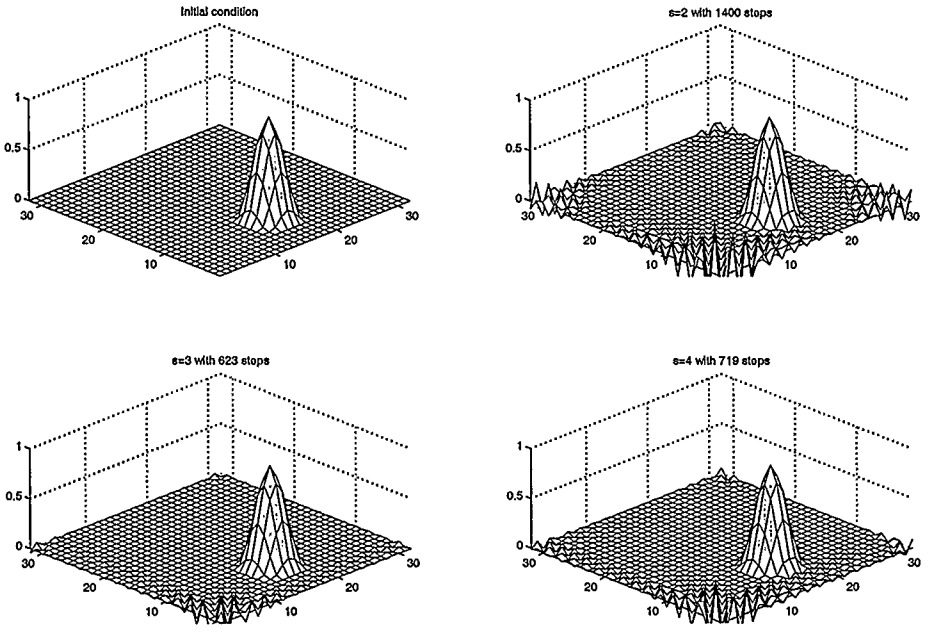

Figure 2.4: **Instability test:** an example of the rotating cone at three different orders. The minimum number of time steps were used, so that the instabilities were just beginning to form in the corners.

substituting and solving for **g**. The idea behind using **g** in this way is so that the true solution of the problem is simply the initial condition.

For the values $\nu = 5 \times 10^{-4}, T = 10$ and taking 2000 steps, the results can be seen in Table 2.3. As in Section 2.5.1, $\mathcal{E}_N$ is the infinity norm for N as defined and '$*$' are cases where the solution has blown up. The results in Table 2.3 are acceptable since they are all less than 10^{-8}. There is no change as N increases for orders 5 and 6 as the resolution in time overshadows the spatial resolution.

Order	1	2	3	4	5	6
$\mathcal{E}_{16}$	0.0184	0.0167	1.3	0.0244	285	209
$\mathcal{E}_{32}$	0.0403	0.039	1.3	0.0255	285	209
$\mathcal{E}_{64}$	22.7	0.014	1.3	0.0316	285	209
$\mathcal{E}_{128}$	*	0.00155	0.013	0.022	285	*

Table 2.3: PS errors ($\times 10^{-12}$) for the standing sine problem

2.6 Stability of the method

One of the true faults with the pseudospectral method (2.23) is that it is only conditonally stable, meaning that there is a minimum size for Δt such that the method will converge. Naturally this aspect of the pseudospectral method has triggered research into methods which allow for a correction of these instabilities. This correction has been accomplished via the addition of a corrective term [9] and through the use of different basis functions for the different parts of the equations [39]. Since the pseudospectral method is only being used as a basis for comparison in this case then these corrections need not be considered since the time steps will be sufficiently small.

First, we shall look at the example of the 4^{th} order pseudospectral method applied to the rotating cone problem, which was examined in Section 2.5.1. From the graph in Figure 2.5 we can see a sharp increase in the error when the number of steps has only been reduced by 10.

So we can say that the minimum number of steps required for $T = 2\pi$ is approximately 1650. In this case approximately means ± 2.5. When similar experiments are performed with different orders the results are as given in Table 2.4

What we would like to do is look at the stability regions of the rotating cone

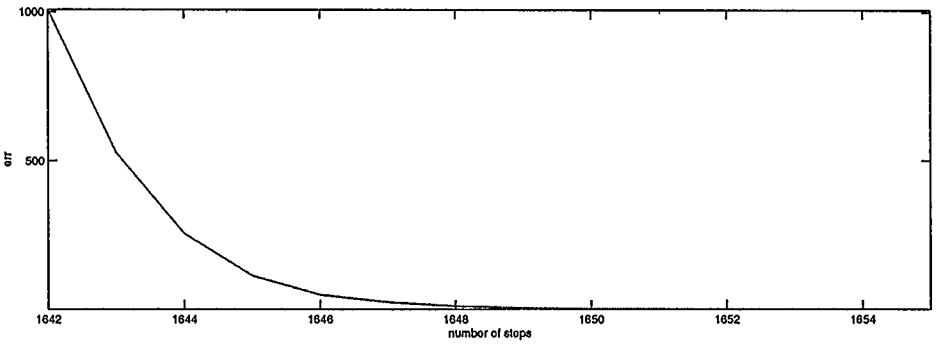

Figure 2.5: **Instability example** The errors inccurred by the rotating cone problem for the 4^{th} order pseudospectral method for 1642-1655 steps.

equation (2.34) or a suitably similar equation so that we may obtain some insight into the conditions that determine stability. For finite difference methods, stability is determined by the ratio $\frac{\Delta t}{\Delta x}$ using the Courant-Friedrichs-Lewy (CFL) condition. Since the derivatives are global in spectral methods rather than local as in finite difference methods, then the CFL condition and the domain of dependence triangles do not easily apply [32, 49]. What we can do instead is look at the eigenvalues of a suitable similar problem. For this purpose we choose a linear advection problem

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{2.39}$$

where a is a constant. As has been done for the pseudospectral method, a BDF

Order	2	3	4	5	6
min. steps	3360	1420	1650	2420	4110

Table 2.4: **Minimum steps for instability:** approximately the minimum number of steps required in order for instabilities to be noticeable in the corners of Figure 2.4, for $T = 2\pi$ and $N = 64$.

is used to solve for the time derivative and the Newton extrapolation scheme from Section 2.4 is used for the advective part of the equation. When the two s order schemes are applied to (2.39) we get

$$u^{m+1} - \sum_{j=0}^{s-1} \alpha_j u^{m-j} + a\Delta t \nabla \beta \sum_{j=0}^{s-1} \gamma_j u^{m-j} = 0, \tag{2.40}$$

where α_j and β are given according to Table 2.1, and γ_j can be found in Table 2.5 or can be defined as $\gamma_j = (-1)^{j-1}\binom{s}{j}$.

s	$\{\gamma_n, \ldots, \gamma_{n-s+1}\}$
2	$\{2, -1\}$
3	$\{3, -3, 1\}$
4	$\{4, -6, 4, -1\}$
5	$\{5, -10, 10, -5, 1\}$
6	$\{6, -15, 20, -15, 6, -1\}$

Table 2.5: Newton extrapolation coefficients

When (2.39) is transformed into Fourier space, it gives

$$\hat{u}^{m+1}(p) = \sum_{j=0}^{s-1} (\alpha_j - ia\Delta t p \beta \gamma_j) \hat{u}^{m-j}(p).$$

The characteristic polynomial for this equation can be given as

$$-\lambda^s + \sum_{j=0}^{s-1} \lambda^{s-1+j} (\alpha_j - ia\Delta t p \beta \gamma_j) = 0, \tag{2.41}$$

which when solved for $a\Delta tp$ gives

$$a\Delta tp = \frac{\lambda^s - \sum_{j=0}^{s-1} \alpha_j}{i\beta \left(\sum_{j=0}^{s-1} \gamma_j \right)}. \tag{2.42}$$

Define the stability region to be the region in the complex plane of values of $(ia\Delta tp)$, for which all the values of λ satisfying (2.41) lie within the unit circle. To plot the stability region we evaluate λ on the unit circle $e^{i\theta}$, and we end up with Figure 2.6. What we would like is for the eigenvalues of (2.40) to lie inside the regions plotted in Figure 2.6 [65]. Looking at Figure 2.6 we can see where z, the maximum value on the imaginary axis that is still inside the stability region, occurs and compare that to the experimentally observed stability boundary from Figure 2.4 and Table 2.4. From (2.36), a cannot be more than π and for that experiment $N = 64$, so $p \leq 32$. The results using the Δt values obtained via Table 2.4 can be seen in Table 2.6. All of the stability regions follow the imaginary axis for some interval, more or less closely. Only the 3$^{\text{rd}}$ and 4$^{\text{th}}$ orders include the axis inside the region, as seen in Figure 2.7. For the others the axis is only slightly outside the stability region immediately once you've left the origin.

It is worth noting that the results only differ by a constant value, so the results of Table 2.4 appear to match well with the computed values of z. For example

Order	2	3	4	5	6
z	*	0.63	0.54	*	*
$a\Delta tp$	0.19	0.44	0.38	0.26	0.15

Table 2.6: Comparison of z and $a\Delta tp$

when looking back to Figure 2.4 we see that given $s = 3$, $a\Delta tp = 0.51$ and given $s = 4$, $a\Delta tp = 0.44$. As expected these results are comparable to those in Table 2.6.

2.7 Variations on the pseudospectral method

Adaptivity in time could easily be implemented via a Runge-Kutta time step or an unevenly spaced variation on the BDF. Another option would be to follow the method described in Section 4.2.

Instead of using Fourier basis functions we could use wavelets to discretize in space. This method would allow for more flexibility in where the grid points are placed. More grid points could be placed where there is more activity in the domain and fewer where there is less activity [34, 42]. Another option for spatial discretization could be to use Legendre polynomials instead of the Fourier basis functions [2].

A filtered pseudospectral method could also be employed. At each time step all values of the vorticity field that are below a specified threshold value of $\mathbb{T}_z$ are set to zero. $\mathbb{T}_z$ must be carefully chosen for this method to work: too big and the energy of the system is lost; too little and the method is not effective [22]. The expected gain with this variation is a saving in space and time since we would only be working with sparse matrices.

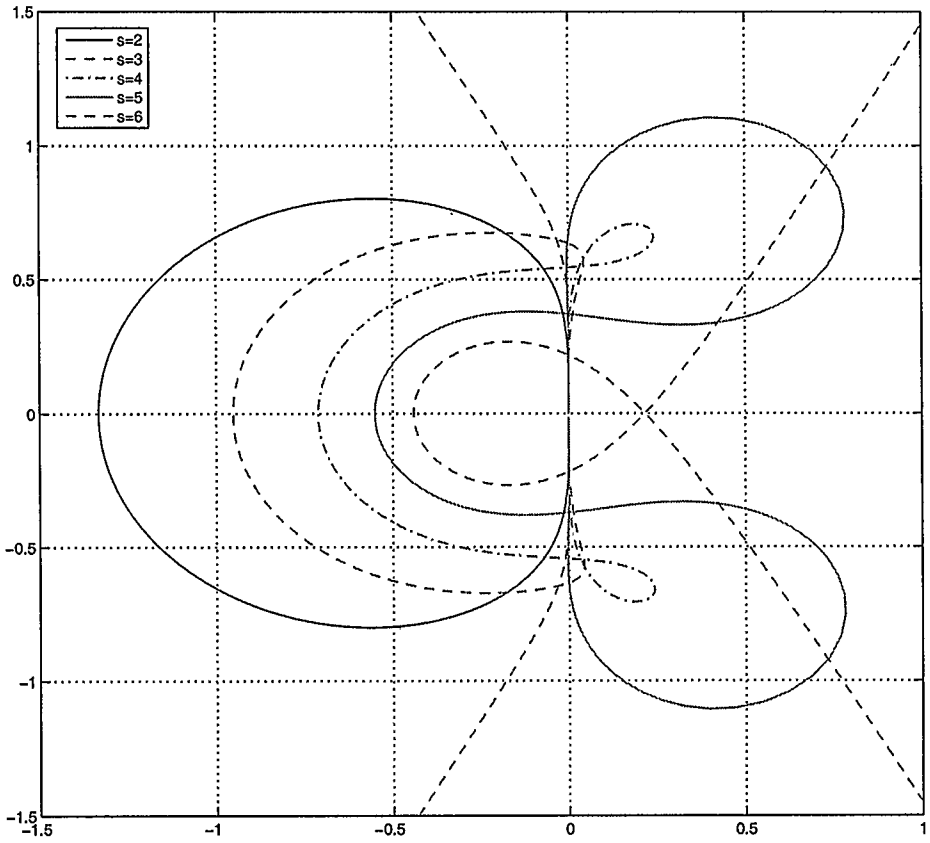

Figure 2.6: **Stability regions:** in order to have stability the eigenvalues of the equations (2.40) must lie inside the regions plotted above. Since we are considering an advection equation, then we are interested in how much of the imaginary axis lies inside the regions.

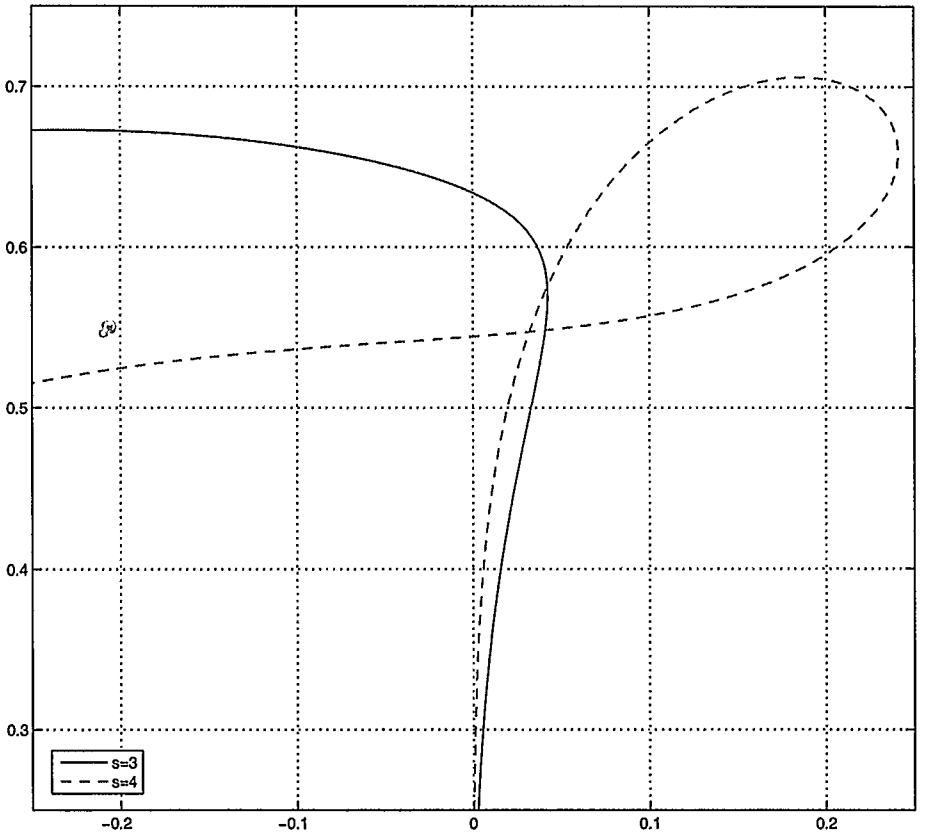

Figure 2.7: **A zoom of part of the imaginary axis:** $s = 3$ and 4 are the only regions that actually contain part of the imaginary axis, and the points of intersection can be seen above. These points of intersection are what determine the values z.

Chapter 3

The Lagrange-Galerkin method

This method is specifically designed for convection-dominated problems. For this reason semi-Lagrangian schemes have been used by the meteorological community for massive numerical simulations [21, 50] and more recently in computer animation to obtain more realistic fire and water effects [23, 24]. Due to its unconditional stability and convergence, the Lagrange-Galerkin (LG) method is immensely popular with finite element methods particularly with atmospheric sciences and other such areas. The advantage with finite element methods is that the grids can easily be adapted to fit awkward shapes [47, 54, 57, 61, 62]. The downside is that they are not as accurate as spectral methods. Since we have made the decision to deal with periodic grids then it seems more sensible to employ a spectral Lagrange-Galerkin method, thus maintaining a high level of accuracy [66, 67]. With these convection dominant problems we look for ways to simplify or remove the computation of the convective term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ that appears in the Navier-Stokes equations, as this is the part that causes the most computational difficulty. The method has a few advantages over the pseudospectral method. First, as we shall see the method does not have any stability issues, such as those seen in Section 2.6, provided the order is kept less than or equal to 6. This means that the size of the time step depends purely on the desired accuracy.

The main idea is to apply a change of variables to a coordinate system that will simplify the equation. In this case we transform the grid so that it matches the

charateristic curves, since in this system the convective term is zero. This means that we will be changing coordinates at each time step based on the progression of the flow. One way to look at the method is to start with a general advection-diffusion equation (2.34) [38]. First, for some trajectory $a(\mathbf{x}, t)$, we must solve

$$\frac{\partial}{\partial t}(X(\mathbf{x}, \sigma; t)) = a(X(\mathbf{x}, \sigma; t), t) \tag{3.1}$$

$$X(\mathbf{x}, \sigma; \sigma) = \mathbf{x}. \tag{3.2}$$

$X(\mathbf{x}, \sigma; t)$ is the path of the particle that passes through the position $\mathbf{x} \in \Omega$ at a time $\sigma \in (0, T]$ with the solution $X(\mathbf{x}, \sigma; t)$ given as follows

$$X(\mathbf{x}, \sigma; t) = \mathbf{x} + \int_{\sigma}^{t} a(X(\mathbf{x}, \sigma; \tau), \tau) \mathrm{d}\tau.$$

This will tell us the trajectories. We then look to evaluate the material derivative D_t, given by

$$\begin{aligned} D_t \mathbf{u}(\mathbf{x}, t) :=& \frac{\partial}{\partial t} a(X(\mathbf{x}, \sigma; t), t)|_{\sigma=t} \\ D_t \mathbf{u}(\mathbf{x}, t) =& \frac{\partial}{\partial t} \mathbf{u}(\mathbf{x}, t) + a(\mathbf{x}, t) \cdot \nabla \mathbf{u}(\mathbf{x}, t) \quad \forall \mathbf{x} \in \Omega, t \in (0, T]. \end{aligned}$$

3.1 Characteristic curves

In order to illustrate the procedure, we consider an example with the transport equation, which was first seen in Section 2.6,

$$u_t + a u_x = 0.$$

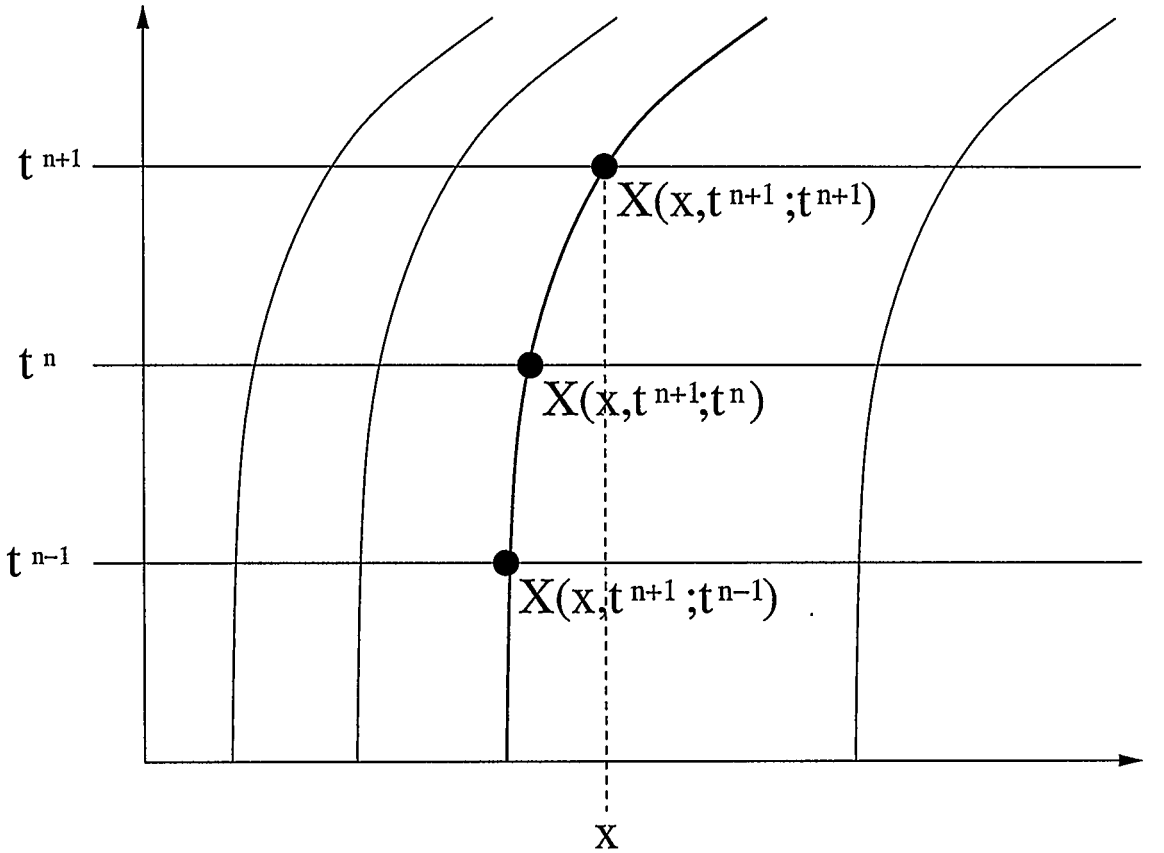

Figure 3.1: An example of characteristic curves

This can be rewritten using the chain rule to get

$$\frac{\mathrm{d}}{\mathrm{d}t}u(at+c,t)=0. \tag{3.3}$$

u is constant on the curves $(at+c,t)$. If we are given some initial condition $u(x,t_0)=u_0(x)$ then we choose the curve that passes through the initial point (x,t_0). The solution then has the form

$$u(x+a(t-t_0),t)=u_0(x)$$

$$u(x,t) = u_0(x - a(t - t_0)),$$

where $x - a(t - t_0)$ are the characteristic curves, or in our notation

$$X(\mathbf{x}, t_0; t) = \mathbf{x} + a(t - t_0).$$

If we return to our Navier-Stokes equations then the first task is to solve

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \mathbf{u}(X(\mathbf{x}, t^{n+1}; t), t), \tag{3.4}$$

where

$$X(\mathbf{x}, t^{n+1}; t^{n+1}) = \mathbf{x}. \tag{3.5}$$

To examine this in a graphical context consider Figure 3.1. If we are sitting at point $X(x, t^{n+1}; t^{n+1})$ and wish to know the value at $X(x, t^{n+1}; t^n)$, then we would look along the characteristic curve which we can obtain via (3.3) or the solution of (3.4) and (3.5). Then we simply follow the trajectory to the previous time step to get $X(x, t^{n+1}; t^n)$. This is what is known as the method of characteristics.

So the solution that we actually end up with is the position $X(x, t^{n+1}; t)$ that the characteristic curve $\mathbf{u}(X, t)$ will pass through at time t, if it also passes through the point $\mathbf{x}$ at time step t^{n+1}. This is often refered to as the backward-trajectory scheme, since we need to look backward from the regular grid at t^{n+1} to the previous position on the Lagrangian grid at t^n. To look forward from the regular grid at t^n to a Lagrangian grid at t^{n+1} is refered to as a forward-trajectory scheme [6, 50]. So at every time step we are recomputing the characteristics with respect to some fixed

initial grid $\mathbf{x}$. This will reduce the deformation that would occur in the non iterative case [59, 67]. To see how this helps to deal with the convection term we define a new function U to be

$$U(\mathbf{x},t) = \mathbf{u}(X(\mathbf{x},t^{n+1};t),t)$$
$$U(\mathbf{x},t^{n+1}) = \mathbf{u}(\mathbf{x},t^{n+1}),$$

applying the chain rule, then using (3.4), we obtain

$$\frac{\mathrm{d}U(\mathbf{x},t)}{\mathrm{d}t} = \mathbf{u}_t(X,t) + \frac{\mathrm{d}X}{\mathrm{d}t}\cdot\nabla\mathbf{u}(X,t) \tag{3.6}$$
$$\frac{\mathrm{d}U(\mathbf{x},t)}{\mathrm{d}t} = \mathbf{u}_t(X,t) + (\mathbf{u}\cdot\nabla\mathbf{u})(X,t). \tag{3.7}$$

Note that when $t = t^{n+1}$ then (3.7) is simply $\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}$. We can then rewrite (1.3) as

$$D_t\mathbf{u} = -\frac{1}{\rho}\nabla p(X) + \nu\nabla^2 U + \mathbf{g} \tag{3.8}$$

at t. If the pressure is constant and $\mathbf{g} = 0$, then (3.8) simplifies to

$$D_t\mathbf{u} = \nu\nabla^2 U.$$

For $t = t^{n+1}$ this can be written as

$$D_t\mathbf{u} = \nu\nabla^2\mathbf{u}. \tag{3.9}$$

This is similar to a heat equation for $\mathbf{u}$ with the exception that it involves a directional

derivative. This directional derivative is the reason that we cannot solve for a series solution as we would for a heat equation, and hence must use an iterative method instead.

3.2 Solving for the characteristics

If equation (3.9) is discretized in time then at each time step we would look to solve for $\mathbf{u}^{n+1} = \mathbf{u}(X(\mathbf{x}, t^{n+1}; t^{n+1}), t^{n+1})$. We shall use the following notation

$$\mathbf{u}^{j,j-k}(\mathbf{x}) = \mathbf{u}(X(\mathbf{x}, t^j; t^k), t^k),$$

with the two conventions

$$\mathbf{u}^{j,j-k} = \mathbf{u}^{j,j-k}(\mathbf{x})$$
$$\mathbf{u}^j = \mathbf{u}^{j,0}.$$

For reasons similar to those in Section 2.3 the BDF will be used. Assuming that the previous s values of $\mathbf{u}$ are available then our time discrete equations are

$$(I + \Delta t \beta \nu \nabla^2)\mathbf{u}^{n+1} = \sum_{k=1}^{s} \alpha_k \mathbf{u}^{n+1,k}.$$

In order to obtain the previous s values of $\mathbf{u}$ we must first know $X(\mathbf{x}, t^{n+1}; t)$ so that we may evaluate $\mathbf{u}^{n+1,k}$ for $k = 1, \ldots, s$. $X(\mathbf{x}, t^{n+1}; t)$ is defined by equations (3.4) and (3.5). Since the accuracy of the solution depends on X, then the higher the accuracy of the X value the more accurate the solution will be. To solve for X

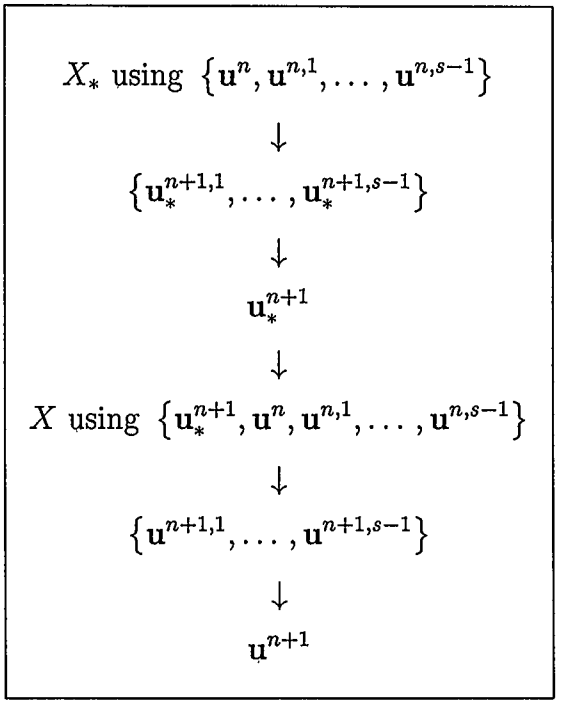

Figure 3.2: Predictor-Corrector algorithm

we use an Adams-Bashforth (A-B) with an Adams-Moulton (A-M) scheme. These methods are used together in what is known as a predictor-corrector algorithm. The predictor-corrector algorithm uses the A-B method to make an initial guess (predict) and then employs the A-M method to improve upon that initial guess (correct) [45]. The first time around, if we are at time step t^n, then we solve for X_*, an estimate of the current X, using the A-B method and the previous s values $\mathbf{u}^n, \mathbf{u}^{n,1} \ldots, \mathbf{u}^{n,s-1}$. We could even iterate this part several times in order to get a better value of X_*. In terms of Figure 3.2 this would involve iterating the first two steps. After the previous values $\mathbf{u}^{n,1}, \ldots, \mathbf{u}^{n,s-1}$ have been evaluated at this new X_*, then the new values $\mathbf{u}_*^{n+1,1}, \ldots \mathbf{u}_*^{n+1,s-1}$ are then used to solve for $\mathbf{u}_*^{n+1}$ using the BDF. This process

is then repeated only this time using $\mathbf{u}_*^{n+1}, \mathbf{u}^{n,1}, \ldots, \mathbf{u}^{n,s-1}$ and the A-M scheme to solve for X. Again this could be iterated to gain an improved value of X. This would correspond to repeating steps 4 and 5 in Figure 3.2. This new X is closer to the true value of X than X_*, as a result evaluating $\mathbf{u}^{n+1}$ using X will be closer to the true solution than evaluating $\mathbf{u}^{n+1}$ using X_*. The final step is to solve for $\mathbf{u}^{n+1}$ using $\mathbf{u}^{n+1,1}, \ldots, \mathbf{u}^{n+1,s-1}$ and the BDF.

3.2.1 Evaluating X

It is useful to note that for some $t^n, t^{n-1}, t^{n-2} \in [0, T]$,

$$X(\mathbf{x}, t^n; t^{n-2}) = X(X(\mathbf{x}, t^n; t^{n-1}), t^{n-1}; t^{n-2}). \tag{3.10}$$

In terms of the above notation we can interpret this as

$$\mathbf{u}^{n-1,1}(X(\mathbf{x}, t^n; t^{n-1})) = \mathbf{u}^{n,2}.$$

Now to take a closer look at how we solve for X using the previous $\mathbf{u}^n, \mathbf{u}^{n,1}, \ldots, \mathbf{u}^{n,s-1}$ values. To solve for $X(\mathbf{x}, t^{n+1}; t^n)$ from (3.4) we need to evaluate

$$\mathbf{x} - X(\mathbf{x}, t^{n+1}; t) = \int_{t^n}^{t^{n+1}} \mathbf{u}(X(\mathbf{x}, t^{n+1}; \tau), \tau) \mathrm{d}\tau. \tag{3.11}$$

This is done with the approximation

$$\mathbf{x} - X(\mathbf{x}, t^{n+1}; t) \approx \sum_{j=n-s+1}^{n} a_j \mathbf{u}^{n,n-j}. \tag{3.12}$$

where a_j corresponds to the appropriate Adams coefficient. Note that the integral (3.11) is from t^n to t^{n+1}. So for (3.12)

$$\{\mathbf{u}^n, \mathbf{u}^{n,1}, \ldots, \mathbf{u}^{n,s-1}\}$$

are the $\mathbf{u}$ values that are used. These values of $\mathbf{u}$ values are obtained by taking $\mathbf{u}(\mathbf{x}, t^{n-j})$ for $j = 0, \ldots, s$ and pushing them along the trajectory X to t^n. This is done at every step so that the stored list of the previous values is always up to date. To illustrate this idea, consider a list of values at a time t^n where $s = 3$,

$$\{\mathbf{u}^n, \mathbf{u}^{n,1}, \mathbf{u}^{n,2}\}.$$

To find $\mathbf{u}^{n+1}$, the first step would be to evaluate all the elements of the list at $X(\mathbf{x}, t^{n+1}; t^n)$, and using (3.10) to get

$$\{\mathbf{u}^{n+1,1}, \mathbf{u}^{n+1,2}, \mathbf{u}^{n+1,3}\}.$$

Now that the list is updated to reflect the time t^{n+1}, it may be used to evaluate $\mathbf{u}(\mathbf{x}, t^{n+1})$. $\mathbf{u}(\mathbf{x}, t^{n+1})$ is then queued into the front of the list,

$$\{\mathbf{u}^{n+1}, \mathbf{u}^{n+1,1}, \mathbf{u}^{n+1,2}\}.$$

The list can now be used to evaluate the next time step.

What has not yet been discussed is how when we have X do we evaluate $\mathbf{u}^{j,j-k}(X)$ using $\mathbf{u}^{j,j-k}(\mathbf{x})$. Since the points are irregularly spaced, we cannot use the IFFT, so

for now we need to do it the long way with

$$\mathbf{u}^{n-j}(X) = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \hat{\mathbf{u}}_{p,q}^{n-j}\, \mathrm{e}^{ipX_1+iqX_2} \qquad j = 0, \ldots, s-1$$

then use the FFT to return to $\mathbf{u}^n(X, t^n)$, then solve for $\mathbf{u}^{n+1}$ using the BDF. This way it would take $\mathcal{O}(N^4)$ operations. However there are ways to approximate this unevenly spaced FT that can be accomplished in little more time than it takes to evaluate the regular FFT.

3.3 Approximate FT using Taylor polynomials

There are several methods to approximate the FT for irregularly spaced data. A good overview of these various methods can be seen in [68] with some newer methods described in [26, 33]. In this thesis, Taylor polynomials are used to approximate the value of the function $\mathbf{u}$ at an irregular grid point $\mathbf{x}_l$ around the regularly spaced grid points $\bar{\mathbf{x}}_n$. This method is based on the principle that in one dimension we can write a truncated Taylor expansion of $\mathbf{u}$ as

$$\mathbf{u}(x_l) = \sum_{k=0}^{K} (I_N \mathbf{u})^{(k)}(x_l) \frac{z_l^k}{k!}, \quad z_l = \frac{x_l - \bar{x}_n}{r_x}.$$

As usual the derivatives $\mathbf{u}^{(k)}$ are done in Fourier space and r_x is the maximum deviation of the points x_l from $\bar{x}_n$.

This algorithm can be divided into three parts. The first part we need only find the maximum deviation in the x and y direction from the nearest regular grid $\bar{\mathbf{x}}_n$.

The maximum deviation in the x and y direction are denoted by r_x and r_y and are defined as follows

$$r_x = \max_l \min_n |x_l - \bar{x}_n|$$
$$r_y = \max_l \min_n |y_l - \bar{y}_n| .$$

Since the domain Ω is periodic, we must have $r_x \leq \frac{\Delta x}{2}$ and $r_y \leq \frac{\Delta y}{2}$ where Δx is the distance between the grid points in $\bar{x}_n$ in the x direction, and Δy the distance in the y direction. In two dimensions this can be done in $\mathcal{O}(N^2)$ operations. For the second part of the algorithm we calculate

$$\mathbf{u}_n^{j,k} = \sum_{p=-\frac{M}{2}}^{\frac{M}{2}}{}'' \sum_{q=-\frac{N}{2}}^{\frac{N}{2}}{}'' \frac{(ip)^j (iq)^k}{j!k!} \hat{\mathbf{u}}_{p,q} e^{i(p\bar{x}_n + q\bar{y}_n)} \quad j,k = 0,\ldots,K.$$

This is done by applying the IFFT to the matrix with elements given by $\frac{(ip)^j (iq)^k}{j!k!}\hat{\mathbf{u}}_{p,q}$. This is the most time consuming part of the algorithm taking $\mathcal{O}(2K^2N^2 \log N)$ operations where N is the grid size, and where K refers to the number of Taylor coeffecients that are used. The last step is to compute

$$\mathbf{u}(\mathbf{x}_l) \approx \sum_{j,k=0}^{K} \mathbf{u}_n^{j,k} \mathbf{z}_l^{j,k}$$

where

$$\mathbf{z}_l^{j,k} = \left(\frac{x_l - \bar{x}_n}{r_x}, \frac{y_l - \bar{y}_n}{r_y} \right).$$

This final step is $\mathcal{O}(K^2N^2)$. What must be decided ahead of time is how big to

make K. In order for this algorithm to be effective we need to have $K \ll N$. The appropriate size of K can be determined from the maximum error

$$E_p = \frac{(|p| \max(r_x, r_y))^{K+1}}{(K+1)!},$$

which is essentially just the maximum of the next term in the Taylor expansion. K is chosen such that E_p is less than a predetermined tolerance $\mathbb{T}$. For the overall method the number of operations at $\mathcal{O}(2K^2N^2 \log N)$ has the potential to take a lot longer than a regular FFT of the same size. In practice K is quite small in comparison to N. For example $K \approx 8$ for a 64×64 grid and $\mathbb{T} = 10^{-3}$. Approximating the FT in this way will usually run faster for smaller time steps since r_x and r_y will often be smaller in those cases. As a result, E_p will be less than $\mathbb{T}$ for smaller values of K. While this may be true in practice there is no way to determine exactly how much time is saved since that would require knowing X ahead of time.

3.4 Divergence free space

As mentioned in the previous chapter, when the transformation to the vorticity formulation is made, we see that ω is automatically incompressible. That is to say that ω is divergence free, or that ω exists in a divergence free space. For the Lagrange-Galerkin method it is not a given that the solution $\mathbf{u}$ is automatically divergence free, especially since we will be transforming $\mathbf{u}$ so that it follows the characteristic curves. To ensure that $\mathbf{u}$ is still in a divergence free space, we apply an operator R to $\mathbf{u}$ at each time step, so that $R\mathbf{u}$ is the projection of $\mathbf{u}$ onto the divergence free

zero-mean space V [66]. Define

$$V = \{\mathbf{u} \in \mathcal{H}^1 | \nabla \cdot \mathbf{u} = 0 \text{ and } \bar{\mathbf{u}} = 0\}$$

where $\bar{\mathbf{u}}$ is the mean of $\mathbf{u}$. The Hilbert space $\mathcal{H}^\sigma$ is given as

$$\mathcal{H}^\sigma(\Omega) = \{\mathbf{u} | D^\alpha \mathbf{u} \in L^2(\Omega), 0 \leq |\alpha| \leq \sigma\}$$

where D^α for $\alpha = (a_1, \ldots, a_d)$, is the derivative in the sense of periodic distributions defined as

$$D^\alpha \mathbf{u} = \frac{\partial^{a_1 + \cdots + a_d} \mathbf{u}}{\partial^{a_1} \ldots \partial^{a_d}}.$$

Now define

$$R = (I - \nabla(\nabla^2)^{-1}\nabla\cdot).$$

To see that this projection works, note that in Fourier space R is

$$\begin{aligned}\hat{R} &= \left(I - \begin{pmatrix} -ip \\ -iq \end{pmatrix} \left(\frac{-1}{p^2+q^2}\right) \begin{pmatrix} -ip \\ -iq \end{pmatrix}\right) \\ &= \left(I - \frac{1}{p^2+q^2} \begin{pmatrix} p^2 & pq \\ pq & q^2 \end{pmatrix}\right)\end{aligned}$$

where $p^2 + q^2 \neq 0$. Take $\hat{\mathbf{u}}$ to be $\binom{u_{pq}}{0}$ then

$$\hat{R}\hat{\mathbf{u}} = \begin{pmatrix} u_{pq} \\ 0 \end{pmatrix} - \frac{1}{p^2+q^2} \begin{pmatrix} p^2 u_{pq} \\ pq u_{pq} \end{pmatrix}$$

$$= \frac{u_{pq}}{p^2+q^2}\begin{pmatrix} q^2 \\ -pq \end{pmatrix}$$

and

$$\nabla \cdot \hat{R}\hat{\mathbf{u}} = \frac{u_{pq}}{p^2+q^2}(q^2 ip - pqiq) = 0.$$

$R\mathbf{u}$ lies in the divergence free space V as required. A similar demonstration will yield similar results for $\hat{\mathbf{u}} = \binom{0}{v_{pq}}$. For a general $\mathbf{u}$ we look at the span of $\mathbf{u}$ using $R\binom{u_{pq}}{0}$ and $R\binom{0}{v_{pq}}$. What this means is that instead of worrying about whether or not we are in the space V, all that needs to be done is to apply the projection R to the whole equation and the result will follow.

3.5 Some test problems

For Sections 3.5.1 and 3.5.2, we apply the Lagrange-Galerkin method to two problems with known solutions to illustrate the behavior of the method. Specifically, we are looking for a demonstration of the order of convergence. For Sections 3.5.3 and 3.5.4, the Lagrange-Galerkin method is applied to two fluid flow problems for which a reference solution is computed. These problems are of interest since they are designed to have steep gradients, which can often be difficult to compute.

3.5.1 Rotating cone problem

This is the same problem that was presented in Section 2.5.1. For this method we do not need as many steps as the pseudospectral method since we have no need to worry about stability. The difference in this case is that we look to solve (2.34) instead of

Order	1*	1	2	3	4	5	6
$\mathcal{E}_{\frac{\pi}{4}}$	0.000752	0.000973	0.000973	0.000977	0.000972	0.000976	0.000967
$\mathcal{E}_{\frac{5\pi}{4}}$	0.000752	0.000965	0.000965	0.000983	0.000984	0.000979	0.000975
$\mathcal{E}_{2\pi}$	0.00062	0.00135	0.00135	0.00134	0.00135	0.00136	0.00137

Table 3.1: LG errors for order versus T

Order	1*	1	2	3	4	5	6
$\mathcal{E}_{\frac{\pi}{4}}$	0.000453	0.000648	0.000654	0.000656	0.000655	0.000658	0.000686
$\mathcal{E}_{\frac{5\pi}{4}}$	0.000562	0.000498	0.000496	0.000489	0.000486	0.000531	0.000574
$\mathcal{E}_{2\pi}$	0.00186	0.000312	0.000172	0.000166	0.000165	0.000168	0.000168

Table 3.2: LG errors for order versus T with $\nu = 5 \times 10^{-4}$

(2.37) as was done in Section 2.5.1.

Initially we will have $\nu = 5 \times 10^{-16}, \mathbb{T} = 10^{-2}, N = 64$ and measure the error $\mathcal{E}_T$ in the infinity norm, i.e. $\mathcal{E}_T = \|\mathbf{u} - \vec{\mathbf{u}}\|_{T,\infty}$, where $\mathbf{u}$ is the exact solution at time T. The results are recorded in Table 3.1. All results were acheived using 10 time steps, with the exception of the first column (1*) where only one time step was used.

Results from the same problem but with some diffusion can be achieved by setting $\nu = 5 \times 10^{-4}$. These results can be seen in Table 3.2. The extra column (1*) is displayed as an additional illustration of the unconditional stability of the method. Since the trajectories are given for this problem, and are therefore exact to within machine accuracy, then the error that occurs is from the solution of the parabolic equation (3.9) and the approximate evaluation of $\mathbf{u}^{j,j-k}(X)$. When fewer time steps are taken, fewer errors accumulate from the successive iterations of (3.9). Then the smallest cumulative error occurs when only a single step is taken as is shown in

Tables 3.1 and 3.2 [66].

Another check is to compare the error of the solution as the size N increases, where the error is defined as $\mathcal{E}_N = \|\mathbf{u} - \vec{\mathbf{u}}\|_{N,\infty}$. The conditions for the results seen in Table 3.3were as follows: $\nu = 5 \times 10^{-16}, \mathbb{T} = 10^{-2}$, and 10 time steps. In Table

Order	1	2	3	4	5	6
$\mathcal{E}_{16}$	0.0237	0.0237	0.0196	0.0196	0.0200	0.0202
$\mathcal{E}_{32}$	0.00542	0.00542	0.00555	0.00557	0.00558	0.00565
$\mathcal{E}_{64}$	0.00136	0.00136	0.00136	0.00136	0.00138	0.00138
$\mathcal{E}_{128}$	0.000338	0.000338	0.000300	0.000300	0.000302	0.000302

Table 3.3: LG errors for order versus N

Order	1	2	3	4	5	6
$\mathcal{E}_{16}$	0.0237	0.0237	0.0196	0.0196	0.0200	0.0202
$4\mathcal{E}_{32}$	0.0217	0.0217	0.0222	0.0223	0.0223	0.0226
$16\mathcal{E}_{64}$	0.0218	0.0218	0.0217	0.0218	0.0221	0.0221
$64\mathcal{E}_{128}$	0.0216	0.0216	0.0192	0.0192	0.0193	0.0193

Table 3.4: Comparison of LG errors for order versus N

3.4 we can see that when the grid size is doubled the error is reduced by a factor of 4. With the trajectories given exactly the errors are then only restricted by the grid size.

3.5.2 Steady sine function

This is the same problem as described in Section 2.5.2 with the same conditions, $T = 1$, $\nu = 5 \times 10^{-4}$, $\mathbb{T} = 10^{-3}$, except in this case order is being compared against increasing step sizes. All results can be seen in Table 3.5. The order of convergence

Order	1	2	3	4	5	6
$\mathcal{E}_{20}$	0.454	0.154	0.0345	0.00931	0.00278	0.000704
$\mathcal{E}_{40}$	0.238	0.0446	0.00556	0.000813	0.000108	1.99×10^{-5}
$\mathcal{E}_{80}$	0.122	0.0120	0.000789	5.81×10^{-5}	3.53×10^{-6}	7.86×10^{-7}
$\mathcal{E}_{160}$	0.0616	0.00312	0.000105	3.8×10^{-6}	1.1×10^{-7}	2.01×10^{-8}
$\mathcal{E}_{320}$	0.0309	0.000796	1.35×10^{-5}	2.4×10^{-7}	6.97×10^{-8}	5.72×10^{-8}

Table 3.5: LG errors for the standing sine problem

of the method can be observed in orders 1,2 and 3, meaning that for a given order s if the number of steps is doubled than the resulting error is 2^{-s} times the previous error. The value of the tolerance $\mathbb{T}$ makes errors less than 10^{-4} difficult to analyse due to the dominance of the spatial discretization, and as a result the convergence is not noticeable for the higher orders.

3.5.3 Vortex convergence problem

This is the three vortex convergence problem as described in [22]. For this problem we have three vorticies in the center of the domain. This problem is designed so that the three vorticies will converge into two after a certain amount of time. Given the method we need to ensure that we will get reasonable results; i.e. we mean that the relative error using the L^2 norm $\mathcal{E}_N = \|\mathbf{u} - \vec{\mathbf{u}}\|_{N,2}$, where $\mathbf{u}$ is the reference solution and $\vec{\mathbf{u}}$ is the result obtained from the method, is such that $\mathcal{O}(\mathcal{E}_N) = \mathcal{O}(10^{-2})$. The reference solution in this case is not the same as the true solution, but is sufficiently close for our purposes. What is really being used for $\mathbf{u}$ is the result of the pseudospectral method obtained using a very large number of time steps ($\approx 10^6$).

The initial condition for this problem is defined by letting

$$\omega(x,y) = \sum_{j=1}^{3} A_j e^{-r_j}$$

where

$$r_j = \frac{1}{\varsigma^2}\left((x-x_j)^2 + (y-y_j)^2\right),$$

the weights are $A_1 = A_2 = \pi, A_3 = -\frac{2}{3}\pi$ and the points are $\mathbf{x}_1 = \left(\frac{3\pi}{4}, \pi\right), \mathbf{x}_2 = \left(\frac{5\pi}{4}, \pi\right)$ and $\mathbf{x}_3 = \left(\frac{5\pi}{4}, \pi\left(1+\frac{\varsigma}{2}\right)\right)$ with $\varsigma = \frac{1}{\pi}$.

For these results the following conditions were used: $T = 40, \nu = 5 \times 10^{-4}, N = 128, \mathbb{T} = 10^{-2}$. The order s was compared to the number of time steps and recorded in Table 3.6. Figure 3.3 shows the progress of the flow for $s = 3$ and 400 time steps. Figure 3.4 shows $\mathcal{E}$ plotted versus the number of steps taken on a $\log_2$ scale. The higher orders ($s = 5$ and 6) show no useful information since they immediately attain the maximum resolution for the conditions used. The other orders display the order of convergence of the method. For $s = 2$ we can see that the slope is about 2, in terms of convergence, as we would expect it to be. The $s = 3$ slope seems to be a bit steeper than the expected value of 3. In fact, with a value of about $\frac{10}{3}$, it is almost the same as the slope for $s = 4$. This would seem to indicate that $s = 4$ did

Order	2	3	4	5	6
$\mathcal{E}_{100}$	0.527	0.434	0.267	0.023	0.0189
$\mathcal{E}_{200}$	0.15	0.0371	0.0240	0.00103	0.000752
$\mathcal{E}_{400}$	0.0424	0.00383	0.00209	0.000319	0.000292
$\mathcal{E}_{800}$	0.0101	0.000554	0.000471	0.000345	0.000364

Table 3.6: LG errors for the vortex problem

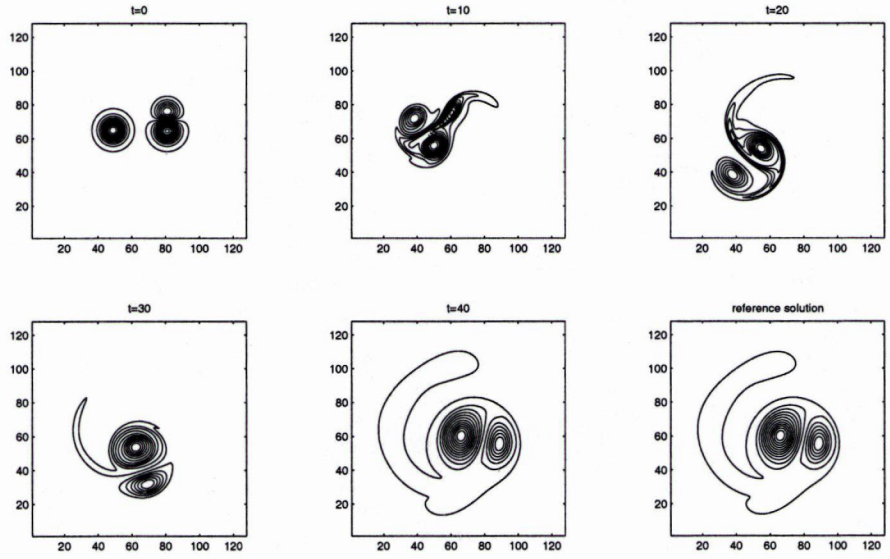

Figure 3.3: **LG solution of the vortex problem:** contours of the vorticity $\omega(\mathbf{x}, t)$ of the solution of the vortex problem using the LG method, with $s = 3$ and 400 time steps, at selected times t. There is no discernable difference between the LG solution and the reference solution.

a little worse than expected while $s = 3$ did better than expected.

3.5.4 Stream vortex problem

The initial condition for this problem is a rapidly moving stream in the middle of a nearly stationary fluid. The boundary between the stream (middle) and the edges (top and bottom) is meant to be tight as this will cause vortices to form and collide. As in Section 3.5.3 the reference solution is actually the result of the pseudospectral method with a very large number of time steps used.

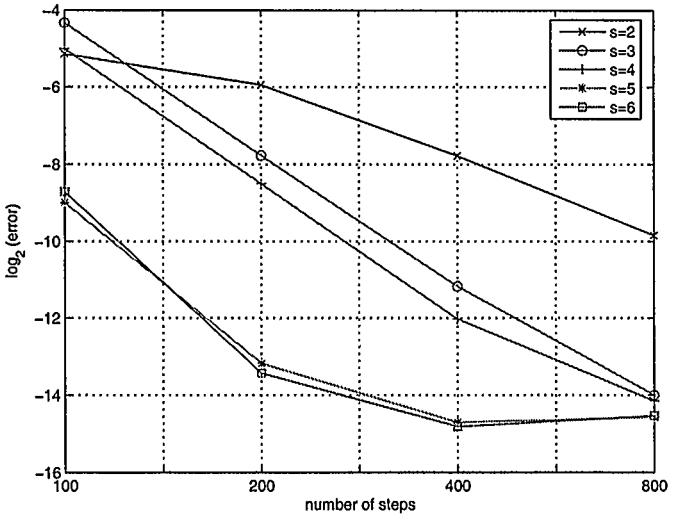

Figure 3.4: $\log_2 \mathcal{E}$ **of the vortex errors:** a plot of the $\log_2 \mathcal{E}$ of the errors for the vortex problem. We would expect to see the negatives of the slopes of the lines to match the order. The resolution of the method limits the use of the plot for $s = 5$ and 6.

The initial condition is given in [66] and is described as follows,

$$\omega(x,y) = \operatorname{sech}^2(10(y-a)) - \operatorname{sech}^2(10(y-b)) + 10^{-2}\left(\cos 4x + \frac{\cos 2x}{10}\right),$$

where $a = \pi\left(1 - \frac{1}{8}\right)$ and $b = \pi\left(1 + \frac{1}{8}\right)$.

For these results the following conditions were used: $T = 150, \nu = 5 \times 10^{-4}, N = 128, \mathbb{T} = 10^{-2}$. The order s was compared to the number of time steps and recorded in Table 3.7. Figure 3.5 shows the progress of the flow for $s = 3$ and 100 time steps. Overall, Table 3.7 seem to show faster convergence and smaller errors when compared to Table 3.6. This can be attributed to the fact that the initial condition in Section 3.5.3 is not as smooth as the one in this section. Figure 3.6 tells a similar

story to Figure 3.4. Orders $s = 5$ and 6 once again give little information about the order of convergence as $\Delta t \to 0$, since the errors are quickly dominated by the spatial discretization errors. For $s = 2$ and $s = 3$ the slopes look to be almost exactly 2 and 3 respectively, while $s = 4$ has a slope of approximately $\frac{11}{3}$.

3.6 Stability of the method

Being able to get a result in Section 3.5.1 by only taking a single time step, as seen in column 1* of Tables 3.1 and 3.2, is a result of the unconditional stability of the method. A method is unconditionally stable if there exists a C, that depends only on the divergent velocity field $(\nabla \cdot a)$, such that

$$\|\mathbf{u}^n\| \leq e^{Ct^n} \|\mathbf{u}^0\|, \quad \forall n \geq 0.$$

Note that there is no restriction on Δt, such as in Section 2.6. A sketch of the proof of the unconditional stability, as related to equations (2.34) a (2.35), shall be presented here. This presentation will only be for the exact Galerkin method, meaning that all of the integrals are computed exactly. For more details see [66, 67], where Ware established stability for the fully discrete method for linear problems.

Order	2	3	4	5	6
$\mathcal{E}_{100}$	0.0424	0.0111	0.0188	0.00462	0.00541
$\mathcal{E}_{200}$	0.00821	0.00108	0.00124	0.00132	0.00149
$\mathcal{E}_{400}$	0.00221	0.000298	0.000108	7.37×10^{-6}	1.31×10^{-5}
$\mathcal{E}_{800}$	0.000542	4.55×10^{-5}	8.06×10^{-6}	6.64×10^{-7}	1.41×10^{-6}

Table 3.7: LG errors for the stream vortex problem

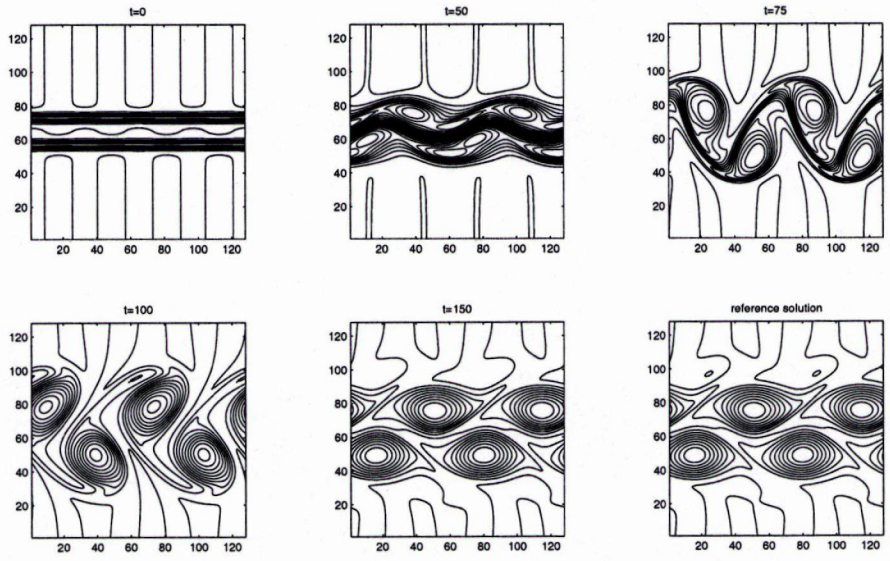

Figure 3.5: **LG solution of the stream vortex problem:** contours of the solution of the vorticity $\omega(\mathbf{x}, t)$ of the stream vortex problem using the LG method, with $s = 3$ and 100 time steps, at selected times t. There is some distortion between the computed solution and the reference solution, but considering that only 100 steps are being used the solution looks good.

The following lemma will be useful for the completion of the proof.

Lemma 3. *let a be a sufficiently-smooth velocity field. If J is the Jacobian for the change of variables from $\mathbf{x}$ to X, where X is defined by equations* (3.1) *and* (3.2), *and $\nabla \cdot a = 0$, then $J = I$.*

This lemma could be considered a corollary of Liouville's Theorem and so the proof is similar [46].

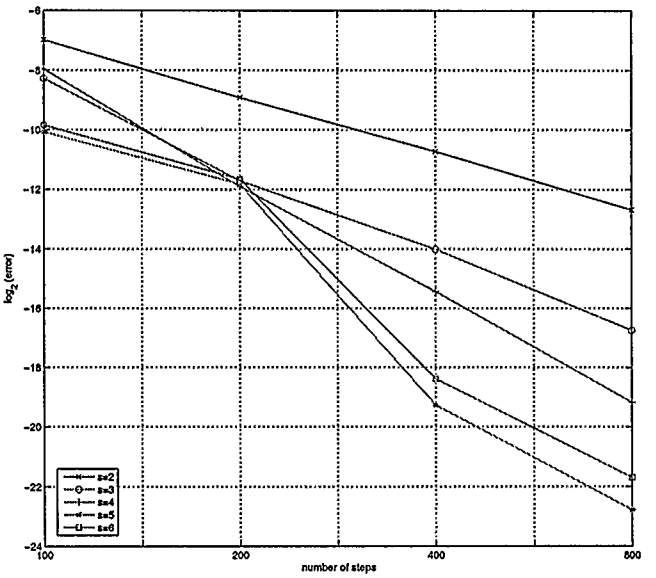

Figure 3.6: $\log_2 \mathcal{E}$ **of the stream vortex errors:** a plot of $\log_2 \mathcal{E}$ of the errors in Table 3.7. With the exception of $s = 5$ and 6, the negatives of the slopes of the lines are approximately equal to the order.

Proof. Since $J_{jk} = \frac{\mathrm{d}x_j}{\mathrm{d}X_k}$, define $\mathcal{J}(\mathbf{x}, t) = \frac{\partial X(\mathbf{x}, s; t)}{\partial \mathbf{x}}$. Then from (3.1)

$$\begin{aligned}
\frac{\partial}{\partial t}\mathcal{J}(\mathbf{x}, t) &= \frac{\partial}{\partial t}\frac{\partial}{\partial \mathbf{x}}X(\mathbf{x}, s; t) \\
&= \frac{\partial}{\partial \mathbf{x}}a(X(\mathbf{x}, s; t), t) \\
&= \nabla a(X(\mathbf{x}, s; t), t) \cdot \frac{\partial}{\partial \mathbf{x}}X(\mathbf{x}, s; t) \\
&= \nabla a(X(\mathbf{x}, s; t), t) \cdot \mathcal{J}(\mathbf{x}, t).
\end{aligned}$$

Now if we expand $\mathcal{J}(\mathbf{x}, t+h)$ about t,

$$\begin{aligned}\mathcal{J}(\mathbf{x}, t+h) &= \mathcal{J}(\mathbf{x}, t) + h\frac{\partial \mathcal{J}(\mathbf{x}, t)}{\partial t} + \mathcal{O}(h^2) \\ &= \mathcal{J}(\mathbf{x}, t) + h\nabla a(X(\mathbf{x}, s; t), t) \cdot \mathcal{J}(\mathbf{x}, t) + \mathcal{O}(h^2).\end{aligned}$$

Then by multiplying both sides by $\mathcal{J}^{-1}(\mathbf{x}, t)$ and taking the determinant, the right side is then equivalent to $1 + h\nabla \cdot a(X(\mathbf{x}, s; t), t) + \mathcal{O}(h^2)$. Following the definition of a derivative

$$\begin{aligned}\lim_{h\to 0} \frac{1}{h} \det\left((\mathcal{J}(\mathbf{x}, t+h) - \mathcal{J}(\mathbf{x}, t))\mathcal{J}^{-1}(\mathbf{x}, t)\right) &= \lim_{h\to 0} \frac{h\nabla \cdot a(X(\mathbf{x}, s; t), t) + \mathcal{O}(h^2)}{h} \\ &= \nabla \cdot a(X(\mathbf{x}, s; t), t).\end{aligned}$$

Since $\nabla \cdot a = 0$ then $\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{J}(\mathbf{x}, t) = 0$ and finally, since J is initially I, we get $J = I$. □

One last definition will be needed before continuing. Recall the notation from Section 3.2, then define the operator E to be such that $E\mathbf{u}^n = \mathbf{u}^{n,1}$.

Theorem 1. *For equation* (2.34)*, where a is linear, and for the method described in this chapter, we have* $\|\mathbf{u}^{n+1}\| \leq \|\mathbf{u}^n\|$*, where the norm used is the* L^2 *norm defined by* (2.3).

Proof. Using the definition $\mathbf{u}^{n+1} = P_N E\mathbf{u}^n$, then from (2.4) we can say

$$\langle \mathbf{u}^{n+1}, \mathbf{v}\rangle = \langle E\mathbf{u}^n, \mathbf{v}\rangle \quad \forall \mathbf{v} \in S_N.$$

The use of P_N implies that $\mathbf{u}^{n+1} \in S_N$, then we can say the following

$$\begin{aligned} \langle \mathbf{u}^{n+1}, \mathbf{u}^{n+1} \rangle = \langle E\mathbf{u}^n, \mathbf{u}^{n+1} \rangle \quad &\Rightarrow \\ \left\| \mathbf{u}^{n+1} \right\|^2 \leq \left\| E\mathbf{u}^n \right\| \left\| \mathbf{u}^{n+1} \right\| \quad &\Rightarrow \\ \left\| \mathbf{u}^{n+1} \right\| \leq \left\| E\mathbf{u}^n \right\| . \end{aligned}$$

From the definition (2.3), with $|J|_\infty$ meaning the maximum absolute value in the matrix J,

$$\begin{aligned} \left\| E\mathbf{u}^n \right\|^2 &= \int_\Omega \left| \mathbf{u}^n(X(\mathbf{x}, t^{n+1}; t^n), t^n) \right|^2 \mathrm{d}\mathbf{x} \\ &= \int_{E\Omega} \left| \mathbf{u}^n(X, t^n) \right|^2 |J| \, \mathrm{d}X \\ &\leq |J|_\infty \int_{E\Omega} \left| \mathbf{u}^n(X, t^n) \right|^2 \mathrm{d}X \\ &= |J|_\infty \int_\Omega \left| \mathbf{u}^n(X, t^n) \right|^2 \mathrm{d}X \text{ (by periodicity)} \\ &= |J|_\infty \left\| \mathbf{u}^n \right\|^2 . \end{aligned}$$

Applying Lemma 3 the desired conclusion is reached. □

These results can be extended to the N-S equations.

3.7 Variations on the Lagrange-Galerkin method

There are many possible variations that could be applied to the Lagrange-Galerkin method, a few of which are described in this section. Since adaptivity in time is to be covered in Chapter 4, another possible variable that could be made adaptive is

order. For adaptivity in order, a lower order would be used when rapid changes are occuring and small time steps are used, and where a higher order is used otherwise. Although possible, very little would be gained in terms speed as most of that would be covered by the time stepping.

The method could also be made adaptive in space; i.e. a refined grid spacing in areas where there is much activity and a looser grid elsewhere. The first challenge to applying this idea to the Lagrange-Galerkin method is the need for a spatial error estimate and a corresponding tolerance. The second challenge would be the need to adapt the Fourier derivatives, and implement an efficient Fourier transform for an irregularly spaced grid as in Section 3.3. Some examples of adaptive grid methods such as these can be seen in [11, 22, 42].

The Lagrange-Galerkin method could be modified so that it solves (2.16) rather than (1.3), as was done for the pseudospectral method in Chapter 2. In two dimensions solving for the vorticity ω rather than the velocity $\mathbf{u}$ would cut out about half of the computations. In reality very little would be saved since the $\mathbf{u}$ values would still need to be shifted along the trajectories, and this leads into a bottleneck for this method.

Chapter 4

Adaptive Lagrange-Galerkin method

With the current setup of the Lagrange-Galerkin method, if we were faced with a problem in which there is some period on $(0, T]$ which has rapid change or steep gradients, then we would need to have a small time step in order to attain a certain level of accuracy. Since the time is fixed then we would be using unnecessarily small time steps for the parts which are changing at a comparatively slower pace. The idea behind the adaptive Lagrange-Galerkin (A-LG) method is then to allow the time step to change so that larger time steps can be taken during these slow changing parts of the solution. The adaptivity could also work the other way. That is to say that the time step could become smaller in order to maintain the desired accuracy if the solution starts changing too fast.

4.1 Error estimate

The first issue that must be resolved before we can continue any further is how to measure the error. This error will be used to make a decision as to whether to increase or decrease the time step. Error measurements can be divided into two categories: *a priori* and *a posteriori.*

For the *a priori* error estimate we would need to know certain attributes of our problem. Determining an *a priori* error estimate can be tricky and possibly unrepresentative of our solution. We have little to gain from using this type of error

estimate. For a few examples of an *a priori* estimate see [30].

For an *a posteriori* error estimate we do not need to make any projections or determine any boundary values. We need only use what values are already known and perhaps carry an extra value along in the computations. Examples of *a posteriori* error extimates can be seen in [3] and in [19]. One such example can be summed up as follows. If $\|\cdot\|$ is the Euclidean norm and let ψ be the solution of the linearized problem $\frac{\partial \mathbf{u}}{\partial t} = A\mathbf{u}$ at time T, then the dual problem which runs backwards in time can be defined as $-\frac{\partial \phi}{\partial t} = A^{\top}\phi$, with the initial condition $\phi(T) = \psi$. An upper bound on the error can be defined in the following way

$$S_c(T) = \max_{\|\psi\|=1} S_c(T, \psi)$$

where

$$S_c(T, \psi) = \int_0^T \left\| \frac{\partial \phi}{\partial t} \right\| \mathrm{d}t$$

and it can be shown that

$$\|\mathcal{E}(T)\| \leq S_c(T) \max_{0 \leq t \leq T} \|k(t) R(\mathbf{u}(t))\|.$$

For this upper bound, $k(t) = k_n = t_n - t_{n-1}$ for $t \in (t_{n-1}, t_n]$, and the residual is defined as

$$R(\mathbf{u}(t)) = (\mathbf{u}(t_n) - \mathbf{u}(t_{n-1}))/k_n.$$

Then the step size k_n can be chosen so that

$$\|\mathbf{u}(t_n) - \mathbf{u}(t_{n-1})\| \approx \frac{\mathbb{T}_a}{S_c(T)}, \tag{4.1}$$

where $\mathbb{T}_a > 0$ is some predetermined tolerance value. Since this is an *a posteriori* error estimate, then the way it would work is to choose some new time value t_{n+1} to be used as the next time step. We then evaluate $\mathbf{u}(t_{n+1})$ and if it satisfies the condition (4.1) then it is kept, otherwise a new t_{n+1} is chosen and that particular time step is reevaluated [20].

Another *a posteriori* error estimate would be to run a similar algorithm of the same order, using the same data and then comparing the results. This is not truly an estimate but more of an indicator since it does not attempt to place an upper bound on the error, as an estimate would, but instead only gives an idea of the size of the error. At a quick glance this may seem like a lot of extra computational work and hence a much slower way of estimating the error. In reality this will do very little to slow the algorithm down in this case since the bottleneck of the method is still the unevenly spaced FT. We know from Section 3.3 that the unevenly spaced FT is $\mathcal{O}(2K^2N^2 \log N)$ whereas implementing a time stepping scheme such as the BDF is only $\mathcal{O}(N^2)$. This second algorithm can also be chosen so that it shares similar structure to the original algorithm and will only require a few extra computations.

4.1.1 A modified backward difference formula

We will use the modified Backward Difference Formula (mBDF) for the second algorithm in the adaptive method. The mBDF can be thought of as a higher order

trapezium rule and, for a general time stepping problem

$$\mathbf{u}_t = \mathcal{L}(\mathbf{u}),$$

is given by the following formula

$$\mathbf{u}^{n+1} - \beta\Delta t\left(\mathcal{L}(\mathbf{u}^{n+1}) + \mathcal{L}(\mathbf{u}^n)\right) = \sum_{j=n-s+1}^{n} \alpha_j \mathbf{u}^j \tag{4.2}$$

where β and α_j correspond to the values in Table 4.1. It has been shown in [60] that for $1 \leq s \leq 6$, the mBDF is both convergent and stable. The mBDF is also

s	β	$\{\alpha_{n+1}, \ldots, \alpha_{n-s+1}\}$
1	$\frac{1}{2}$	$\{1, -1\}$
2	$\frac{1}{2}$	$\{1, -1, 0\}$
3	$\frac{6}{13}$	$\{1, \frac{-15}{13}, \frac{3}{13}, \frac{-1}{13}\}$
4	$\frac{3}{7}$	$\{1, \frac{-19}{14}, \frac{9}{14}, \frac{-5}{14}, \frac{1}{14}\}$
5	$\frac{60}{149}$	$\{1, \frac{-235}{149}, \frac{180}{149}, \frac{-140}{149}, \frac{55}{149}, \frac{-9}{149}\}$
6	$\frac{60}{157}$	$\{1, \frac{-283}{157}, \frac{300}{157}, \frac{-300}{157}, \frac{175}{157}, \frac{-57}{157}, \frac{8}{157}\}$

Table 4.1: mBDF coefficients

stable and convergent for $s = 7$. Since this is not so for the BDF, there is no point in discussing the mBDF for such an order. Aside from the differing coefficients, the main difference between (4.2) and (2.19) is the additional term $\mathcal{L}(\mathbf{u}^n)$ on the left side of the equation (4.2) [60]. For this extra term we need only store one extra value from the previous time step.

The final piece that we need for the error indicator is the coefficients c_1 and c_2 from the respective BDF and mBDF methods. These coefficients indicate the size of the error term for the given method, and will provide a way of normalizing the effect of these errors for the two methods. These c values are called the local error constants and are given in Table 4.2. Now that we have all of the components we

method	c	1	2	3	4	5	6
BDF	c_1	$-\frac{1}{2}$	$-\frac{2}{9}$	$-\frac{3}{22}$	$-\frac{12}{125}$	$-\frac{10}{137}$	$-\frac{20}{343}$
mBDF	c_2	0	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{10}$	$-\frac{2}{15}$	$-\frac{8}{63}$

Table 4.2: Local error constants for the BDF and mBDF methods

can define our error indicator in the following way. Starting with $\mathbf{u}^n, \ldots, \mathbf{u}^{n-s+1}$, we first compute $\mathbf{u}^{n+1}$ using (2.19). Then we compute $\mathbf{u}_*^{n+1}$ using (4.2). The error indicator $\mathcal{E}$ is then given by

$$\mathcal{E} = \left| \frac{c_1}{c_1 - c_2} \right| \frac{\|\mathbf{u}^{n+1} - \mathbf{u}_*^{n+1}\|}{\|\mathbf{u}^{n+1}\|}.$$

A ratio of the norms has been used in this definition, to give a norm relative to $\mathbf{u}^{n+1}$. This relative norm is used since we are really only interested in how far $\mathbf{u}_*^{n+1}$ deviates from $\mathbf{u}^{n+1}$ as opposed to the true difference in the Euclidean norm [40, 45].

Once the error indicator has been decided then there are three choices based on the result. If the error is too small, i.e. $\mathcal{E} < \frac{\mathbb{T}_a}{10}$, then we can increase the step size. If the error is too large, i.e. $\mathcal{E} > \mathbb{T}_a$, then we ignore that particular value of $\mathbf{u}^{n+1}$ since it may be incorrect and then we decrease the step size. Otherwise if the error is within the tolerance that we have set, then we continue with the same step size.

This process is refered to as the Milne device. At this stage we have the option to choose exactly how much we will be increasing or decreasing the step size based on how far the error deviates from the tolerance boundaries (e.g. $\Delta t = \frac{\Delta t}{\log_{10} \varepsilon}$). For this type of adaptive time stepping it would be better to use a true error estimate such as (4.1), since it is an actual upper bound on the error. Since we are using an error indicator and we know that overall, the timesteps will be small, it will suffice to simply double or half the timestep as required.

4.2 Modifying the step size

Once it has been decided that a doubling or halving of the time step is needed, there are a few more things to be done, as the list that is being used only contains elements that are Δt apart.

4.2.1 Halving Δt

In order to halve the time step we need to have the elements that fit in between the values in the current list. Rather than $\{\mathbf{u}^n, \ldots, \mathbf{u}^{n-s+1}\}$, which are Δt appart, we want $\{\mathbf{u}^n, \mathbf{u}^{n-\frac{1}{2}}, \ldots, \mathbf{u}^{n-\frac{s-1}{2}}\}$ where the spacing is $\frac{\Delta t}{2}$. This can be accomplished using a similar form of interpolating polynomial to what was seen in Section 2.6, the Lagrange polynomial. In this case we are doing a true interpolation rather than an extrapolation.

The general form of the polynomial for an s order algorithm can be defined as

$$p(x) = a_0 l_0(x) + a_1 l_1(x) + \cdots + a_s l_s(x)$$

where

$$l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{s} \frac{x - x_i}{x_k - x_i}, \quad \text{for } k = 0, \ldots, s.$$

We can simplify this since we are dealing with evenly spaced elements and even further since we only need a few extra terms [14]. Knowing that $x_k = k\Delta t$ for $k = 0, \ldots, s$ then with $s' = \lfloor \frac{s}{2} \rfloor$,

$$l_k((j + \frac{1}{2})\Delta t) = \prod_{\substack{i=0 \\ i \neq k}}^{s} \frac{(j + \frac{1}{2}) - i}{k - i} \quad \text{for } k = 0, \ldots, s, \ j = 1, \ldots, s', \tag{4.3}$$

which is independent of Δt so we can pre-compute the interpolating matrix L. Then, given the list $\{\mathbf{u}^n, \ldots, \mathbf{u}^{n-s+1}\}$ with a spacing of Δt we can obtain a new list $\{\mathbf{u}^n, \mathbf{u}^{n-\frac{1}{2}}, \ldots, \mathbf{u}^{n-\frac{s-1}{2}}\}$ where the new values $\{\mathbf{u}^{n-\frac{1}{2}}, \mathbf{u}^{n-\frac{3}{2}}, \ldots, \mathbf{u}^{n-\frac{s-1}{2}}\}$ are computed using the matrix multiplication

$$[\mathbf{u}^{n-\frac{1}{2}} | \mathbf{u}^{n-\frac{3}{2}} | \cdots | \mathbf{u}^{n-\frac{s-1}{2}}] = [\mathbf{u}^n | \cdots | \mathbf{u}^{n-s+1}] L.$$

We can now use the normal BDF with a time step of $\frac{\Delta t}{2}$ on this new list to obtain $\mathbf{u}^{n+\frac{1}{2}}$.

The error term for an interpolating polynomial is dominated by the next term in the expansion. From (4.3) it can be seen that the polynomial will depend on Δt. So if an s order polynomial is used then the error will be $\mathcal{O}(\Delta t^{s+1})$ which is also the order of the error for the time stepping formula. So the newly interpolated values are sufficiently accurate provided all s points are used for the interpolation.

4.2.2 Doubling Δt

The process for doubling the time step is very similar to the initialization procedure seen in Section 2.4.1. In fact there is no need for an initialization procedure with this method as the step size will increase on its own when needed. As a consequence the initial time step for the method must be chosen to be sufficiently small in order that the results will still be meaningful.

Starting with the list $\{\mathbf{u}^n, \ldots, \mathbf{u}^{n-s+1}\}$, we use 'leapfrogging' to get $\{\mathbf{u}^{n+(s-1)}, \ldots, \mathbf{u}^{n-s+1}\}$ and then choose every second value and let this be the new list as in Figure 4.1. Since the time increment has already been increased by $(s-1)\Delta t$ there

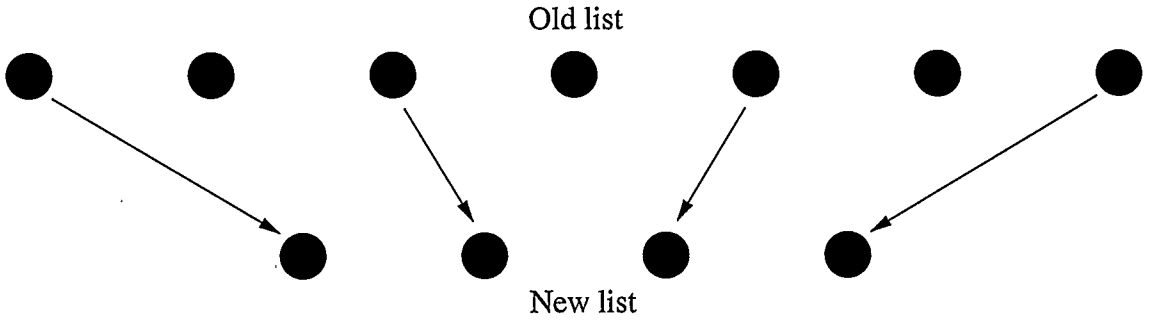

Figure 4.1: Old list to new list

is no need to immediately use the BDF with a time step of $2\Delta t$ to find a new value $\mathbf{u}^n$. It is better to simply return to the loop and start again using $2\Delta t$ as the time step. That way the procedure will be a little more sensitive to any changes in the solution.

4.2.3 Overshooting

Since the time steps are variable it could end up that the distance to the final time T is much smaller than Δt, as shown in Figure 4.2. Care must be taken to ensure

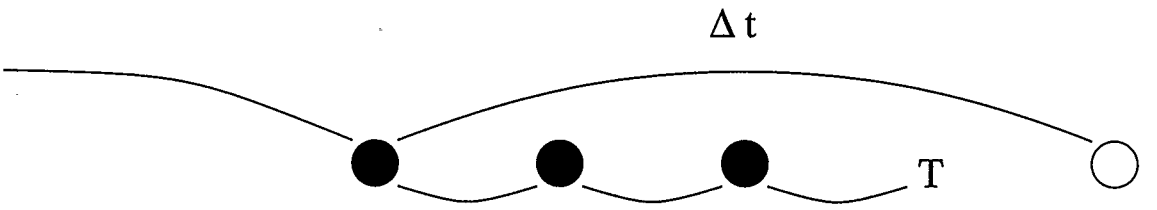

Figure 4.2: Overshooting T

that the final steps allow the method to finish exactly at T. This is accomplished by employing the algorithm for halving Δt from Section 4.2.1 whenever overshooting is going to occur.

4.3 Some test problems

Again, the method is implemented for a few problems to illustrate its behavior. When the rotating cone and the standing sine problems are solved with $\mathbb{T}_a = 10^{-2}$, results are obtained which are similar to those in Sections 3.5.1 and 3.5.2 and will thus be omitted. For the remainder of this thesis the errors are measured using the L^2 norm unless otherwise specified.

4.3.1 Vortex convergence problem

This problem was described previously in Section 3.5.3. The conditions used will be the following: $N = 128$, $T = 40$, $\mathbb{T} = 10^{-2}$ and $\nu = 5 \times 10^{-4}$. The time stepping will be decided adaptively based on $\mathbb{T}_a = 10^{-j}$, for $j = 1, 2, 3$. The $\mathcal{E}_j$ are the corresponding errors to the $\mathbb{T}_a = 10^{-j}$. Results can be seen in Figure 4.3 and Table 4.3. Figure 4.4 shows the size of Δt as time progresses. It is clear from this figure that around $t \approx 23$, Δt was doubled.

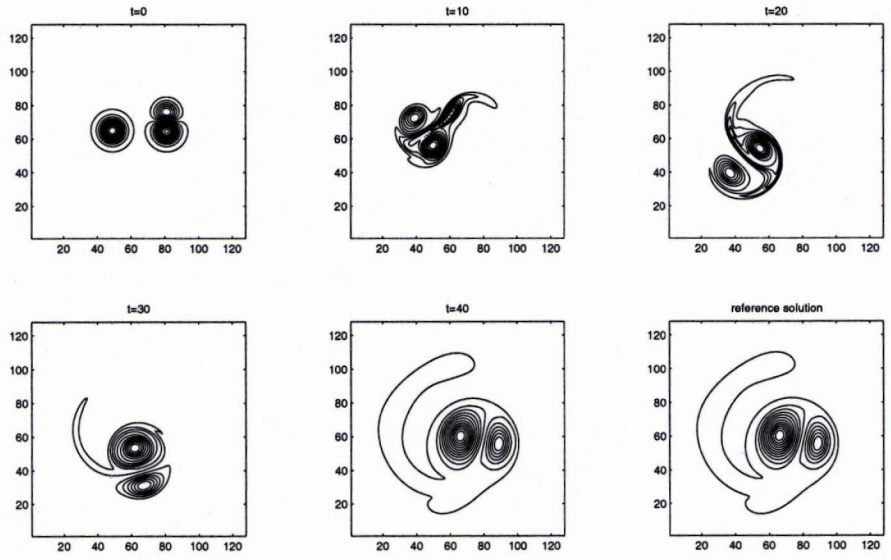

Figure 4.3: **A-LG solution for the vortex problem:** contours of $\omega(\mathbf{x}, t)$ of the solution to the vortex problem for the times indicated using $s = 3$ and $\mathbb{T}_a = 10^{-2}$ at selected times t.

4.3.2 Stream vortex problem

This is the same initial condition as that which was described in Section 3.5.4. The values used for this computation are, $T = 150, \nu = 5 \times 10^{-4}, \mathbb{T} = 10^{-2}$ and $N = 128$. The results can be seen in Figure 4.5 and Table 4.4, where the error $\mathcal{E}_j$ corresponds to the adaptive tolerance $\mathbb{T}_a = 10^{-j}$. Figure 4.6 shows not only that Δt was doubled at $t \approx 40$ but that Δt was quite large ($\Delta t > 1$) for most of the computation.

Order	2	3	4	5	6
$\mathcal{E}_1$	0.257	0.174	0.0464	0.167	0.191
$\mathcal{E}_2$	0.077	0.0366	0.00185	0.155	0.167
$\mathcal{E}_3$	0.111	0.000845	0.000155	0.0275	0.0583

Table 4.3: A-LG errors for the vortex problem

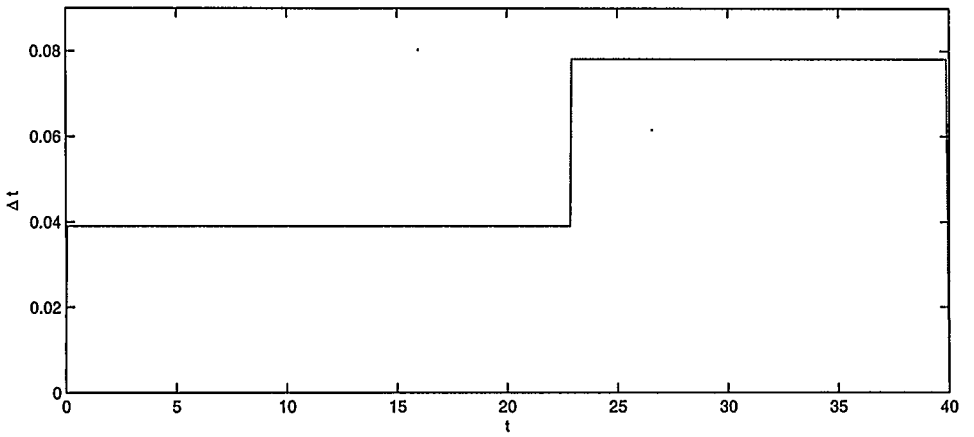

Figure 4.4: Δt vs. t for the vortex problem

Looking at Tables 4.3 and 4.4, it seems that $s = 3$ or 4 would be good choices. This could be due to the idea that the higher order methods, $s = 5, 6$, tend to get a bit cumbersome. In Section 4.2.2 we saw that in order to double the step size from Δt to $2\Delta t$, that $(s-1)$ steps at Δt are needed. There is also the fact the $\left|\frac{c_1}{c_1 - c_2}\right| = 24$ for $s = 4$. This high value will make the method even less likely to take larger time steps.

Order	2	3	4	5	6
$\mathcal{E}_1$	0.0487	0.0485	0.0195	0.08	0.0667
$\mathcal{E}_2$	0.0342	0.009	0.0032	0.0212	0.0290
$\mathcal{E}_3$	0.00556	0.000107	0.000301	0.000661	0.00431

Table 4.4: A-LG errors for the stream vortex problem

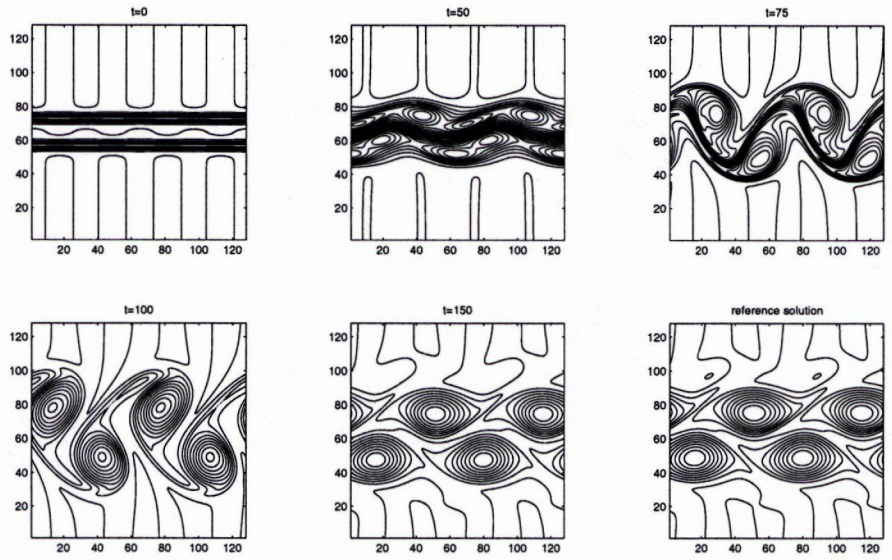

Figure 4.5: **A-LG solution to the stream vortex problem:** contours of $\omega(\mathbf{x}, t)$ of the solution to the stream vortex problem using the A-LG method at order 3 and with $\mathbb{T}_a = 10^{-2}$ for selected times t.

4.4 Variations on the adaptive Lagrange-Galerkin method

This is not the only way to make the method adaptive in time. Some alternatives could be to use Runge-Kutta time stepping or a modified multistep method that can deal with variable Δt.

4.4.1 Runge-Kutta time stepping

As an alternative to the adaptive scheme presented above a one-step scheme could be used. The most widely known of these schemes are the Runge-Kutta (R-K) schemes for which the general r-stage version for a time dependent equation such as (2.14)

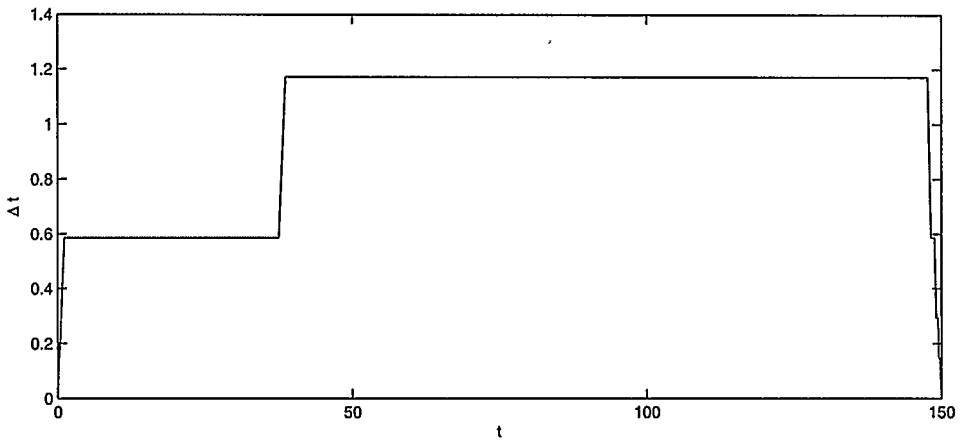

Figure 4.6: Δt vs. t for the stream vortex problem

can be defined as [53]

$$
\begin{aligned}
K_1 &= \mathcal{L}(u^n) \\
K_j &= \mathcal{L}\left(u^n + \Delta t \sum_{k=1}^{j-1} a_{jk} K_j\right), \quad j = 2, \ldots, r \\
u^{n+1} &= u^n + \Delta t \sum_{k=1}^{r} b_j K_j.
\end{aligned}
$$

If a R-K method is used then the halving and doubling of Δt becomes much simpler, as all that is really needed is to double or halve Δt between the steps of the method and the rest is already inherent in the algorithm. For this same reason it may be thought that a R-K scheme would not need a 'start up' procedure such as the leapfrogging method presented in Section 2.4.1. This is only partially true as it may also be necessary to use a different number of stages r in order to keep the same order of accuracy.

Another reason to choose R-K schemes is to save storage space. s-order R-

K methods have been developed so as to only use a fraction of the storage usually required. For example, a 4^{th} order scheme that only requires the storage of 2 previous values of K, whereas a BDF method would require all four previous values.

The main reason not to use R-K methods is that in general they are not useful for solving stiff problems or operators such as $\mathcal{L}$ that require some implicit computations. Unfortunately, the N-S equations tend to fit both of these criteria.

4.4.2 Variable multistep methods

A variable multistep method is a time stepping algorithm that does not require the time steps to be evenly spaced as the BDF and Adams schemes do. There may be several of advantages to computing the time stepping with a multistep method for irregularly spaced data, such as generalized BDF. First, there would be a savings in the complexity of the computation, and as a consequence, the execution time. Savings would occur near the end of the iterations where the current method must take care not to overshoot the final time, T. The variable multistep method would simply finish with a single time step rather than 'winding' down as the current method does. The other advantage would be in the increasing (or the decreasing) of the time step. There would be no need to iterate $s-1$ times as mentioned in Section 4.2.2. Δt could simply be doubled, then a single time step could be taken and then the next iteration could be computed.

Chapter 5

Results and comparisons

In this chapter we look to make a direct comparison between the adaptive and the non-adaptive Lagrange-Galerkin method. As was done in the previous sections, a pseudospectral method with a very small time step will be used to compute the reference solution to the given problems.

5.1 Speed vs. accuracy

The problems examined in Sections 3.5.3, 3.5.4, 4.3.1 and 4.3.2 will now be reexamined. This time around we will restrict our attention to only the 3^{rd} order methods, noting the time required for the computations. This will help to determine if the adaptive method can indeed provide an improvement over the unadaptive method. The way this will be done is to fix a desired accuracy, say 10^{-2}, and then to choose

	Vorticity			Stream		
Method	$\mathcal{E}$	steps	time (h:m:s)	$\mathcal{E}$	steps	time (h:m:s)
LG	0.00458	206	0:42:58	0.00323	100	0:20:59
A-LG	0.000845	825	1:21:15	0.009	189	0:27:42
LG	0.00284	810	1:12:57	0.00456	182	0:23:43

Table 5.1: **Computational time:** a comparison of the LG and A-LG methods for the vorticity and stream problems.

the error in the $s = 3$ column of Tables 3.6, 3.7, 4.3 and 4.4 which are less than the tolerance 10^{-2}. The results are given in Table 5.1. Note that the two methods are not exactly on even ground as can been seen by the number of timesteps that each method takes. For the vortex problem the error for the A-LG method is much smaller than the error for the first LG method. This accounts for the large difference in the time taken for the two methods. The fact that the A-LG method takes 4 times as many steps as the first LG method also backs up this idea. Table 4.3 shows that if $\mathbb{T}_a$ is loosened a degree then the resulting error is larger than the desired tolerance. For a fairer comparison, the LG method was used to evaluate the same problems but with the number of time steps chosen to more closely match the steps taken by the A-LG method, these results are shown in the last row of Table 5.1. While the LG method was a bit faster in both cases, the error for the vortex problem was much more than its A-LG counterpart, whereas for the stream problem the error for the LG method was better. These results seem to indicate that the problem at hand will determine whether the A-LG method will be superior to the LG method or not. To investigate this further we shall devise a new problem.

5.2 More comparative tests

Now we shall look at a couple of other problems and compare the results obtained for the two different methods. For these new problems the initial conditions were chosen to have random phase and energy in a similar fashion to the inital conditions described in [36]. With the initial condition defined in this way we introduce a new parameter $\mathfrak{r}$. $\mathfrak{r}$ determines the decay of the Fourier coefficients of the initial condition,

the larger the parameter $\mathfrak{r}$, the faster the decay of the coefficients, the smoother the initial conditions will be. For $\mathfrak{r} < \frac{1}{4}$ the chosen grid size is not large enough to resolve all of the details in the initial condition. In other words, with $N = 128$, the initial condition cannot be determined with sufficient accuracy for our purposes. The initial condition is also scaled so that the maximum absolute value of the vorticity is 1.

5.2.1 A random problem

The results are shown in Figures 5.1 and 5.2, and were obtained using the following conditions: $N = 128$, $T = 150$, $\mathbb{T}_a = 10^{-\frac{5}{2}}$, $s = 3$, $\nu = 5 \times 10^{-4}, \mathfrak{r} = \frac{1}{4}$ and $\mathbb{T} = 2$. Table 5.2 displays the errors, the number of steps and time taken for both methods.

. The results for the random problem seem to indicate that the two methods give comparable results, with the adaptive solution a bit slower than its non-adaptive counterpart. This can be attributed to the conservative nature of the Milne device. One option that could change this is to use a looser tolerance $\mathbb{T}_a$, and/or modify the value that determines whether $\mathcal{E}$ is too small. Currently if $\mathcal{E} < \frac{\mathbb{T}_a}{10}$, then the step size is doubled. As a result of this decision, the method refused to change Δt, during the computation, except at the beginning and the end, as seen in Figure 5.3. This means that the problem does not vary enough to necessitate variable values

Method	$\mathcal{E}$	steps	time (h:m:s)
LG	0.00767	280	0:36:08
A-LG	0.00894	282	0:40:26

Table 5.2: Results for the problem with a random initial condition

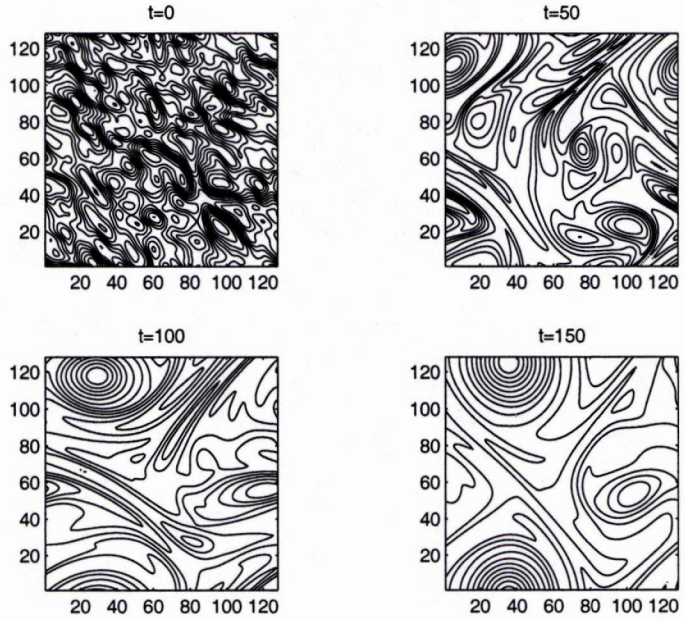

Figure 5.1: **LG solution for the random problem:** contours of $\omega(\mathbf{x}, t)$ of the LG solution using 280 time steps at selected times t with $\nu = 5 \times 10^{-4}$. The results are visually indistinguishable from those in Figure 5.2.

of Δt. What has occured, is that the method has computed the solution using the most optimal value of Δt it had available. To make the method more apt to change Δt, we could change the value $\frac{\mathbb{T}_a}{10}$ to $\frac{\mathbb{T}_a}{4}$, but in doing so there is a greater risk of incurring excessive error, due to the increased possibility of having the method take an erroneously large time step.

With the method modified such that, Δt will be doubled if $\mathcal{E} < \frac{\mathbb{T}_a}{4}$, we will look at a new problem, that will illustrate the adaptivity of the method. Again we take a random initial condition with the same values as stated above, only this time the viscosity is much higher, $\nu = 5 \times 10^{-3}$ and $\mathfrak{r} = \frac{1}{2}$. Increasing the viscosity will have

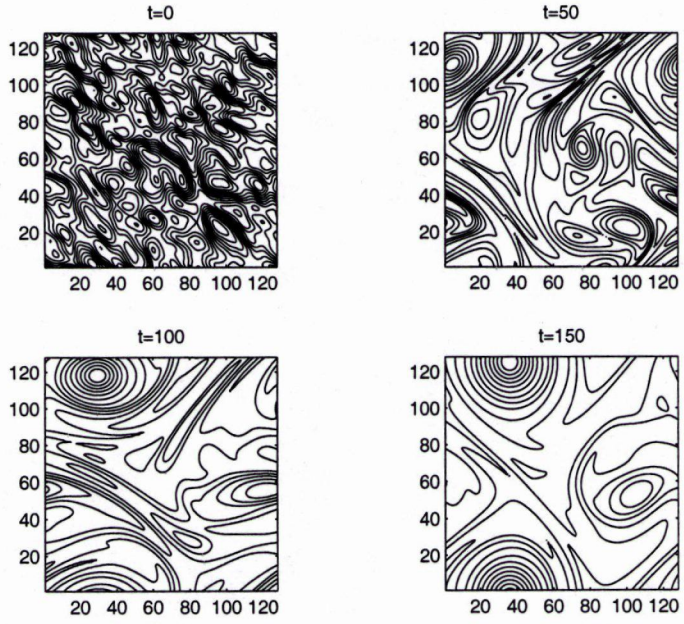

Figure 5.2: **A-LG solution for the random problem:** contours of $\omega(\mathbf{x}, t)$ of the A-LG solution using $\mathbb{T}_a = 10^{-\frac{5}{2}}$ at selected times t with $\nu = 5 \times 10^{-4}$. The results are visually indistinguishable from those in Figure 5.1.

similar results as extending the final time T. The results are shown in Figures 5.4, 5.5 and Table 5.3. Figure 5.6 shows the variation of Δt as the time progresses. Since this problem is more diffusive than the previous problem, it becomes much smoother as it approaches the final time T. This smoothness is the reason for the progressively larger time steps. While this additional diffusion may allow the A-LG method to take larger steps and thus finish the computation quickly, it also diffuses any errors incurred at the early stages by the LG method. This makes the results in Table 5.3 comparable again, with neither method a clear improvement over the other.

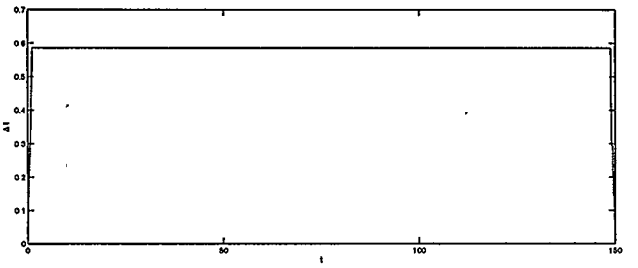

Figure 5.3: Δt **vs.** t **for the random problem:** Δt does not vary for this problem (except at the ends) due to the conservative values chosen for the A-LG method.

5.3 Further possible modifications

It is by no means necessary to stick to the standard BDF, A-B or A-M formula, provided the version being used still satifies the stability and convergence conditions required for the problem. A few possible modifications could be to use a variable time stepping formula vBDF,vA-B or vA-M [53]. This would save a lot of time at higher orders when going through the growing phase in Section 4.2.2, since we would only have to iterate once. For the same reason there would be very few problems with overshooting the final time T. All that would be required is to set Δt to be equal to the distance to T. One drawback to this idea is that often, the coefficients associated with the methods would have to be recomputed at each time step.

Method	$\mathcal{E}$	steps	time (h:m:s)
LG	0.00339	404	0:52:52
A-LG	0.00475	336	0:50:19

Table 5.3: Results for a more viscous problem

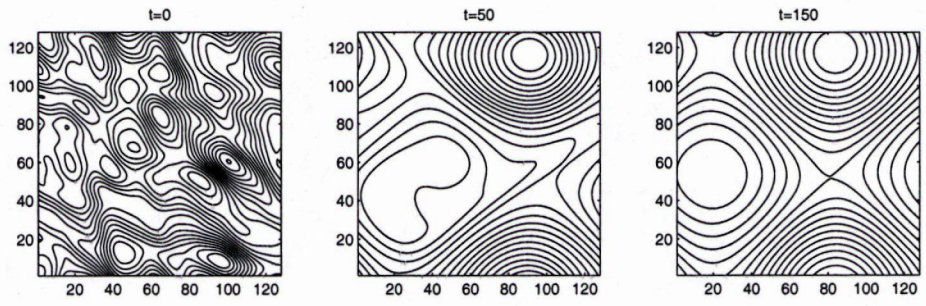

Figure 5.4: **LG solution for a more viscous problem:** contours of $\omega(\mathbf{x}, t)$ of the LG solution using 404 time steps at selected times t with $\nu = 5 \times 10^{-3}$. Due to the high diffusion, there is little that changes visually between $t = 50$ and $t = 150$.

Another possible variation is to chose a specific version to match the problem. One example can be seen in [28] where the A-BDF is especially suited to deal with highly oscillatory problems. This variation of the BDF can even go as high as $s = 7$. The drawback to the A-BDF is that since it is specially suited to this certain type of problem then it may not work so well in a more general setting.

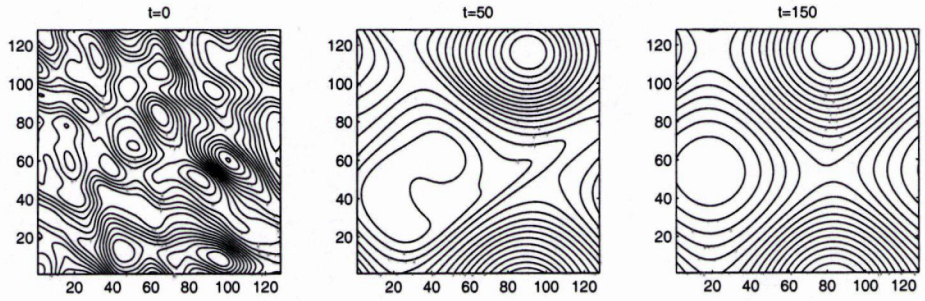

Figure 5.5: **A-LG solution for a more viscous problem:** contours of $\omega(\mathbf{x}, t)$ of the A-LG solution using $\mathbb{T}_a = 10^{-\frac{5}{2}}$ at selected times t with $\nu = 5 \times 10^{-3}$.

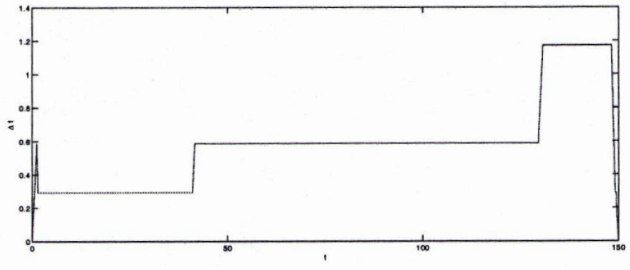

Figure 5.6: Δt **vs.** t **for a more viscous problem:** the smoothness of this problem allows the A-LG method to take larger steps as t progresses. The doubling of Δt is noticable at $t \approx 45$ and 130.

Chapter 6

Conclusions

When restricted to simple boundaries it is difficult to find a class of methods that can provide the combination of accuracy and simplicity that spectral methods provide. The pseudospectral method, which is possibly the most basic of the spectral methods was examined in Chapter 2. Provided sufficient time steps are used, so as to avoid any instabilities, the pseudospectral method provides quite accurate results. Often it will be such that the sufficient number of time steps required will be so large that the method will be too time consuming and end up with results which are more accurate than needed. While it is not necessarily wrong to have too much accuracy, the point here is that the method could take more time than needed to satisfy the minimum required accuracy.

This conditional instability for the pseudospectral method is the very strength of the Lagrange-Galerkin method. As was mentioned in Section 3.6 it has been shown that the method is unconditionally stable provided the integrations are computed exactly. This means that the size of the time step is determined solely by the accuracy not the stability. The Lagrange-Galerkin method could be constructed so as to be as fast as the pseudospectral method, but with the addition of stability.

Unfortunately with some of the choices made in this thesis this particular spectral Lagrange-Galerkin method is not as fast as this particular pesudospectral method. However it was never really the intention to compare the two methods directly. What we were looking to do is to provide an improvement to the existing spectral

Lagrange-Galerkin method.

The regular fixed time stepping algorithm was replaced with an adaptive algorithm, in order to achieve this improvement. This is a natural progression for the Lagrange-Galerkin method since, as was already mentioned, the method is unconditionally stable, up to certain constraints. The stability properties mean that there is no need to monitor the size of Δt in order to maintain some stability condition, as would be required for an adaptive pseudospectral method.

It is possible that adaptive time stepping can improve over fixed time stepping in specific cases. These cases include problems there the energy of the system decays rapidly, or needs to be computed over a long time period, but fine details still exist at the final time. This was not the case with the problems chosen in Section 5.2, as the diffusion in the problem had a tendency to smooth out the details at the final time. Although it did not show improved results, the adaptive method described in this thesis gave results that were comparable to its unadaptive counterpart. One slight advantage that the A-LG method has, is that given a desired tolerance, it is a bit easier to determine the tolerance $\mathbb{T}_a$ that will satisfy that tolerance than to determine the required number of time steps for the LG method.

Bibliography

[1] D.J. Acheson. *Elementary Fluid Dynamics.* Oxford applied mathematics and computing science series. Oxford University Press, New York, 1990.

[2] F. Auteri and N. Parolini. A mixed-basis spectral projection method. *Journal of Computational Physics*, 175:1–23, 2002.

[3] John W. Barrett and Peter Knabner. An improved error bound for a Lagrange-Galerkin method for contaminant transport with non-Lipschitzian adsorption kinetics. *SIAM Journal of Numerical Analysis*, 35(5):1862–1882, October 1998.

[4] Antony N. Beris and Costas D. Dimitropoulos. Pseudospectral simulation of turbulent viscoelastic channel flow. *Computer Methods in Applied Mechanics and Engineering*, 180:365–392, 1999.

[5] J.P. Boyd. *Chebyshev & Fourier Spectral Methods.* Number 49 in Lecture notes in Engineering. Springer-Verlag, Berlin, 1989.

[6] Christopher S. Bretherton and David E. Stevens. A forward-in-time advection scheme and adaptive multilevel flow solver for nearly incompressible atmospheric flow. *Journal of Computational Physics*, 129:284–295, December 1996.

[7] C. Sidney Burrus, Michael T. Heideman, and Don H. Johnson. Gauss and the history of the fast Fourier transform. *IEEE ASSP Magazine*, pages 14–21, October 1984.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral Methods in Fluid Dynamics.* Springer-Verlag New York Inc., 1988. Fifth Edition.

[9] C. Canuto and V. Van Kemenade. Bubble-stabilized spectral methods for the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 135:35–61, 1996.

[10] Ching J. Chen, Yousef Haik, and Vinay M. Pai. Bio-magnetic fluid dynamics. In Wei Shyy and Ranga Narayanan, editors, *Fluid dynamics at interfaces*, chapter 34, pages 439–452. Cambridge University Press, 1999.

[11] A. Cherhabili, N. K.-R. Kevlahan, and O. Vasilyev. An adaptive wavelet method for turbulence in complex geometries. In Michel Deville and Robert Owens, editors, 16^{th} *Imacs world congress 2000, proceedings, Lausanne-August 21-25,2000*, pages 411–439. IMACS, 2000.

[12] Alexandre J. Chorin and Jerrold E. Marsden. *A Mathematical Introduction to Fluid Mechanics.* Number 4 in Texts in Applied Mathematics. Springer-Verlag, New York, 3^{rd} edition, 1993.

[13] Peter Consantin. Some open problems and research directions in the mathematical study of fluid dynamics. In Björn Engquist and Wilfred Schmid, editors, *Mathematics unlimited - 2001 and beyond*, pages 353–360. Springer-Verlag, January 2001.

[14] S.D. Conte and Carl de Boor. *Elementary Numerical Analysis: An algorithmic approach.* McGraw-Hill, Inc., New York, 3^{rd} edition, 1965.

[15] John B. Conway. *A course in functional analysis.* Springer-Verlag, New York, 1990. Second Edition.

[16] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, April 1965.

[17] A. Dipankar and T. K. Sengupta. A comparative study of time advancement methods for solving Navier-Stokes equations. *Journal of Scientific Computing*, 21(2):255–250, October 2004.

[18] Tobin A. Driscoll, Satish C. Reddy, Lloyd N. Trefethen, and Anne E. Trefethen. Hydrodynamic stability without eigenvalues. *Science*, 261(5121):578–584, July 1993.

[19] K. Eriksson, C. Johnson, and A. Logg. Adaptive computational methods for parabolic problems. In Erwin Stein, René de Borst, and Thomas J.R. Hughes, editors, *Encyclopedia of computational mechanics*, volume 1, chapter 24. John Wiley & Sons, Ltd, November 2004.

[20] Kenneth Eriksson and Claes Johnson. Error estimates and automatic time step control for nonlinear parabolic problems, I. *SIAM Journal on Numerical Analysis*, 24(1):12–23, February 1987.

[21] Maurizio Falcone, Roberto Ferretti, and Tiziana Manfroni. Optimal discretization steps for a class of semi-Lagrangian schemes. In *Numerical methods for viscosity solutions.* Iraklion, 1999.

[22] M. Farge, N.K.-R. Kevlahan, and K. Schneider. Comparison of an adaptive wavelet method and nonlinearly filtered pseudospectral methods for two-dimensional turbulence. *Theoretical and Computational Fluid Dynamics*, 9:191–206, 1997.

[23] Ronald Fedkiw, Henrik Wann Jensen, and Duc Quang Nguyen. Physically based modeling and animation of fire. In *Proceedings of the* 29^{th} *annual conference on computer graphics and interactive techniques*, pages 721–728, New York, 2002. SIGGRAPH, ACM Press.

[24] Ronald Fedkiw, Henrik Wann Jensen, and Joe Stam. Visual simulation of smoke. In *Proceedings of the* 28^{th} *annual conference on computer graphics and interactive techniques*, pages 15–22, New York, 2001. SIGGRAPH, ACM Press.

[25] Charles L. Fefferman. Existence and smoothness of the Navier-Stokes equation, May 2000. http://www.claymath.org/millenium/Navier-Stokes_Equations.

[26] Jeffrey A. Fessler and Bradley P. Sutton. Nonuniform fast Fourier transforms using min-max interpolation. *IEEE Transactions on Signal Processing*, 51(2):560–574, February 2003.

[27] Bengt Fornberg and David M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, pages 203–267, 1994.

[28] Christoph Fredebeul. A-BDF: A generalization of the backward differentiation formulae. *SIAM Journal of Numerical Analysis*, 35(5):1917–1938, October 1998.

[29] K. Gerdes, D. Schötzau, C. Schwab, and T. Werder. hp discontinuous Galerkin

time stepping for parabolic problems. *Computer Methods in Applied Mathematics and Engineering*, 190:6685–6708, 2001.

[30] Vivette Girault, Béatrice Rivière, and Mary F. Wheeler. A discontinuous Galerkin method with nonoverlapping domain decomposition for the Stokes and Navier-Stokes problems. *Mathematics of Computation*, 74(249):53–84, 2004.

[31] David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications.* Number 26 in CBMS-NSF Regional Conference Series on Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1977.

[32] David Gottlieb and Eitan Tadmor. The CFL condition for spectral approximations to hyperbolic initial-boundary value problems. *Mathematics of Computation*, 56(194):565–588, April 1991.

[33] Leslie Greengard and June-Yub Lee. Accelerating the nonuniform fast Fourier transform. *SIAM Review*, 46(3):443–454, 2004.

[34] Michael Griebel and Frank Koster. Adaptive wavelet solvers for the unsteady Navier-Stokes equation. In Josef Malek, Jindrich Necas, and Mirko Rokyta, editors, *Advances in Mathematical Fluid Mechanics*, Lecture notes of the sixth international school "Mathematical Theory in Fluid Mechanics", Pasecky, Czech Republic, September 19-26 1999. Springer-Verlag, 2000.

[35] Wilhelm Heinrichs. Least-squares spectral collocation for the Navier-Stokes equations. *Journal of Scientific Computing*, 21(1):81–90, August 2004.

[36] William D. Henshaw, Heinz-Otto Kreiss, and Jacob Ystrom. Numerical experiments on the interaction between the large and small-scale motions of the Navier-Stokes equations. *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, 1(1):119–149, 2003.

[37] Richard S. Hirsh, Hwar C. Ku, and Thomas D. Taylor. A pseudospectral method for the solution of the three-dimensional incompressible Navier-Stokes equations. *Journal of Computational Physics*, 70:439–462, 1987.

[38] Paul Houston and Endre Süli. Adaptive Lagrange-Galerkin methods for unsteady convection-dominated diffusion problems. *Mathematics of Computation*, 70(233):77–106, 2001.

[39] Weizhang Huang and David M. Sloan. A new pseudospectral method with upwind features. *IMA Journal of Numerical Analysis*, 13:413–430, 1993.

[40] Arieh Iserles. *A first course in the numerical analysis of differential equations.* Cambridge University Press, Cambridge UK, 1996.

[41] Hans Johnston, Jian-Guo Liu, and Chen Wang. A fourth order scheme for incompressible Boussinesq equations. *Journal of Scientific Computing*, 2003.

[42] N. Kevlahan and O. Vasilyev. An adaptive wavelet method for fluid-structure interaction. In B.J. Geurts, R. Friedrich, and O. Métais, editors, *Direct and large-eddy simulation workshop 4: University of Twente*, pages 142–145, 2001.

[43] R. M. Kirby and Z. Yosibach. Solution of von-Kármán dynamic non-linear plate equations using a pseudo-spectral method. *Computer Methods in Applied*

Mechanics and Engineering, 193:575–599, 2004.

[44] Markus Kraft and Sebastian Mosbach. A new explicit numerical scheme for large scale combustion problems. Technical Report 12, Cambridge Center for Computational Chemical Engineering, July 2003.

[45] John Denholm Lambert. *Numerical methods for ordinary differential systems: the initial value problem.* John Wiley & Sons, Ltd, 1991.

[46] Carlo Marchioro and Mario Pulvirenti. *Mathematical theory of incompressible nonviscous fluids.* Springer-Verlag, New York, 1994.

[47] P.D. Minev. A stabilized incremental projection scheme for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 36:441–464, 2001.

[48] Rajat Mittal. A Fourier-Chebyshev spectral collocation method for simulating flow past spheres and spheroids. *International Journal for Numerical Methods in Fluids*, 30:921–937, 1999.

[49] K.W. Morton and D. F. Mayers. *Numerical solution of partial differential equations.* Cambridge University Press, Cambridge, 1994.

[50] Ramachandran D. Nair, Jeffrey S. Scroggs, and Frederick H. M. Semazzi. A forward-trajectory global semi-Lagrangian transport scheme. *Journal of Computational Physics*, 190:275–294, 2003.

[51] Steven A. Orszag. Numerical simulation of incompressible flows within simple boundaries: accuracy. *Journal of Fluid Mechanics*, 49(1):75–112, 1971.

[52] Steven A. Orszag. Numerical simulation of incompressible flows within simple boundaries: Galerkin (spectral) representations. *Studies in Applied Mathematics*, 50(4):293–327, 1971.

[53] Roger Peyret. *Spectral methods for incompressible viscous flow.* Springer-Verlag New York Inc., 2002.

[54] A. Priestley. Exact projections and the Lagrange-Galerkin method: a realistic alternative to quadrature. *Journal of Computational Physics*, 112:316–333, 1994.

[55] J. P. Pulicani and E. Serre. A three-dimensional pseudospectral method for rotating flows in a cylinder. *Computers & Fluids*, 30:491–519, 2001.

[56] Satish C. Reddy and Lloyd N. Trefethen. Pseudospectra of the convection-diffusion operator. *SIAM Journal on Applied Mathematics*, 54(6):1634–1649, December 1994.

[57] D. Schötzau and C. Schwab. Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method. *SIAM Journal of Numerical Analysis*, 38:837–875, 2000.

[58] Dominik Schötzau. *hp-DGFEM for Parabolic Evolution Problems.* PhD thesis, Swiss Federal Institute of Technology, Zürich, 1999.

[59] Mohammed Seaïd. Semi-Lagrangian integration schemes for viscous incompressible flows. *Computational Methods in Applied Mathematics*, 2(4):392–409, 2002.

[60] Andrew Stuart. Modified backward difference formula. Private communication with Tony Ware, June 1990.

[61] E. Süli. Stability and convergence of the Lagrange-Galerkin method with non-exact integration. In J. R. Whiteman, editor, *The Mathematics of Finite Elements and Applications VI (MAFELAP 1987)*, volume 6, pages 435–442. BICOM, Academic Press, 1988.

[62] Vidar Thomée. *Galerkin Finite Element Methods for Parabolic Problems.* Number 1054 in Lecture notes in mathematics. Springer-Verlag, Berlin, 1984.

[63] Kim-Chuan Toh and Lloyd N. Trefethen. Calculation of pseudospectra by the Arnoldi iteration. *SIAM Journal on Scientific Computing*, 17(1):1–15, 1996.

[64] Lloyd N. Trefethen. Pseudospectra of linear operators. *SIAM Review*, 39(3):383–406, September 1997.

[65] Lloyd N. Trefethen. *Spectral Methods in MATLAB.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[66] Antony F. Ware. *A spectral Lagrange-Galerkin method for convection-dominated diffusion problems.* PhD thesis, Oxford University, 1991.

[67] Antony F. Ware. A spectral Lagrange-Galerkin method for convection-dominated diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 116:227–234, 1994.

[68] Antony F. Ware. Fast approximate Fourier transforms for irregularly spaced data. *SIAM Review*, 40(4):838–856, December 1998.