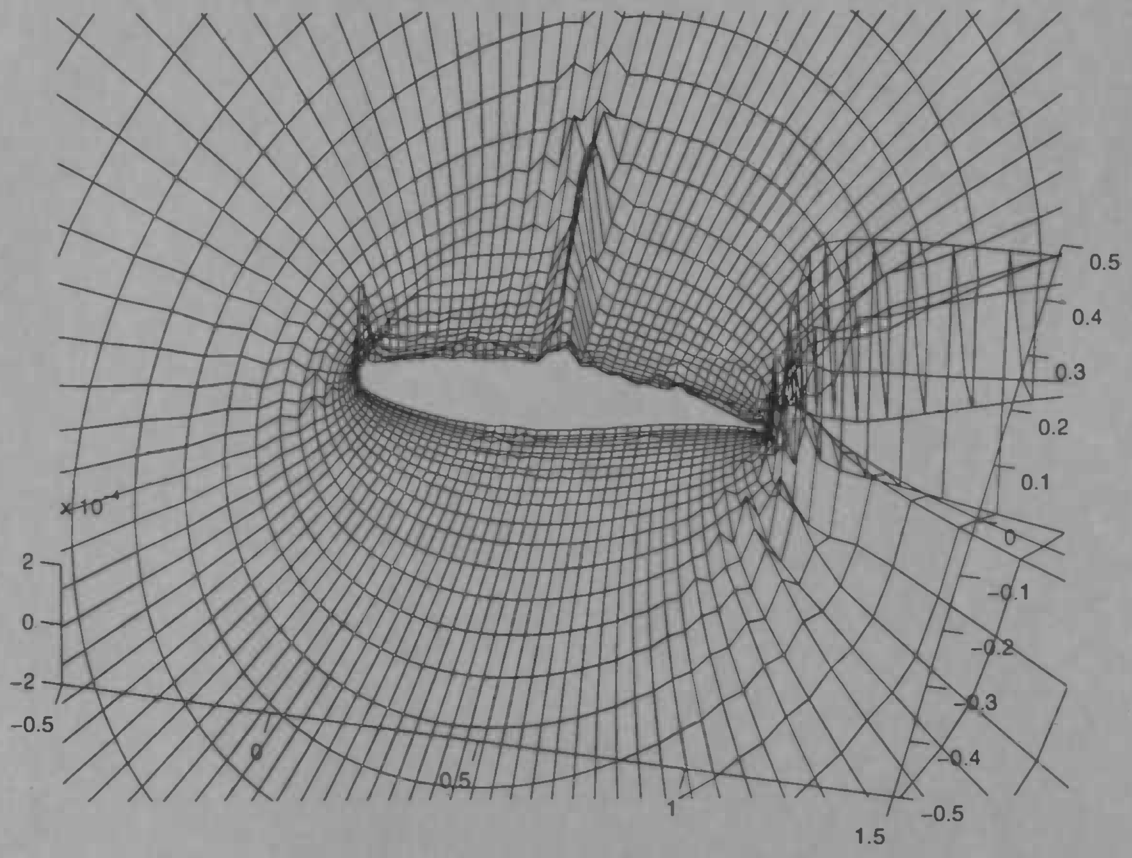
Optimizing the Addition of Artificial Dissipation using an Inverse Eigenvalue Method

Anne Baas

added artificial dissipation after using 'mixed' method, 5000 iterations



Department of Mathematics

RuG

Master's thesis

Optimizing the Addition of Artificial Dissipation using an Inverse Eigenvalue Method

Anne Baas

University of Groningen
Department of Mathematics
P.O. Box 800
9700 AV Groningen

November 1997

Contents

Preface

The thesis you are about to read is the result of more than one year of work, hope, despair and, finally, a lot of relief after the job was done. Writing about a subject of which you know just a little is quite a difficult thing to do, and in accomplishing this task, many people were of great help.
First of all, I had a lot of support from my supervisor in Groningen, professor Veldman. For someone as occupied with so many things as you, it was remarkable to see that reports of 60+ pages could be read and reviewed in one day. Your patience and support made it really a pleasure to work with you.
A lot of people are responsible for the fact that I enjoyed my stay in Indonesia as much as I have done. My supervisors Dr. Edy Alimin and Dr. Hadi Winarto helped me a lot with getting used to working at IPTN, and were a great stimulation for me. A very great help and hospitality were the people I met at ITB. From the staff members, I wish to thank especially 'Pak' Edy Soewono and 'Pak' Sembiring, without whose attention and care my stay would definitely have been a lot more difficult. Terima kasih banyak!
The enthousiasm, attention and friendship I got from many of the members of the ITB student society Himatika brings back good memories. I am very grateful for the fact that you all made me feel partly Indonesian as well, by sharing a part of your lives as students in Indonesia with me. To Emil, Dini, Uke, Sena, Andi, Indro, Pras and so many others, let me just say: Saya sangat menikmati saat-saat waktu di Bandung dan saya harap saya dapat segera kembali ke sana.
Finally, a lot of thanks go out to all the people around me who had the patience, the understanding and the courage to give me the feeling that my work would not be in vain. Sjoerd, Ineke and Thijs (it's all in the family) as the stabilizing factor one needs once in a while. Rutger and Frederiek L.E.C. and all the others in Groningen without whom I would have lost contact with the real world without a doubt. And most of all, a lot of love to Merel, whose everlasting attention, enthousiasm, criticism (always supportive!) and lack of 'that can be done later'-ism was the best support one can have in writing a thesis.
The job is done, the result of which you are about to read now. I hope you will enjoy it and, moreover, that your knowledge on the subject is increased somewhat.

Anne Baas, november 1997.

Chapter 1

Introduction

In Computational Fluid Dynamics, the problem of solving the Navier-Stokes equations numerically is a widely studied subject of research. These equations describe the motion of continuous media like fluids, gasses etcetera. As one can imagine, they are of great importance in many aspects of everyday life, for example in building waterways (describing the flow in a river or, on a larger scale, in oceans), and in aircraft industries. The focus of this report will be on the last category, as my research took place at the Indonesian aircraft industry Industri Pesawat Terbang Nusantara (IPTN) in Bandung, Indonesia.
The Navier-Stokes equations consist of the conservation laws of mass, momentum and energy, and the thermo-dynamical state equations. Section 1.2 deals with the formulation of these equations.
The Navier-Stokes equations need to be solved numerically, since they cannot be solved analytically. Therefore, a solver has to be used which, in my research, was an existing Navier-Stokes solver, a brief description of which can be found in section 1.3. In chapter 3, a more thorough insight in this solver is given.
In this solver, changes were made to optimize the addition of artificial dissipation. To perform this optimization, I made use of the relationship between the added dissipation and the eigenvalues of the local matrix, i.e. the matrix obtained by considering separate gridpoints. The ultimate goal was to see whether it is possible to 'tune' the eigenvalues of the local matrix (and thus the added dissipation) to achieve a numerical solution which is as accurate as possible. Let me first give a little more understanding of this problem.

1.1 Formulation of the Problem

As mentioned above, I used an existing Navier-Stokes solver. This solver was provided by the Research Department of the Indonesian aircraft company IPTN and it was developed by Dantje K. Natakusumah. To test this solver, a RAE2822 wing profile was used. The grid was made by a grid generator, also provided by IPTN.
The object of this project was to implement an Inverse Eigenvalue Method (IEM) into the Navier-Stokes solver. In general, this IEM uses a given matrix and given eigenvalues, and subsequently computes coefficients to this matrix such that the resulting matrix has the given (demanded) eigenvalues (see section 1.4). In this particular case, the IEM uses a matrix corresponding with a separate gridpoint. In this way, a 4×4 matrix is

obtained, consisting of the continuity equation, the two momentum equations and the energy equation. The relevant theory can be found in [1]. In the Navier-Stokes solver, artificial dissipation is added to maintain stability in 'sensitive' regions, such as shocks. However, adding too much artificial dissipation will cause inaccuracies in the numerical solution with respect to the exact solution. Therefore, it is necessary to make an estimate of the amount of added artificial dissipation and to limit this amount of dissipation. To obtain a means of estimation, we have to take into account the fact that artificial dissipation affects the eigenvalues of the local matrix. To tune the added dissipation, we can now restrict the eigenvalues in their (negative) magnitude, and thus impose restrictions on the amount of artificial dissipation. In chapter 2, a deeper insight into the world of artificial dissipation and Inverse Eigenvalue Methods is given.

1.2 The Navier-Stokes Equations

For convenience, let me first give the equations my research was based upon, the two-dimensional Navier-Stokes equations. In these equations, $\underline{u}$ is the velocity vector $(u,v)^T$, ρ is the density, $\underline{F}$ is a mass force (like gravity, elecro-magnetic force etc.), p is the pressure, e is the internal energy per mass unit $(= \frac{1}{1-\gamma}\frac{p}{\rho} + \frac{1}{2}(u^2+v^2))$ and μ is the dynamical viscosity.
Furthermore, ∇ denotes the gradient vector: $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})^T$, so that $\nabla \cdot \underline{u}$ means the divergence of $\underline{u}$.
First, we have the continuity equation, also known as the law of conservation of mass:

$$\frac{\partial \rho}{\partial t} + (\nabla \cdot \rho \underline{u}) = 0 \tag{1.1}$$

We see that for incompressible media (i.e. $\rho \equiv$ constant), this equation becomes

$$\nabla \cdot \underline{u} = 0 \tag{1.2}$$

The second and third equations are the momentum equations, describing the law of conservation of momentum:

$$\frac{\partial}{\partial t}\underline{u} + (\underline{u}\cdot\nabla)\underline{u} = \underline{F} \underbrace{- \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\Delta\underline{u} + \frac{1}{3}\mu\nabla(\nabla\cdot\underline{u})}_{\sigma} \tag{1.3}$$

Note that, for non-viscous media (i.e. for $\mu \equiv 0$), the above equations become the Euler equations.
Usually, the last three terms are denoted with σ, the stress tensor. It describes the force working on the surface of separation between the elements in a fluid, for example. In a non-viscous medium, only the stress in the normal direction is involved, and this is denoted by the pressure.
The last equation is the energy equation:

$$\frac{\partial}{\partial t}e + (\underline{u}\cdot\nabla)e = -\frac{p}{\rho}(\nabla\cdot\underline{u}) + \mathcal{D} + \frac{1}{\rho}\nabla\cdot(\lambda\nabla T), \tag{1.4}$$

where $\mathcal{D}$ is the viscous dissipation function, consisting of linear and nonlinear first order derivatives of the velocity components.
These equations may appear in many different formulations, sometimes making it difficult to notice that actually the same kind of equations are being dealt with.

1.3 Discretizing the Navier-Stokes equations

Having derived the analytical form of the Navier-Stokes equations, we have to discretize them in order to make numerical computations. For this discretization, we use the finite volume scheme as explained in the next chapter, with an explicit Runge-Kutta time stepping method.
The numerical scheme presented uses a central difference scheme for the spatial derivatives. Therefore it is necessary to add artificial dissipation to maintain stability in sensitive regions. In section 1.3.1, I will give a more detailed description of how and why this artificial dissipation has to be added.
In contrast to the finite difference method, the finite volume method is based upon the intgral form of the Navier-Stokes equations, written in a simple form as

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \tag{1.5}$$

To discretize this integral form, a grid is needed. In this particular case, a regular grid was used, as shown in figure 1.1. On this grid, the finite volume method is applied as

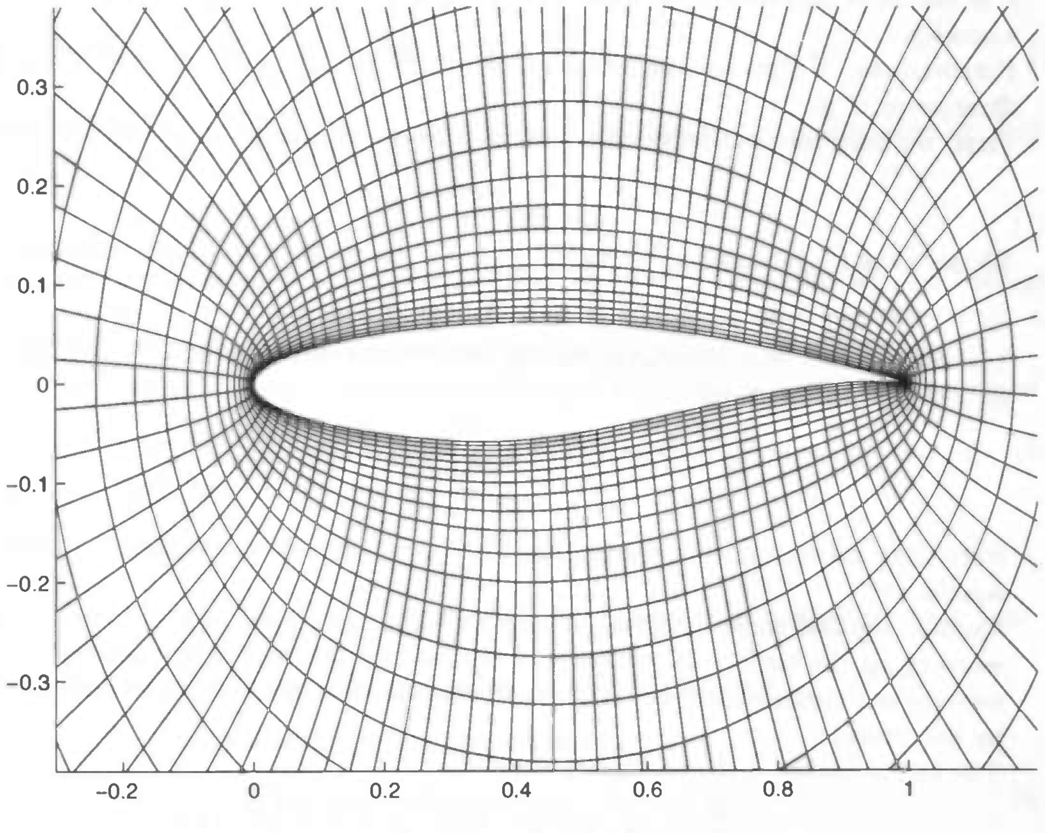


Figure 1.1: Regular grid around RAE2822 airfoil

follows to discretize the integral form:
Consider a quadrilateral cell with the cell center located at (i, j), as shown in figure 1.2

For this control volume, equation (1.5) can be rewritten as

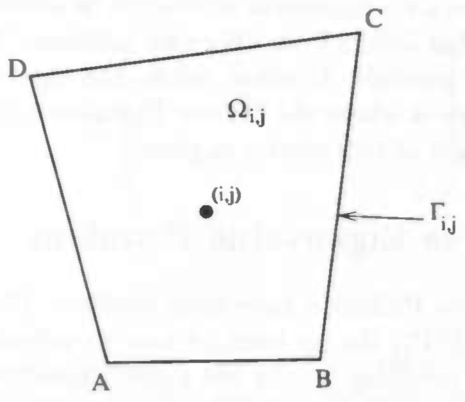


Figure 1.2: Example of a quadrilateral cell

$$\frac{\partial}{\partial t}\iint\limits_{\Omega_{i,j}} \mathbf{q}\,d\Omega + \oint\limits_{\Gamma_{i,j}} (\mathbf{F}dy - \mathbf{G}dx) = 0 \tag{1.6}$$

Rewriting the terms on the lefthand side and the righthand side, we get to the next approximation of (1.6):

$$S_{i,j}\frac{\partial}{\partial t}\mathbf{q}_{i,j} + \sum_{k=AB}^{DA} (\mathbf{F}_k\Delta y_k + \mathbf{G}_k\Delta x_k) = 0 \tag{1.7}$$

with $S_{i,j}$ the area of cell $\Omega_{i,j}$.
Having derived this, we proceed by evaluating the viscous and the convective fluxes separately, which gives the following, symbolically written, equation:

$$S_k\frac{\partial}{\partial t}(\mathbf{q}_k) + C(\mathbf{q}_k) + V(\mathbf{q}_k) = 0 \tag{1.8}$$

with $C(\mathbf{q}_k)$ and $V(\mathbf{q}_k)$ the convective and viscous operators, respectively. Now, the problem arises that, for various reasons, instabilities may occur. To avoid these instabilities, we have to add artificial dissipation.

1.3.1 Adding Artificial Dissipation

To maintain stability of the numerical scheme, artificial dissipation must be added. This artificial dissipation damps out oscillations in the numerical solution due to shocks and due to the discretization, and in regions with low viscosity, where oscillations can not be damped out automatically.
In this research, a local matrix is obtained for each grid point separately. This is done by taking advantage of the explicit formulation of the finite volume scheme. How this is done will be explained in chapter 2.
Adding artificial dissipation to a numerical scheme causes the eigenvalues of the local

matrix to become more negative in their real part. Indeed, we know from theory that this is a necessary condition for a numerical scheme to be stable. But adding dissipation gives a numerical solution that differs from the exact solution. Therefore, it is necessary to add as little dissipation as possible. In other words, the eigenvalues need to be negative, but not too much. And this is where the Inverse Eigenvalue Method can be used. In chapter 3, a further investigation of this matter is given.

1.4 The Inverse Eigenvalue Problem

For many years, Inverse Problems have been studied. This thesis deals with the Inverse Eigenvalue Problem (IEP), the problem of how to adjust certain coefficients of a given matrix such that the resulting matrix has given eigenvalues. Or, to put it more mathematically, a general Inverse Eigenvalue Problem may be defined in the following way:
Let $A(c_1, c_2, \ldots, c_n)$ be an $n \times n$ matrix having elements which are functions of the n parameters $c_1, \ldots, c_n$. The problem is now to determine $(c_i, i = 1, 2, \ldots, n)$ such that $A(\underline{c})$ has n given eigenvalues $\lambda_1^\star, \lambda_2^\star, \ldots, \lambda_n^\star$.
A more specific formulation of an IEP is as follows (see [6]):
Let $A(\underline{c})$ be the family

$$A(\underline{c}) = A_0 + \sum_{k=1}^{n} c_k A_k, \tag{1.9}$$

where $\underline{c} \in \mathbb{C}^n$ and $\{A_k\}$ are $n \times n$ matrices. If we denote the eigenvalues of $A(\underline{c})$ by $\{\lambda_i(\underline{c})\}_1^n$, we want to find $\underline{c} \in \mathbb{C}^n$ such that

$$\lambda_i(\underline{c}) = \lambda_i^\star, \quad i = 1, \ldots, n. \tag{1.10}$$

In [6], a clear summary of different types of IEP's is given. Generally, we can distinguish two types of IEP's, the *additive* and the *multiplicative* IEP. A brief overview of the class of additive IEP's is given here, as this was the topic of research in this project.
An additive inverse Eigenvalue Problem is obtained when the A_k's in (1.9) are defined by

$$A_k = \underline{e}_k \underline{e}_k^T, \quad k = 1, \ldots, n \tag{1.11}$$

with $\underline{e}_k$ the unit vector. In other words, we can write

$$A(\underline{c}) = A_0 + D \tag{1.12}$$

where $D = \mathrm{diag}(c_k)$. In [5] it is proven that this problem is always solvable over the complex field.
Let me first give an example of how an additive IEP can be applied.

Example 1.1 Consider the boundary value problem

$$-u''(x) + p(x)u(x) = \lambda u(x), \quad u(0) = u(\pi) = 0 \tag{1.13}$$

Suppose $p(x)$ is unknown, but the spectrum $\{\lambda_i^\star\}_1^\infty$ is given. The problem is now to determine $p(x)$. If we consider the discretization of this problem, using finite differences, we obtain

$$\frac{-u_{k-1} + 2u_k - u_{k+1}}{h^2} + p_k u_k = \lambda_j^\star u_k, \quad k = 1, \ldots, n, \quad u_0 = u_{n+1} = 0, \tag{1.14}$$

where λ_j^* is an eigenvalue in the set $\{\lambda_i^*\}_1^n$. This gives us an additive IEP with

$$A_0 = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & & \ddots & \\ & & & -1 & 2 \end{pmatrix} \tag{1.15}$$

and $D = \mathrm{diag}(p_k)$.

$\square$

As this simple example shows, (additive) Inverse Eigenvalue Problems can occcur in different types of problems. The way an IEP arises in this case, will be explained below, and in more detail in section 2.3.1

1.4.1 The Inverse Eigenvalue Method and Artificial Dissipation

Using the explicit formulation of the integral form, we can consider a 4×4 matrix at each individual grid point (see [10]). This matrix is achieved by considering a semi-discretization of the integral form of the Navier-Stokes equations. A system of ordinary differential equations is now obtained by applying this integral form to each cell separately. These differential equations have the following form:

$$\frac{d}{dt}(S_{i,j}\; q_{i,j}) + Q_{i,j} = 0 \tag{1.16}$$

For each of these matrices, the 'necessary' artificial dissipation must be computed. In section 2.3, a general and a more specific formulation of the additive IEP used in this research will be given. Many different ways of solving this problem have been described in past papers. I will consider in detail a paper by Wilkinson which can be found in [17]. In his article, an algorithm is presented based on Newton's method for solving a nonlinear system of n algebraic equations. In chapter 2 a mathematical analysis will be given of this algorithm.

Chapter 2

Mathematical Model

2.1 The Navier-Stokes Equations

The Navier-Stokes equations describe the motion of a flow. They consist of the continuity equation (2.1), the momentum equations (2.2) and the energy equation (2.3):

$$\frac{\partial \rho}{\partial t} + (\nabla \cdot \rho \underline{u}) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}\underline{u} + (\underline{u} \cdot \nabla)\underline{u} = \underline{F} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\Delta \underline{u} + \frac{1}{3}\mu \nabla(\nabla \cdot \underline{u}) \tag{2.2}$$

$$\frac{\partial}{\partial t}E + (\underline{u} \cdot \nabla)E = -\frac{p}{\rho}(\nabla \cdot \underline{u}) + \mathcal{D} + \frac{1}{\rho}\nabla \cdot (\lambda \nabla T), \tag{2.3}$$

where $\mathcal{D}$ is the viscous dissipation function, consisting of linear and nonlinear first order derivatives of the velocity components, and λ and μ are viscosity constants.
Many different ways have been used and are still being used to solve these equations. The Finite Volume Method is one of them, and because this method is widely accepted as one of the most powerful, and because the Navier-Stokes solver that was used in this research is based on the Finite Volume Method, I will give a brief overview of it.

2.1.1 The Finite Volume Method

Applying the Finite Volume Method to discretize Partial Differential Equations has two important advantages:

- It preserves the property of conservation (e.g. of mass, momentum and energy)
- Complicated geometries (like airfoils) can be dealt with rather easily

Let me illustrate the Finite Volume Method with an example in which I first give the general outlines of how a conservation law is achieved. A first-order equation is used to clarify these outlines. This equation is defined in a two-dimensional domain Ω, enclosed by a boundary Γ (see figure 2.1).

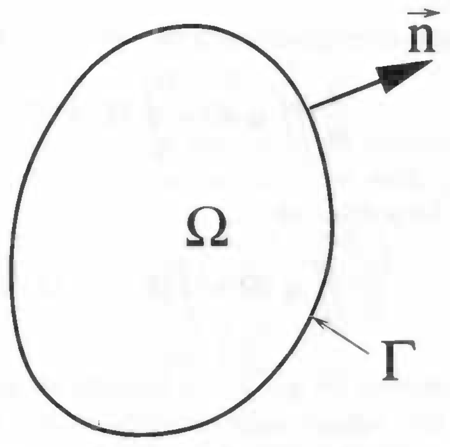


Figure 2.1: The domain Ω with boundary Γ

Example 2.1 A conservation law is the formulation of a physical phenomenon and it can be stated as follows:
For every subvolume $\Omega_1 \subset \Omega$ we have

$$\underbrace{\frac{\partial}{\partial t}\int_{\Omega_1} u \, d\Omega_1}_{\textit{increase per subvolume}} = \underbrace{\int_{\Gamma_1} \underline{\mathbf{H}}(u)\cdot \mathbf{n} \, d\Gamma_1}_{\textit{Flux through the edge}} + \underbrace{\int_{\Omega_1} f \, d\Omega_1}_{\textit{Source term}} \tag{2.4}$$

where Γ_1 is the surface of Ω_1, $\underline{\mathbf{H}} = (\mathbf{F}, \mathbf{G})$ is the flux function and $\mathbf{n}$ is the outward normal of Γ.
If $\underline{\mathbf{H}}$ is differentiable, Gauss' divergence theorem can be applied. Then we have

$$\int_{\Omega_1} \frac{\partial u}{\partial t} \, d\Omega_1 + \int_{\Omega_1} \text{div } \underline{\mathbf{H}}(u) \, d\Omega_1 = \int_{\Omega_1} f \, d\Omega_1 \tag{2.5}$$

Because this equality holds for every subvolume $\Omega_1 \subset \Omega$, a Partial Differential Equation of the following form results:

$$\frac{\partial u}{\partial t} + \text{div } \underline{\mathbf{H}}(u) = f \tag{2.6}$$

And of course a given Differential Equation can be reformulated in the integral form (2.4) as well. Consider the following first-order Partial Differential Equation:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \tag{2.7}$$

Integrating (2.7) within Ω gives

$$\frac{\partial}{\partial t}\iint_{\Omega} \mathbf{q} \, d\Omega + \iint_{\Omega} \nabla \cdot \underline{\mathbf{H}} \, d\Omega = 0 \tag{2.8}$$

After applying the Gauss-divergence theorem, we get to the integral form of (2.7):

$$\frac{\partial}{\partial t}\iint\limits_{\Omega} \mathbf{q}\; d\Omega + \oint\limits_{\Gamma} \underline{\mathbf{H}} \cdot \mathbf{n}\; d\Gamma = 0 \tag{2.9}$$

The last equation can be written as

$$\frac{\partial}{\partial t}\iint\limits_{\Omega} \mathbf{q}\; d\Omega + \oint\limits_{\Gamma} (\mathbf{F}\; dy - \mathbf{G}\; dx) = 0 \tag{2.10}$$

Instead of a volume integral, we get a line integral along Γ, which implies that the time variation of $\mathbf{q}$ inside the volume only depends on the variations of the flux along the surface. This is an essential property of the finite volume method.
If the volume Ω is divided into n subvolumes, the same conservation law can be applied to each subvolume separately. Adding up these 'sub-equations' should give (2.10) again. This property must be satisfied by the numerical discretization as well in order for the scheme to be conservative (i.e. only if the original differential equation is conservative!).

□

The advantages of using the Finite Volume Method are that problems in which u is discontinuous can be dealt with easier, because the conservation law demands less on smoothness of the solution than the partial differential equation. Furthermore, discrete jump relations become a consistent approach of analytical ones.
Jameson, Schmidt and Turkel ([9]) have proposed a scheme in which a semi-discretization is applied. In this semi-discretization one only approximates the spatial derivatives. This way, we get a formulation in which we can consider a 4×4 matrix for each cell separately. This will be explained in the next section.
We now focus on the Navier-Stokes equations, formulated in the integral form according to (2.4). From there, we can formulate them in the following way:

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \tag{2.11}$$

where

$$\mathbf{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p - \sigma_{xx} \\ \rho u v - \sigma_{xy} \\ \rho u H - u\sigma_{xx} - v\sigma_{xy} + q_x \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} \rho v \\ \rho v u - \sigma_{xy} \\ \rho v^2 + p - \sigma_{yy} \\ \rho v H - u\sigma_{xy} - v\sigma_{yy} + q_y \end{pmatrix} \tag{2.12}$$

In these equations, E is the internal energy per unit of mass and H is the total enthalpy per unit of mass. They are defined as

$$E = e + \frac{1}{2}(u^2 + v^2) \tag{2.13}$$

and

$$H = E + \frac{p}{\rho} \tag{2.14}$$

Furthermore, we have the elements of the viscous stress tensors, denoted with σ_{xx}, σ_{xy} and σ_{yy}, and the elements of the heat flux vector, denoted with q_x and q_y. They are defined as follows:

$$\begin{aligned}
\sigma_{xx} &= \mu_{tot}\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) \\
\sigma_{xy} &= \mu_{tot}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\
\sigma_{yy} &= \mu_{tot}\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right) \qquad (2.15) \\
q_x &= -k_{tot}\frac{\partial T}{\partial x} \\
q_y &= -k_{tot}\frac{\partial T}{\partial y}
\end{aligned}$$

where

$$\mu_{tot} = \mu + \mu_t$$

$$k_{tot} = k + k_t = \frac{\gamma}{\mathrm{Pr}}\mu + \frac{\gamma}{\mathrm{Pr}_t}\mu_t \tag{2.16}$$

where Pr is the constant Prandtl number ($\mathrm{Pr} = 0.72$), Pr_t is the constant turbulent Prandtl number ($\mathrm{Pr}_t = 0.90$). Note that the indexes do *not* mean derivatives in x or y direction! If the fluid is a perfect gas, then p and ρ are related by the state equation

$$p = \rho RT \tag{2.17}$$

with R the gas constant. Furthermore, if the specific heat at constant volume and pressure, c_v and c_p respectively, are constant, we have the following relationships:

$$e = c_v T, \qquad c_v = \frac{R}{\gamma - 1}, \qquad \gamma = \frac{c_p}{c_v} \tag{2.18}$$

If we combine equations (2.13), (2.17) and (2.18), we can relate p and T to the state variables:

$$p = (\gamma - 1)\rho\left[E - \frac{1}{2}(u^2 + v^2)\right] \tag{2.19}$$

$$T = \frac{p}{(\gamma - 1)\rho}\frac{1}{c_v} \tag{2.20}$$

After writing the equations in non-dimensional form (which does not change the equations itself) we can define

$$\mu = \frac{\gamma^{1/2} M_\infty}{Re_\infty}\left[\frac{T_\infty + S_0}{(\gamma - 1)TT_\infty + S_0}\right]\left[(\gamma - 1)T\right]^{3/2} \tag{2.21}$$

where S_0 is a constant characteristic for the gas in study. For air at normal temperature, $S_0 = 110.4$ K.
M_∞ and Re_∞ are the Mach number and the Reynolds number, respectively. They are defined as

$$M_\infty = \frac{U_\infty}{c_\infty}, \quad Re_\infty = \frac{\rho_\infty U_\infty L}{\mu_\infty} \tag{2.22}$$

The index ∞ means the free stream value, and L is the characteristic length of the object (in the case of the present research, the airfoil chord length). Finally, c_∞ is defined as

$$c_\infty = \left(\frac{\gamma p_\infty}{\rho_\infty}\right)^{1/2} \tag{2.23}$$

2.2 The Navier-Stokes Equations in One Cell

In this section, the theory on how and why a 4×4 cell matrix can be used to compute artificial dissipation will be explained. And although this chapter treats the mathematical model, already some numerical theory will be used here.
In the system of equations treated above, we distinguish a convective part (together with the pressure) and a viscous part. The convective part describes the transport of a particle in a fluid caused by the flowing of this liquid. The viscous part mainly consists of second order spatial derivatives and describes the transport of a particle in a fluid caused by differences in concentration. The two parts are:

$$\mathbf{F}^c + \mathbf{G}^c = \begin{pmatrix} \rho u + \rho v \\ \rho u^2 + \rho uv + p \\ \rho uv + \rho v^2 + p \\ \rho uH + \rho vH \end{pmatrix} \equiv \mathbf{F_I}$$

and

$$\mathbf{F}^v + \mathbf{G}^c = \begin{pmatrix} 0 \\ -(\sigma_{xx} + \sigma_{xy}) \\ -(\sigma_{xy} + \sigma_{yy}) \\ -u(\sigma_{xx} + \sigma_{xy}) - v(\sigma_{xy} + \sigma_{yy}) + q_x + q_y \end{pmatrix} \equiv \mathbf{F_V}$$

respectively.
Equation (2.11) is integrated in each cell volume, giving (see also [13])

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{1}{V}\int_S [\mathbf{F}\mathbf{n}_x \cdot dS + \mathbf{G}\mathbf{n}_y \cdot dS] \tag{2.24}$$

which can be grouped as

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{1}{V}[(\mathcal{F_I} + \mathcal{F_V}) + (\mathcal{G_I} + \mathcal{G_V})] \tag{2.25}$$

Define $\mathcal{T} = \frac{\partial \mathbf{F_I}}{\partial \mathbf{q}}$ and $\mathcal{S} = \frac{\partial \mathbf{F_V}}{\partial \mathbf{q}}$. This gives

$$\mathbf{F_I} = \mathcal{T}\mathbf{q}, \quad \mathbf{F_V} = \mathcal{S}\mathbf{q} \tag{2.26}$$

If we write the inviscid part $(\mathcal{F}_\mathcal{I} + \mathcal{G}_\mathcal{I})$ as $\mathcal{F}_\mathcal{I}$, and the viscous part $(\mathcal{F}_\mathcal{V} + \mathcal{G}_\mathcal{V})$ as $\mathcal{F}_\mathcal{V}$, we have

$$\mathcal{F}_\mathcal{I} = \sum_{f=1}^{N_k} \mathbf{F_I}, \quad \mathcal{F}_\mathcal{V} = \sum_{f=1}^{N_k} \mathbf{F_V} \tag{2.27}$$

with N_k the number of faces surrounding cell k. $\sum_{f=1}^{N_k}$ is of course the discrete version of the line integral along Γ appearing in (2.24).
This way, we have constructed a matrix similar to the much simpler finite difference discretizations. Considering the resulting system matrix, we get to the following, symbolically written, equation:

$$\frac{\partial\{\mathbf{q}\}}{\partial t} = \frac{1}{V}[\mathcal{M}]\{\mathbf{q}\} \tag{2.28}$$

where $\{\mathbf{q}\}$ is the grid state vector $\{\rho_1, (\rho u)_1, (\rho v)_1, (\rho E)_1, \ldots, (\rho v)_N, (\rho E)_N\}$, and $[\mathcal{M}]$ is a 4×4 block tridiagonal matrix.
The idea is now to adjust this matrix, such that its eigenvalues become negative. This is done by considering only the 4×4 cell matrix. Let me first clarify this concept before I proceed with the rest of the theory.

2.2.1 Considering a 4×4 Cell Matrix

First, consider the two-dimensional Partial Differential Equation

$$\frac{\partial \phi}{\partial t} = P(\phi) \tag{2.29}$$

where $P(\phi)$ is a differential operator.
As we know from elementary discretization theory, we can write the discretized version of this PDE in two dimensions as

$$\frac{d\phi}{dt} = A(\phi_{i-1,j} - \phi_{i,j}) + B(\phi_{i+1,j} - \phi_{i,j}) + C(\phi_{i,j-1} - \phi_{i,j}) + D(\phi_{i,j+1} - \phi_{i,j}) \tag{2.30}$$

where $A, B, C, D \in \mathbb{R}$.
Equation (2.30) can be rewritten as

$$\frac{d\phi}{dt} = A\phi_{i-1,j} + B\phi_{i+1,j} + C\phi_{i,j-1} + D\phi_{i,j+1} - (A + B + C + D)\phi_{i,j} \tag{2.31}$$

If $A, B, C, D \geq 0$, the matrix originating from these coefficients is a K-matrix, which means that the eigenvalues of this matrix are in the negative half plane. In this case, ϕ is a one-dimensional vector. If we extend this theory to the four-dimensional vector $\mathbf{q}$ originating from the discretization of the Navier-Stokes equations, we have the following: Consider again equation (2.31). To obtain a K-matrix, the four coefficients have to be positive, which means that the coefficient of $\phi_{i,j}$ has to be negative. We can also say that the *eigenvalue* of this coefficient has to be negative. In the case of the Navier-Stokes equations, we have a similar case, only the one-dimensional coefficients A, B, C and D are now 4×4 matrices. The idea is to argue that the *eigenvalues* of the coefficient of $\phi_{i,j}$ again have to be negative to obtain a K-matrix. In other words, we ignore the contribution of

the neighbouring cells and only focus on the contribution of the 4×4 cell matrix itself. Santos ([13]) shows in his article that the eigenvalues of the cell matrix will become more negative if more artificial dissipation is added. This can be seen as a justification for the above idea: because adding artificial dissipation means an increase of stability of the system (i.e. we obtain a 'better' K-matrix), Santos' experiment shows that obtaining a K-matrix implies more negative eigenvalues of the cell matrix. The tricky part here is that the argumentation is reversed in this case; first we prescribe the eigenvalues, then the 'corresponding' artificial dissipation is computed.
We now have the formulation of the Navier-Stokes equations, confined to one cell. This way, we can consider the contribution of artificial dissipation, which affects the eigenvalues of this 4×4 cell matrix. For the sake of simplicity, we assume that the artificial dissipation only affects the diagonal elements of the matrix. Later we will see that this simplification is justified by the results.
Concluding, we make the following simplifications:

1. The Navier-Stokes equations are considered in one cell and by discretizing the Jacobian matrices we obtain a 4×4 cell matrix
2. The artificial dissipation only affects the diagonal elements of this cell

Having constructed the 4×4 matrix, we now have to use it in an Inverse Eigenvalue Method, which will be described in the next section.

2.3 The Inverse Eigenvalue Problem

As mentioned in the introduction, an Inverse Eigenvalue Method is a method for solving an Inverse Eigenvalue Problem, in other words for finding coefficients to a matrix such that this matrix has certain eigenvalues. In this thesis, I will consider the IEP for the restricted class

$$A(\underline{c}) = A_0 + \sum_{k=1}^{n} c_k A_k, \tag{2.32}$$

with A_k symmetric matrices, and later more refinements to the A_k's will be made. This problem is generally known as the *additive* IEP.
In [17, pp. 34–36] an algorithm is given to solve the IEP. It has its origin in a Newton iteration for solving a nonlinear system of n algebraic equations. As I implemented this algorithm in the existing Navier-Stokes solver, the focus on this theoretical analysis will mainly be on this algorithm and on the additive IEP mentioned in (2.32).
Let me first give the Inverse Eigenvalue Problem in a general form:

Problem 2.1 *Suppose we have given eigenvalues $\underline{\lambda}^\star \in \mathbb{C}^n$ and an $n \times n$ matrix $A(\underline{c})$ with eigenvalues $\underline{\lambda}(\underline{c})$. We want to find coefficients $\underline{c}^\star$ resulting in eigenvalues $\underline{\lambda}(\underline{c}^\star)$ for A, such that*

$$\underline{\lambda}(\underline{c}^\star) - \underline{\lambda}^\star = 0 \tag{2.33}$$

□

To solve this problem, we need the following theorem:

Theorem 2.1 *If we denote the current $\underline{c}$ by $\underline{c}^{(r)}$ and the corresponding matrix, eigenvalues and normalized eigenvectors by $A^{(r)}$, $\lambda_i^{(r)}$ and $\underline{x}_i^{(r)}$ respectively, and if we apply a Newton iteration to (2.33) we have the following:*
The first order perturbations in the eigenvalues corresponding to a perturbation $\underline{\delta}^{(r)}$ in $\underline{c}^{(r)}$ are given by $\mathbf{J}^{(r)}\underline{\delta}^{(r)}$, where

$$\mathbf{J}_{ik}^{(r)} = \underline{x}_i^{(r)T} A_k \underline{x}_i^{(r)}. \tag{2.34}$$

Now the resulting Newton iteration is as follows:

$$\mathbf{J}^{(r)}\underline{\delta}^{(r)} = \underline{\lambda}^* - \underline{\lambda}^{(r)}, \quad \underline{c}^{(r+1)} = \underline{c}^{(r)} + \underline{\delta}^{(r)} \tag{2.35}$$

and since $A^{(r)}\underline{x}_i^{(r)} = \lambda_i^{(r)}\underline{x}_i^{(r)}$, this can be written in the form:

$$\mathbf{J}^{(r)}\underline{c}^{(r+1)} = \underline{\lambda}^* + \mathbf{J}^{(r)}\underline{c}^{(r)} - \underline{\lambda}^{(r)} = \underline{\lambda}^* - \underline{b}^{(r)}, \tag{2.36}$$

where $b_i^{(r)} = \underline{x}_i^{(r)T} A_0 \underline{x}_i^{(r)}$.

□

Proof.

1. (2.34) and (2.35) can be proven in a straightforward way, using elementary perturbation theory.

2. (2.36) can be proven as follows:
   The first equality is straightforward, combining the two parts of 2.35. To prove the second equality, we have to consider the vectors by their entries. For convenience, I will skip the iteration index (r). Thus, we have to prove:

$$\underline{\lambda} - \mathbf{J}\underline{c} = \underline{b}, \quad \text{where } b_i = \underline{x}_i^T A_0 \underline{x}_i \tag{2.37}$$

   Rewriting $A\underline{x}_i = \lambda_i \underline{x}_i$ using (2.32) and multiplying both sides with $\underline{x}_i^T$ on the left gives

$$\lambda_i - \sum_{k=1}^{n} c_k \underline{x}_i^T A_k \underline{x}_i = \underline{x}_i^T A_0 \underline{x}_i \tag{2.38}$$

   On the righthandside, we recognize the i-th element of $\underline{b}$ and the sum on the lefthandside is exactly the i-th element of $\mathbf{J}\underline{c}$, as follows from (2.34).

■

2.3.1 The IEP In This Case

As mentioned before, the additive IEP will be considered here, i.e. the problem as stated in (2.32) and (2.33). The formulation given in the previous section needs some specification before we proceed. Following the argumentation of section 1.4 with $n = 4$, we have the following additive IEP:

Problem 2.2 *Given a 4×4 matrix $A(\underline{c})$ with eigenvalues $\{\lambda_i(\underline{c})\}_1^4$. Suppose (2.32) holds, and furthermore, $A_k = \underline{e}_k \underline{e}_k^T$. We now want to find coefficients $\underline{c} \in \mathbb{C}^4$ such that (2.33) holds.*

$\square$

Friedland ([5]) gives a quite thorough analysis of Inverse Eigenvalue Problems, although his main focus is on real and symmetric matrices. In his article, he considers the following additive problem:

Problem 2.3 *Find a diagonal complex valued matrix D such that the spectrum of $A + D$ is a given set $\lambda = \{\lambda_1, \ldots, \lambda_n\} \in \mathbb{C}$.*

$\square$

In his article, Friedland reformulates the Inverse Eigenvalue Problem as a minimization problem. He considers the *ω-inverse problem over $\mathcal{B}$*:

Problem 2.4 *Find $A^* \in \mathcal{B}$ such that*

$$\min_{\mathcal{B}} \rho_\omega(A) = \rho_\omega(A^*) \tag{2.39}$$

$\square$

where B is the closed set of $n \times n$ real symmetric matrices and

$$\rho_\omega(A) = \sum_{i=1}^{n} (\lambda_i(A) - \omega_i)^2. \tag{2.40}$$

Using this, a proof is given for the existence of a finite number of solutions in the complex case.

2.3.2 Choice of A_0 and A_k's

In the algorithm given in the beginning of this section, a general approach was given for a solution of Inverse Eigenvalue Problems. In this work, however, a few refinements and choices had to be made.

Because only changing the diagonal elements of the matrix was concerned here, we could refine definition (2.32). Now, A_0 is the original matrix, i.e. the matrix with the eigenvalues that need to be adjusted. If now $A_k = \underline{e}_k \underline{e}_k^T$, where $\underline{e}_k$ is the unit vector, we see that we have for $n = 4$:

$$A(\underline{c}) = A_0 + \begin{pmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \end{pmatrix} \tag{2.41}$$

We see that for these choices, we have the model for the Inverse Eigenvalue Method needed here.

As said before, many kinds of instabilities can occur in a numerical model, which have to be dealt with by adding artificial dissipation. Before giving a mathematical approach of this artificial dissipation, let me point out some of the difficulties that may occur dealing with non-linear PDEs.

2.4 Instabilities Due To Non-linearity

In general, viscous regions (generally described by second order derivatives) will have a numerically more stable behaviour than convective regions (generally described by first order derivatives). Instabilities in convective regions are often caused by nonlinear parts in the differential equation. This is shown in the example of the occurance of shock waves in Burgers' equation:

Example 2.2 [1] Burgers' equation is:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{2.42}$$

which we can write in conservation form:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\frac{1}{2}u^2) = 0 \tag{2.43}$$

From the theory of shocks, we know that we can now derive the velocity S of shock waves:

$$-S[u] + [\frac{1}{2}u^2] = 0 \tag{2.44}$$

where $[\cdot]$ denotes the jump of the argument of the shock. From here, we see that for the Burgers' equation the following relationship holds:

$$S = \frac{1}{2}(u_{left} + u_{right}) \tag{2.45}$$

It depends on the initial values whether a shock wave will occur or not. If $u_{left} > u_{right}$, we will have a shock wave with velocity $S = \frac{1}{2}$. For a more detailed description of this particular case, see [15, pp. 86–87]

□

As shown in this example, even in relatively simple PDEs, irregularities like shocks may occur. If the PDE can be solved analytically, this is not much of a problem, as these instabilities are usually easy to be dealt with. If the PDE has to be solved numerically, however, there is a problem. The numerical solution will show oscillations because it cannot handle these irregular phenomena. And this is where artificial dissipation comes in sight.

2.5 Artificial Dissipation

Due to discretization, instabilities can occur in the numerical solution. To avoid these instabilities, artificial dissipation is used. As an illustration, let me first give a simple example of how artificial dissipation is used in the discretization of a stationary convection-diffusion equation.

[1] Example taken from [15, pp. 86–87]

Example 2.3 [2] Consider the following equation:

$$u\frac{d\phi}{dx} - k\frac{d^2\phi}{dx^2} = 0, \quad 0 < x < L \tag{2.46}$$

And suppose $u > 0$ constant. After central discretization, this becomes

$$\frac{P}{2}(\phi_{i+1} - \phi_{i-1}) - (\phi_{i+1} - 2\phi_i + \phi_{i-1}) = 0 \quad (i = 1, \ldots, I - 1). \tag{2.47}$$

with $P = \frac{uh}{k}$ and boundary conditions $\phi_0 = T_0$ and $\phi_I = T_L$.
Consider the case $P \gg 0$. The solution of (2.47) can then be written as

$$\phi_i = T_0 + (T_L - T_0)\frac{1 - (-1)^i(1 + 2i\epsilon)}{1 - (-1)^I(1 + 2I\epsilon)} + O(\epsilon^2), \tag{2.48}$$

with $\epsilon = \frac{2}{P} \ll 1$. An analysis of this difference equation shows that the solution depends on whether I is odd or even.
If I is odd, we can prove that the solution in points where i is even corresponds with the left boundary condition and the solution in odd points corresponds with the right boundary condition. Furthermore, simplification of (2.47) yields

$$\phi_{i+1} - \phi_{i-1} = 0, \tag{2.49}$$

and we see that the solutions in points with odd and even i are completely independent.
For even I, the situation is slightly different. Analysis shows that in points where i is even, the solution corresponds with the boundary conditions, but in points where i is odd, the solution goes to infinity for $\epsilon \to 0$.
The phenomena described above are known as odd-even decoupling and they are typical for central discretization of PDEs with high P.
After upwind discretization however, we get to the following discretization:

$$P(\phi_i - \phi_{i-1}) - (\phi_{i+1} - 2\phi_i + \phi_{i-1}) = 0, \tag{2.50}$$

which gives a smoother (although in the boundary layer not completely correct!) solution.
After comparing (2.47) with (2.50), using

$$u\frac{\phi_i - \phi_{i-1}}{h} \equiv u\frac{\phi_{i+1} - \phi_{i-1}}{2h} - \frac{uh}{2}\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2}, \tag{2.51}$$

we see that upwind discretization of (2.46) gives the same as central discretization of

$$u\frac{d\phi}{dx} - \left(k + \frac{uh}{2}\right)\frac{d^2\phi}{dx^2} = 0 \tag{2.52}$$

□

From this example we can see that even in relatively simple PDEs artificial dissipation may have to be used to maintain stability. The instabilities that have to be corrected for are caused by:

[2]Example taken from [15, pp. 24–25]

1. odd-even decoupling,
2. regions in the flow field in which the convective terms dominate the viscous terms
3. oscillations caused by non-linearities such as shock waves

Usually, a nonlinear second-order difference term is added to control oscillations near shocks, and a linear fourth-order difference term is added to damp out oscillations in regions of (nearly) uniform flow (see [9]). These are regions in which the convective terms dominate the viscous terms. Because this fourth order dissipation overshoots near shocks, it has to be set to zero in those regions.
In adding artificial dissipation we have to take a few constraints into account:

1. The added dissipation is not allowed to reduce the accuracy of the solution of the scheme too much
2. The added dissipation should be conservative, i.e. summed over the complete flow field there cannot be a 'production' of mass, momentum or energy
3. Computing the artificial dissipation cannot take too much CPU time. This can especially become a problem when the Inverse Eigenvalue Method is used to compute the optimal artificial dissipation.

These restrictions make it quite difficult to find a proper way to add dissipation. Still, numerous methods have been found to do this addition. In the next section it is explained how this is done in the Navier-Stokes solver used in this research.

2.5.1 Artificial Dissipation in the Navier-Stokes solver

As we saw in section 2.2.1, the Navier-Stokes equations can symbolically be written as follows:

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{1}{V}[\mathcal{F}_{\mathcal{I}} + \mathcal{F}_{\mathcal{V}}] \tag{2.53}$$

However, in this model we do not have any dissipation added yet. After the addition of artificial dissipation, equation (2.53) becomes

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{1}{V}[(\mathcal{F}_{\mathcal{I}} + \mathcal{F}_{\mathcal{V}}) + \mathcal{D}] \tag{2.54}$$

Note that $\mathcal{D}$ has no physical meaning, and that it is only used to control oscillations.
The formulation of the artificial dissipative terms can be constructed in a number of different ways. In the example in the previous section, we saw that a constant 'amount' of artificial dissipation is added. However, many ways of defining artificial dissipation are possible. In a method proposed by Jameson, artificial dissipation is constructed by 'blending' a Laplacian and a Biharmonic operator, multiplied by scaling and switching coefficients (see [4]). This is done in such a way that the original conservative form is maintained, using a flux balance formulation.
The part involving the Biharmonic operator dominates the regions where the solution is smooth (generally the viscous regions), but it is switched off near shock waves. How this is done numerically will be explained in section 3.2. Here, I will give a brief notice of how

the artificial dissipative operators are constructed.
The Laplacian operator is constructed by summing, over all the edges of the control volume, the difference between the flow variables across each edge:

$$\nabla^2 \mathbf{q}_k = \sum_{i=1}^{N_k} (\mathbf{q}_i - \mathbf{q}_k) \tag{2.55}$$

(note that the index i denotes the value of the state vector in the cell centers, not the cell edges!). The summation is over the neighbours of cell k, the number of which is N_k. The Biharmonic operator is constructed by repeating this step, which gives (see [4])

$$\nabla^4 \mathbf{q}_k = \sum_{i=1}^{N_k} (\nabla^2 \mathbf{q}_i - \nabla^2 \mathbf{q}_k) \tag{2.56}$$

Now we can write

$$\mathcal{D} = \mathcal{D}^2(\mathbf{q}_k) - \mathcal{D}^4(\mathbf{q}_k) \tag{2.57}$$

where

$$\mathcal{D}^2(\mathbf{q}_k) = \sum_{i=1}^{N_k} \epsilon_{ik} \frac{S_{ik}}{\Delta t_{ik}} (\mathbf{q}_i - \mathbf{q}_k) \tag{2.58}$$

and

$$\mathcal{D}^4(\mathbf{q}_k) = \sum_{i=1}^{N_k} \epsilon^{(2)} \frac{S_{ik}}{\Delta t_{ik}} (\nabla^2 \mathbf{q}_i - \nabla^2 \mathbf{q}_k) \tag{2.59}$$

The coefficient ϵ_{ik} is an adaptive pressure switch which turns on the Laplacian terms in the vicinity of the shock and turns it off in smooth regions. A further explanation of this matter can be found in chapter 3.

2.5.2 Artificial Dissipation and Convergence

Let us now investigate the relationship between the amount of added artificial dissipation and convergence rate of the method. We expect the method to converge slower when less artificial dissipation is added, as this means that the eigenvalues will be 'less negative' (see also [13]).
In the test cases described below, five cases were considered. In all cases except the case in which Jameson's coefficients were used, the coefficient for the second order dissipation was set to 0.8. In the second case this coefficient was set to 1.0, according to [10, p. 185] (see also section 3.2.1). The fourth order dissipation coefficient (`vis2` in the solver) was set to 0.0, $\frac{1}{32}$, 0.1, 0.2 and 0.4 respectively. In figure 2.2, we see the comparison between these five cases. The solid line shows the convergence rate with `vis2` = 0.0, the dashed line shows the case in which Jameson's coefficients were used, the dotted line shows the case `vis2` = 0.1, the dash-dotted line shows the case where `vis2` = 0.2 and the second solid line shows the case `vis2` = 0.4.
We see that the higher the coefficient, the better the method converges, until the coefficient takes the value 0.2. The convergence rate does not get significantly better as `vis2` gets above this number, which justifies the choice of 0.2 for this coefficient in the course of this research. As we see, the error level does not get below 10^{-7}. This is caused by the fact that the machine accuracy is around 7 figures.

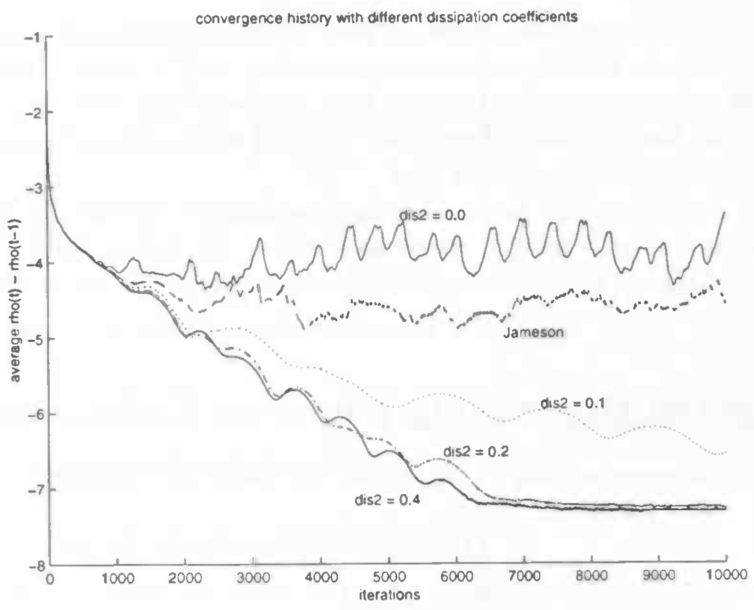


Figure 2.2: Convergence rate of solver with and without the addition of artificial dissipation

Chapter 3

Numerical Model

In chapter 2, an example was given on how a partial differential equation is reformulated in an integral form, which gives a more convenient way to treat this PDE. In this chapter, I will point out how such an integral form is discretized and how this discretization is applied to the Navier-Stokes equations. Because of the general form of this method, the algorithm can handle any type of grid.
The discretization in space and time is done separately. For the time integration, a Runge-Kutta scheme has been used in the present solver. Therefore, only this type of discretization will be treated here, although there are of course a lot of different discretization methods.

3.1 Spatial Discretizing Using the Finite Volume Method

In the example given in chapter 2, we had the following equation:

$$\frac{\partial}{\partial t}\iint\limits_{\Omega} \mathbf{q}\ d\Omega + \oint\limits_{\Gamma}(\mathbf{F}\ dy - \mathbf{G}\ dx) = 0 \tag{3.1}$$

Suppose we now have a grid over Ω which we denote as Ω_k. Of course (3.1) still holds, with Ω and Γ substituted by Ω_k and Γ_k, respectively.
Consider the k-th cell, denoted with $\Omega_k^\star$, with edge $\Gamma_k^\star$. As the precise location and value of the variable $\mathbf{q}$ inside $\Omega_k^\star$ cannot be specified explicitly, we have to find an approximation of $\mathbf{q}$. This is done by considering $\mathbf{q}_k$ in the cell center and defining it as the averaged value of $\mathbf{q}$ in $\Omega_k^\star$:

$$\mathbf{q}_k = \frac{1}{S_k}\iint\limits_{\Omega_k^\star} \mathbf{q}\ d\Omega_k^\star, \tag{3.2}$$

where S_k is the area of $\Omega_k^\star$.
If $\mathbf{F}_i$ and $\mathbf{G}_i$ denote the values of $\mathbf{F}$ and $\mathbf{G}$ along the edges respectively, we can approximate the flux integral of (3.1) as

$$\oint\limits_{\Gamma_k^\star}(\mathbf{F}\ dy - \mathbf{G}\ dx) \approx \sum_{i=1}^{N_k}(\mathbf{F}_i\Delta y_i - \mathbf{G}_i\Delta x_i) \tag{3.3}$$

so that the approximate evaluation of (3.1) becomes

$$S_k \frac{\partial}{\partial t}\mathbf{q}_k + \sum_{i=1}^{N_k}(\mathbf{F}_i \Delta y_i - \mathbf{G}_i \Delta x_i) = 0 \tag{3.4}$$

where the index i denotes the edges, N_k is the number of edges surrounding $\Omega_k^\star$ and Δx_i and Δy_i are the increments of x and y along that edge.
In the solver treated here, the convective and viscous terms are evaluated separately. For the computation of these terms, a 'secondary cell' is constructed, which is done by connecting the cell centers K and L of the neighbouring cells and the vertices M and N of the two neighbouring cells. In figure 3.1, an example of such a secondary cell is shown (solid line) together with the original cells (dashed line). Note that the index i from $\mathbf{q}_i$ is

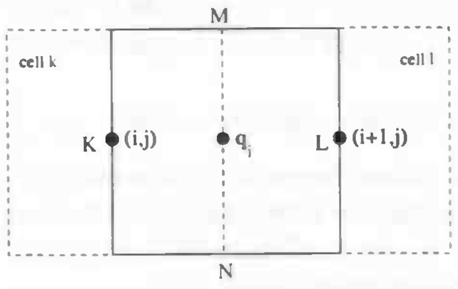


Figure 3.1: Example of a 'secondary' cell

used to denote the value of the flux vector on the edges, whereas the index i in (i, j) and $(i + 1, j)$ denotes the x-coordinate from the grid points.

3.1.1 Calculation of the convective terms

The evaluation of the convective terms of the flux along the edges depends on the selected scheme as well as on the location of the flow variables with respect to the grid. If an upwind scheme is used, the cell face fluxes are determined according to the propagation direction of the wave components. In this Navier-Stokes solver, however, a central scheme is used, which is merely based on local flux estimation. The flow variables are associated with the central point of the cells (i.e. K and L of cells k and l respectively, in figure 3.1). In order to preserve the flux balance in the cell, an integration is performed over the cell edges. The approximation of the convective part of this integration (see (3.4)), is given by a summation over the edges:

$$C(\mathbf{q}_k) = \sum_{i=1}^{N_k}(\mathbf{F}_i^c \Delta y_i - \mathbf{G}_i^c \Delta x_i) \tag{3.5}$$

where N_k is again the number of edges of cell k, $\mathbf{F}^c$ and $\mathbf{G}^c$ the convective parts of the righthandside of (3.4) and $C(\mathbf{q}_k)$ is the convective operator for cell k. If we introduce the flux velocity Q_i for edge i (delimiting cells k and l) as

$$Q_i = u_i \Delta y_i - v_i \Delta x_i, \tag{3.6}$$

$C(\mathbf{q}_k)$ can be written as

$$C(\mathbf{q}_k) = \sum_{i=1}^{N_k} \begin{pmatrix} Q_i\rho_i \\ Q_i\rho_i u_i + p_i\Delta y_i \\ Q_i\rho_i v_i - p_i\Delta x_i \\ Q_i\rho_i H_i \end{pmatrix} \tag{3.7}$$

We see from (3.7) that the flow variables on the edges need to be computed in order to evaluate $C(\mathbf{q}_k)$. In the Navier-Stokes solver used here, these flow variables are computed by taking the average of the values in the neighbouring cells k and l, i.e.

$$\mathbf{q}_i = \frac{1}{2}(\mathbf{q}_k + \mathbf{q}_l) \tag{3.8}$$

If i represents a boundary edge, one of the cells is outside the flow domain. Therefore, an imaginary row of cells is created along both the flow and the wall boundary.

3.1.2 Calculation of the viscous terms

The calculation of the viscous terms is slightly more difficult than that of the convective terms, because of the presence of the viscous stress tensors and heat flux components. The approximation of the viscous flux integrals appearing in equation (3.1) can be formulated in a similar way as equation (3.5):

$$V(\mathbf{q}_k) = \sum_{i=1}^{N_k}(\mathbf{F}_i^v\Delta y_i - \mathbf{G}_i^v\Delta x_i) \tag{3.9}$$

where $V(\mathbf{q}_k)$ is the viscous operator for cell k. We introduce the following variables:

$$\begin{aligned} P_i &= -(\sigma_{xx})_i\Delta y_i + (\sigma_{xy})_i\Delta x_i \\ Q_i &= -(\sigma_{xy})_i\Delta y_i + (\sigma_{yy})_i\Delta x_i \\ R_i &= (q_x)_i\Delta y_i - (q_y)_i\Delta x_i \end{aligned} \tag{3.10}$$

so that $V(\mathbf{q}_k)$ can be written as

$$V(\mathbf{q}_k) = \sum_{i=1}^{N_k} \begin{pmatrix} 0 \\ P_i \\ Q_i \\ u_iP_i + v_iQ_i + R_i \end{pmatrix} \tag{3.11}$$

The shear stresses and the heat flux components are proportional to the first derivatives of the velocity components (u and v) and the temperature (T), respectively, as we see in (2.15). Consequently, the velocity and temperature gradients can be computed at the cell edges (MN in figure 3.1). The complete computations concerning the viscous terms are performed in two steps. The first step is to compute the gradients of u, v and T, the second step involves the calculation of the viscous flux balance across the cell edges.

3.2 Adding Artificial Dissipation in a Numerical Model

After the spatial discretization of the Navier-Stokes equations, we get to the following, symbolically written, difference equation:

$$S_k \frac{\partial}{\partial t}(\mathbf{q}_k) + C(\mathbf{q}_k) + V(\mathbf{q}_k) = 0, \quad k = 1, \dots, N \tag{3.12}$$

with $C(\mathbf{q}_k)$ the convective and $V(\mathbf{q}_k)$ the viscous operator and N the number of cells.
In principle, the viscous terms in the Navier-Stokes equations can provide a numerical scheme which has the dissipative properties necessary to damp out oscillations. In cases with a high Reynolds number, however, regions in which the convective terms dominate the viscous effect, instabilities may occur, caused by shock wave and other effects due to the non-linearity of the equations. To damp out these instabilities, artificial dissipation has to be added, as was shown in chapter 2. After having added this dissipation, equation (3.12) looks like

$$S_k \frac{\partial}{\partial t}(\mathbf{q}_k) + C(\mathbf{q}_k) + V(\mathbf{q}_k) - D(\mathbf{q}_k) = 0 \tag{3.13}$$

Let us now take a closer look at how and where this artificial dissipation is added into a numerical scheme. In chapter 2, three conditions were given which the addition of artificial dissipation has to satisfy. Furthermore, in the same chapter we saw that many different kinds of artificial dissipation exist in mathematical theory. As we saw in section 2.5, dissipative terms can be constructed by blending Laplacian and Biharmonic operators multiplied by scaling and switching coefficients. The numerical treatment of these dissipative terms will be considered in the next section.

3.2.1 Artificial Dissipation in the present solver

If we write the artificial dissipation from (2.54) and (3.13) as

$$\mathcal{D} = \mathcal{D}_x + \mathcal{D}_y \tag{3.14}$$

with $\mathcal{D}_x$ and $\mathcal{D}_y$ the dissipation flux in x and y direction, respectively:

$$\mathcal{D}_x = d_{i+\frac{1}{2},j} - d_{i-\frac{1}{2},j} \tag{3.15}$$

$$\mathcal{D}_y = d_{i,j+\frac{1}{2}} - d_{i,j-\frac{1}{2}} \tag{3.16}$$

The first component in the x direction is, using (2.57), (2.58) and (2.59) with $\epsilon_{ik} = \epsilon_{i+\frac{1}{2},j}$ and $S_{ik} = S_k$:

$$d_{i+\frac{1}{2},j} = \left(\frac{S_k}{\Delta t_k}\right)_{i+\frac{1}{2},j} \{\epsilon_{i+\frac{1}{2},j}(\mathbf{q}_{i+1,j} - \mathbf{q}_{i,j}) - \epsilon^2_{i+\frac{1}{2},j}(\mathbf{q}_{i+2,j} - 3\mathbf{q}_{i+1,j} + 3\mathbf{q}_{i,j} - \mathbf{q}_{i-1,j})\} \tag{3.17}$$

where k denotes the present cell $i + \frac{1}{2}, j$ and Δt is the local time step. More on the maximum allowable time step is found in the section on Time Integration.
As explained in chapter 2, $\epsilon_{i+\frac{1}{2},j}$ should be turned on in the vicinity of shocks. It should be clear that a proper choice of $\epsilon_{i+\frac{1}{2},j}$ is dependent on the pressure coefficients. This way, the presence of shocks can be noticed. An effective sensor of the pressure of a shock wave

can be constructed by taking the second difference of the pressure (see [10, p.184]). If we define

$$\delta_{i,j} = \frac{|p_{i+1,j} - 2p_{i,j} + p_{i-1,j}|}{|p_{i+1,j} + 2p_{i,j} + p_{i-1,j}|} \tag{3.18}$$

we can write $\epsilon_{i+\frac{1}{2},j}$ as

$$\epsilon_{i+\frac{1}{2},j} = \epsilon^{(1)}\{\max(\delta_{i+1,j}, \delta_{i,j})\} \tag{3.19}$$

with $\epsilon^{(1)}$ a given coefficient.
Because the Biharmonic terms can be destabilizing near shocks, they have to be turned off. This is done by $\epsilon^2_{i+\frac{1}{2},j}$, which we define as follows:

$$\epsilon^2_{i+\frac{1}{2},j} = \max\{0, \epsilon^{(2)} - \epsilon_{i+\frac{1}{2},j}\} \tag{3.20}$$

This way, we assure that the artificial dissipation does not become negative, and it will be turned off if the second order dissipation becomes large (i.e., in the vicinity of shocks).
In the literature, different choices for $\epsilon^{(1)}$ and $\epsilon^{(2)}$ are given. Jameson chooses $\epsilon^{(1)} = 1$ and $\epsilon^{(2)} = \frac{1}{32}$ (see also [10, pp. 184–185]). In this research I took $\epsilon^{(1)} = 0.8$, based on [4], and $\epsilon^{(2)}$ was to be replaced by an appropriate factor, which will be computed by the Inverse Eigenvalue Method. In the case a regular solver was iterated, $\epsilon^{(2)}$ was set to 0.2, as explained in chapter 2.
With the artificial dissipation added, the spatial discretization of the Navier-Stokes equations is done, so now we have to find a way for time integration needed to solve the obtained system of differential equations.

3.3 Time Integration

From the spatial discretizations and the addition of artificial dissipation as described in the previous section, we obtain a set of coupled ordinary differential equations:

$$\frac{d}{dt}\mathbf{q}_k = -R(\mathbf{q}_k), \quad k = 1, \ldots, N \tag{3.21}$$

where N is the total number of cells and $R(\mathbf{q}_k)$ is the residual:

$$R(\mathbf{q}_k) = \frac{1}{S_k}\left[C(\mathbf{q}_k) + V(\mathbf{q}_k) - D(\mathbf{q}_k)\right] \tag{3.22}$$

with S_k the surface of cell k.
To integrate this system of ODEs, various integration methods can be used. In this case, a Runge-Kutta scheme was used. The general form of this scheme is as follows:
Let $\mathbf{q}_k^n$ be the value of $\mathbf{q}_k$ after n iterations. The righthandside of (3.21) is now evaluated at several values in the interval between $n\Delta t$ and $(n+1)\Delta t$ and these values are combined in order to obtain a higher-order approximation of $\mathbf{q}_k^{n+1}$. The general form of an m-stage Runge-Kutta scheme is as follows:

$$\mathbf{q}_k^{(0)} = \mathbf{q}_k^n$$

$$\mathbf{q}_k^{(1)} = \mathbf{q}_k^{(0)} - \alpha_1 \Delta t_k R(\mathbf{q}_k^{(0)})$$

$$\mathbf{q}_k^{(2)} = \mathbf{q}_k^{(0)} - \alpha_2 \Delta t_k R(\mathbf{q}_k^{(1)})$$
$$\vdots$$
$$\mathbf{q}_k^{(m)} = \mathbf{q}_k^{(0)} - \alpha_m \Delta t_k R(\mathbf{q}_k^{(m-1)})$$
$$\mathbf{q}_k^{n+1} = \mathbf{q}_k^{(m)}$$

where Δt_k is the discrete time step as explained before and $\alpha_1, \ldots, \alpha_m$ are coefficients depending on the number of stages.
To avoid the computation of all the residual terms, we can use the 'hybrid formulation'. In this formulation, only the convective operator is evaluated at every stage in the process, and the viscous and the artificial dissipative operators are eveluated only at the first stage. Thus, a three stage scheme, as was used in the present research, is as follows:

$$\begin{aligned}
\mathbf{q}_k^{(0)} &= \mathbf{q}_k^{n} \\
\mathbf{q}_k^{(1)} &= \mathbf{q}_k^{(0)} - \alpha_1 \frac{\Delta t_k}{S_k}\left[C(\mathbf{q}_k^{(0)}) + V(\mathbf{q}_k^{(0)}) - D(\mathbf{q}_k^{(0)})\right] \\
\mathbf{q}_k^{(2)} &= \mathbf{q}_k^{(0)} - \alpha_2 \frac{\Delta t_k}{S_k}\left[C(\mathbf{q}_k^{(1)}) + V(\mathbf{q}_k^{(0)}) - D(\mathbf{q}_k^{(0)})\right] \\
\mathbf{q}_k^{(3)} &= \mathbf{q}_k^{(0)} - \alpha_3 \frac{\Delta t_k}{S_k}\left[C(\mathbf{q}_k^{(2)}) + V(\mathbf{q}_k^{(0)}) - D(\mathbf{q}_k^{(0)})\right] \\
\mathbf{q}_k^{n+1} &= \mathbf{q}_k^{(3)}
\end{aligned} \tag{3.23}$$

with coefficients $\alpha_1 = 0.6,\ \alpha_2 = 0.6,\ \alpha_3 = 1.0$.
The local time step Δt_k must be chosen to be consistent with the stability limitation due to the inviscid and viscous characteristics of the Navier-Stokes equations. In the present solver, the following time step is chosen (see [4]):

$$\Delta t_k = \frac{1}{\lambda_k^c / A_k + \lambda_k^v / A_k^2} \tag{3.24}$$

where λ_k^c and λ_k^v are defined by the characteristic convective and viscous propagation speeds respectively, and are given by

$$\lambda_k^c = \oint_{\Gamma_k} |u dy - v dx| + c(dx^2 + dy^2)^{1/2} \tag{3.25}$$

$$\lambda_k^v = \oint_{\Gamma_k} \frac{2\mu}{\rho}(dx^2 + dy^2) \tag{3.26}$$

with c the local speed of sound. These equations are approximated in the usual way, and thus the maximum allowable time step is computed. In the solver, this is done in subroutine `step`, which is not shown here, as it merely speaks for itself.
Note that this time step is also used in the computation of the artificial dissipation, which gives the same result as Jameson in [10].

3.4 Description of the Used Navier-Stokes Solver

The Navier-Stokes solver used in this research was developed by Dantje K. Natakusumah and it was provided by the Indonesian aircraft company IPTN. The important parts of the original program are those in which the convective and the viscous terms are computed, the time integration part, and the part in which the artificial dissipation is computed, respectively. These subroutines will be treated below.

3.4.1 Calculation of the Convective Terms

According to the formulas given in section 3.1.1, the numerical implementation of the convective terms is as follows:

```
          do 220 i=1,icmax
          do 220 j=1,lcol(i)
            n3        = iedg3(j,i)
            n4        = iedg4(j,i)
            dx        = xp(n1) - xp(n2)
            dy        = yp(n1) - yp(n2)
            pa        = 0.5*( p(n3) + p(n4) )
            wa1       = 0.5*( w(1,n3) + w(1,n4) )
            wa2       = 0.5*( w(2,n3) + w(2,n4) )
            wa3       = 0.5*( w(3,n3) + w(3,n4) )
            wa4       = 0.5*( w(4,n3) + p(n3) + w(4,n4) + p(n4) )
            qs        = ( dy*wa2 - dx*wa3 )/wa1
            fs1       = qs*wa1
            fs2       = qs*wa2 + dy*pa
            fs3       = qs*wa3 - dx*pa
            fs4       = qs*wa4
            dw(1,n3) = dw(1,n3) + fs1
            dw(2,n3) = dw(2,n3) + fs2
            dw(3,n3) = dw(3,n3) + fs3
            dw(4,n3) = dw(4,n3) + fs4
            dw(1,n4) = dw(1,n4) - fs1
            dw(2,n4) = dw(2,n4) - fs2
            dw(3,n4) = dw(3,n4) - fs3
            dw(4,n4) = dw(4,n4) - fs4
    220   continue
```

As we see, pa and wa1, ..., wa4 denote the averaged values of the pressure and the four flow varibles. As stated in section 3.1.1, the flow variables are associated in the cell centers, and thus, to get the values at the edges, the mean of their values at the centers is taken. Furthermore, in qs we recognize $Q_i = u_i \Delta y_i - v_i \Delta x_i$. In the continuous case qs can be written as $\frac{\rho u \Delta y - \rho v \Delta x}{\rho}$, which in turn yields (3.6).
Finally, the computed terms are added to the flow variables. Note that the contribution of the convective terms is added to one cell, and substracted from the neighbouring cell. This is done to preserve the flux balance. This concludes the computation of the convective terms.

3.4.2 Calculation of the Viscous Terms

Corresponding with the theory from section 3.1.2, the part of the solver computing the viscous terms is as follows:

```
*
* define constants
*
      g1     = gamma - 1.0
      const1 = 4./3.
      const2 = 2./3.
      const3 = sqrt(gamma)*g1**1.5*rm/rin
      pr     = 0.72
      prt    = 0.9
      twbc   = 1
*
* ***** Begin loop over edges
*
      dye   = yp(n2) - yp(n1)
      dxe   = xp(n2) - xp(n1)
      te    = 0.5*( t3 + t1 )
      ue    = 0.5*( u3 + u1 )
      ve    = 0.5*( v3 + v1 )
      cvl   = const3*te**1.5*( tin + 110.4 )/( te*g1*tin + 110.4 )
      cvta  = 0.5*( cvt( n1 ) + cvt( n2 ) )
      cvl   = cvl/pr + cvta/prt
      tauxx = const1*dudx - const2*dvdy
      tauxy = dudy + dvdx
      tauyy = const1*dvdy - const2*dudx
      viscx = cvl*( tauxx*dye - tauxy*dxe )
      viscy = cvl*( tauxy*dye - tauyy*dxe )
      visce = ue*viscx + ve*viscy +
     :          cvl*gamma*( dtdx*dye - dtdy*dxe )
*
* viscous flux balance
*
      fw(2,n3) = fw(2,n3) + viscx
      fw(3,n3) = fw(3,n3) + viscy
      fw(4,n3) = fw(4,n3) + visce
      fw(2,n4) = fw(2,n4) - viscx
      fw(3,n4) = fw(3,n4) - viscy
      fw(4,n4) = fw(4,n4) - visce
*
* ***** End loop
*
```

The first five terms in the loop are, of course, Δy_i, Δx_i, and the averaged values of the temperature and the velocity components. In `cvl`, we recognize the value of μ as defined

in (2.21). The term cvta is, as we see, the mean of the values of cvt in the neighbouring cells. These two values come from the subroutine viscot, where the viscous flux balance is computed. After redefining cvl, we see that it is exactly the μ_{tot} from (2.16). The other terms are more or less logically defined. tauxx, tauxy and tauyy are σ_{xx}, σ_{xy} and σ_{yy} respectively and in viscx, viscy and visce we recognize respectively P_i, Q_i and R_i from (3.10).
Finally, to maintain the flux balance, the contribution of the viscous terms is substracted from the cell on one side of the edge, and added to the cell on the other side of the edge. This concludes the computation of the viscous terms.

3.4.3 Treatment of Artificial Dissipation in This Case

In section 2.5, a theoretical approach was given on how the addition of artificial dissipation takes place in this Navier-Stokes solver. We saw that the dissipative terms consist of two parts, a second-order part to damp out oscillations near shocks etc., and a fourth-order part which was switched on in smooth regions of the flow field. The main part of the subroutine in the Navier-Stokes solver which computes the artificial dissipation is:

```
      dtl3      = vol(n3)/dtl(n3)
      dtl4      = vol(n4)/dtl(n4)
      sum       = f(n3) + f(n4)
      fil       = amin1( dtl3,dtl4 )/sum
      dis1      = fil*fis1*amax1( ep(n3),ep(n4) )
      dis2      = fil*fis2
      dis2      = dim(dis2,dis1)
      dw1       = w(1,n3) - w(1,n4)
      dw2       = w(2,n3) - w(2,n4)
      dw3       = w(3,n3) - w(3,n4)
      dw4       = w(4,n3) - w(4,n4) + p(n3) - p(n4)
c
c modified dissipation
c
      fs1       = dis1*dw1 - dis2*(ew(1,n3) - ew(1,n4))
      fs2       = dis1*dw2 - dis2*(ew(2,n3) - ew(2,n4))
      fs3       = dis1*dw3 - dis2*(ew(3,n3) - ew(3,n4))
      fs4       = dis1*dw4 - dis2*(ew(4,n3) - ew(4,n4))
      fw(1,n3) = fw(1,n3) - fs1
      fw(2,n3) = fw(2,n3) - fs2
      fw(3,n3) = fw(3,n3) - fs3
      fw(4,n3) = fw(4,n3) - fs4
      fw(1,n4) = fw(1,n4) + fs1
      fw(2,n4) = fw(2,n4) + fs2
      fw(3,n4) = fw(3,n4) + fs3
      fw(4,n4) = fw(4,n4) + fs4
```

As we take a closer look at this part of the program, we see that it corresponds with the theory from the previous section. The variables dtl3 and dtl4 correspond with the

factor $(\frac{S_k}{\Delta t_k})_{i-\frac{1}{2},j}$ and $(\frac{S_k}{\Delta t_k})_{i+\frac{1}{2},j}$, respectively, sum and fil are to determine the scaled minimum dtl. Then dis1 is computed, according to (3.19). Note that fis1 and fis2 are the prescribed values $\epsilon^{(1)}$ and $\epsilon^{(2)}$. Following (3.20), dis2 is determined, where the function dim(dis2,dis1) actually means max(0, dis2 − dis1).
Finally, the actual contribution of the artificial dissipation is computed and stored in fs1 to fs4. And again, to preserve the flux balance, this contribution is added to the cell on one side of the edge, and substracted from the cell on the other side of the edge.
With the addition of artificial dissipation, we have finished the spatial discretization of the Navier-Stokes equations. We now have the righthandside which is to be used by the Runge-Kutta scheme for time integration.

3.4.4 Time Integration

In this solver, a three stage Runge-Kutta method has been used for time integration, as explained above. This Runge-Kutta scheme has been implemented in the subroutine euler. The main lines of this subroutine are the following:

```
*
* compute artificial dissipation terms
*
      call diss( fw,icyc )
*
* compute viscous terms
*
      if( ivisc.ge.1 ) call visco( fw,icyc,ivisc )
*
*** Begin Runge-Kutta scheme
*
      do 200 k=1,mstage
        fn = 0.5*c(k)*cfl
*
        do 210 i=1,ncell
        do 210 j=1,4
          dw(j,i) = fw(j,i)
 210    continue
*
* calculation of the convective terms
*
        do 250 i=1,ncell
          dt      = fn*dtl(i)/vol(i)
          dw(1,i) = dt*dw(1,i)
          dw(2,i) = dt*dw(2,i)
          dw(3,i) = dt*dw(3,i)
          dw(4,i) = dt*dw(4,i)
 250    continue
*
        do 260 i=1,ncell
```

```
            w(1,i) = w0(1,i) + dw(1,i)
            w(2,i) = w0(2,i) + dw(2,i)
            w(3,i) = w0(3,i) + dw(3,i)
            w(4,i) = w0(4,i) + dw(4,i)
            qq     = 0.5*( w(2,i)**2 + w(3,i)**2 )/w(1,i)
            p(i)   = ( gamma - 1.0 )*( w(4,i) - qq )
  260    continue
  200 continue
*
*** End Runge-Kutta scheme
*
```

If we compare these lines with the theory from section 3.3, we can immediately see correspondences. First, the artificial dissipation and the viscous parts ($D(\mathbf{q}_k)$ and $V(\mathbf{q}_k)$ in section 3.3) are computed and stored in fw. These two operators are computed beforehand, as only the convective operator is evaluated at every stage.
In the loop, first the α_i's are defined and stored in fn. For consistency reasons, they are multiplied by 0.5 and the CFL-number ($CFL = 3.5$). The next steps are straightforward. First, the variable dw is initialized by adding the dissipative and the viscous terms. Then, with this variable the convective terms are computed (see section 3.4.1). Finally, the actual Runge-Kutta iteration is performed, where dt $= -\alpha_i \frac{\Delta t_k}{S_k}$, dtl(i) is the maximum allowable time step Δt_k from equation (3.24), and vol(i) is S_k in equation (3.23). The four lines in which dw is changed, correspond with the second part of the righthandside of the routine as given in (3.23).
Finally, in the last loop the values for w, the main vector with the flux variables, are substituted according to the routine. Furthermore, the pressure is computed here following equation (2.19). This concludes the complete discretization of the Navier-Stokes equations for the present solver.

3.5 Description of the Used Inverse Eigenvalue Method

As already mentioned, the Inverse Eigenvalue Method used here is an additive Eigenvalue Mathod. In writing the program, I used the algorithm for the Inverse Eigenvalue Method as mentioned in [17, pp. 34–36]. In this section, I will give a detailed description of the used subroutines.

3.5.1 Notation

In accordance with this program, the following notation will be used:

$$\begin{aligned} \text{eigenvalues}: &\quad \lambda_1, \ldots, \lambda_n \\ \text{eigenvectors}: &\quad \underline{x}_1, \ldots, \underline{x}_n \\ \text{elements of } i^{th} \text{ eigenvector}: &\quad (x_{i1}, \ldots, x_{in}) \end{aligned}$$

Having derived the 4×4 cell matrix (see section 2.2.1), we can now process it in the Inverse Eigenvalue Method. The next sections describe how this was done.

3.5.2 Computing Eigenvalues and Eigenvectors

For the computation of the eigenvalues of the 4×4 cell matrix, an EISPACK routine was used. This subroutine, cgeev, uses the following parameters (a '*' means that the value of this parameter will be changed by cgeev):

A*	complex nonsymmetric input matrix
LDA	leading dimension of A
N	order of A and V
E*	contains the eigenvalues of A
V*	if JOB $= 0$, V is not referenced otherwise, eigenvectors of A are stored in columns of V
LDV	leading dimension of V, if JOB is nonzero
WORK*	temporary storage vector
JOB	set nonzero only if eigenvectors have to be computed
INFO*	$= 0$ if calculation is succesful $= k$ if eigenvalues $k+1$ through N are correct, but no eigenvectors were computed

For a detailed description of this routine I refer to the NETLIB homepage at **http://www.netlib.org/index.html**.
Before the IEM will be explained, let me first give a description of the subroutine gauss, used to solve systems of equations.

3.5.3 Solving a System of Equations

In the implementation of the Inverse Eigenvalue Method, several (linear) systems of equations have to be solved. More specifically, these systems arise in the computation of the eigenvectors and in the actual Inverse Eigenvalue Method. Therefore, the subroutine gauss was written, which solves arbitrary systems of equations.
Suppose the system which has to be solved is $S\underline{x} = \underline{b}$, where S is an $n \times n$ matrix, and $\underline{x}$ and $\underline{b}$ n-tuples. The input for this subroutine consists of S, n and l, with one modification, however. For convenience, S is here transformed into an $n \times (n+1)$ matrix, which for $n = 4$ is constructed as follows:

$$S_{4,5} = \left(\begin{array}{cccc|c} s_{11} & s_{12} & s_{13} & s_{14} & b_1 \\ s_{21} & s_{22} & s_{23} & s_{24} & b_2 \\ s_{31} & s_{32} & s_{33} & s_{34} & b_3 \\ s_{41} & s_{42} & s_{43} & s_{44} & b_4 \end{array} \right) \tag{3.27}$$

The subroutine consists of several sub-subroutines, as we see in this overview:

```
        subroutine gauss(S, n, l, x)
C
        complex x(n), S(n,l)
```

```
C
      nn = n - 1
      do 90 k = 1, nn
           call find(S, n, n+1, k, j)
           call change(S, n, n+1, k, j)
           kk = k + 1
           do 80 i = kk, n
                call rowmul(S, n, n+1, k, i, -S(i,k) / S(k,k))
 80        continue
 90   continue
      call bksub(S, n, n+1, x)
      return
      end
```

The global structure of this subroutine is that first the matrix is brought to an upper diagonal form, and then backwards substituted. The upper diagonal form is obtained as follows:
The subroutine `find` finds the row j with the largest absolute value among $S(k,k)$, $S(k+1,k),\dots,S(n,k)$, and then this j-th row is changed with the current row in `change`. Next, to each row (i.e., for $i=k+1,\dots,n$) we add the k-th row, multiplied by $\frac{-S(i,k)}{S(k,k)}$, which is done in subroutine `rowmul`. This way, all the elements $S(k+1,k),\dots,S(n,k)$ disappear. Now we also see the reason why the row with the largest element had to be changed with the current one. If we should not have done this, elements of the rows $k+1$ to n might have been multiplied with a number greater than one, causing a possible enormous growth. Although this is not a real danger for a 4×4 matrix, for convenience I did implement it in the subroutine.
Now we have an upper diagonal matrix, with a righthandside $\underline{b}^*$ (which, of course, is also changed by the chnges explained above). In `bksub`, we finally back-substitute (as we see, names of the subroutines are not randomly chosen...) the unknowns $x_1,\dots,x_n$ to solve the original system.

3.5.4 Implementing the Inverse Eigenvalue Method

Having computed the eigenvalues and eigenvectors, we get to the most important part of the subroutine, the implementation of the Inverse Eigenvalue Method. Actually, this is done in a few lines, using the analytical algorithm from Wilkinson (see [17]):

```
      do 90 i = 1, 4
           b(i) = labdast(i)
           do 100 j = 1, 4
                do 110 k = 1, 4
                     b(i) = b(i) - eigve(i,j) * eigve(i,k) * A(j,k)
110             continue
                Jac(i,j) = eigve(i,j)**2
100        continue
           Jac(i,5) = b(i)
 90   continue
```

```
        call gauss(Jac, 4, 5, coef)
```

First, the righthandside from (2.36) is computed using the eigenvectors and λ^*. If we look at the definition given in (2.36), we see that the first element of $\underline{b}$ is (for $n = 4$)

$$b_1 = \sum_{i,j=1}^{4} a_{ij} x_{1i} x_{1j} \tag{3.28}$$

Next, the Jacobian is defined according to (2.34), and because $A_k = \underline{e}_k \underline{e}_k^T$, we have $\mathbf{J}_{ij} = x_{ij}^2$. Finally, to be able to use the Jacobian and $\underline{b}$ in gauss, they are 'mingled' into a 4×5 matrix.
This concludes the implementation of the IEM. As we have seen, it merely consists of three parts: the computing of the eigenvalues and eigenvectors, next with these values the computation of the coefficients adjusting the diagonal elements, and finally the subroutine gauss to solve a system of equations.

3.6 Changes Made with respect to Original Program

As the most important changes were made in the solver nsIEM, I will describe this solver here. Other solvers (if new) are more or less derived from this solver.
In implementing the Inverse Eigenvalue Method into the Navier-Stokes solver, a few adjustments had to be made. First, the Jacobian matrices (see the previous section) had to be derived explicitly. In [7], an analytical formulation of the convective Jacobian matrices is mentioned, which with some adjustments could be implemented. The viscous Jacobian matrices also had to be implemented to achieve the full Navier-Stokes matrix. The subroutine which served as the matrix generator is called eigen
Furthermore, in the subroutine diss a loop had to be written which makes the solver perform the IEM every nstep times, where nstep can be defined in the data file rae2822.dat. And of course the Inverse Eigenvalue Method as described before had to be implemented. This was done by calling the subroutine inveig in the subroutine eigen
The solvers used in this project finally were:

1. dantei.f, the original solver, without eigen and inveig,
2. nsIEM.f, the solver which computed the artificial dissipation that could be used by dantei1.f
3. dantei1.f, the solver which used the artificial dissipation computed by nsIEM

Other, slightly different, solvers were also used, but only to store the data in different files, so they have no real differences with nsIEM.

Chapter 4

Results

In the tests described below, a RAE2822 airfoil was used. The grid generated around this airfoil can be found in the file `rae2822.dat` and contains, apart from the grid, some parameters used in the solver. The relevant parameters, used in this research, are:

1. `ncyc`, the number of iterations,
2. `ivisc`, '0' means inviscid flow (Euler), '1' means viscous flow (N-S),
3. `vis1` and `vis2`, the second and fourth order coefficients for artificial dissipation,
4. `rin`, the Reynolds number,
5. `mach`, the Mach number,
6. `alpha`, the angle under which the flow appraoches the airfoil and
7. `labdast(i)`, the four given eigenvalues

In all the tests presented here, a subsonic (`mach` $= 0.73$), laminar (`rin` $= 5.0 \times 10^5$) flow is used, with `alpha` $= 2.73$. From here, I will refer to the case in which regular artificial dissipation was used as the 'regular case'. The case in which the artificial dissipation was computed by using the Inverse Eigenvalue Method, will be referred to as the 'new case'.

4.1 Setting Up the Test Cases

To test the new method, we first had to compute the artificial dissipation coefficients which had to be added. This was done in a partly rewritten version of the original solver, `nsIEM`. As the name indicates, this solver was obtained by implementing the Inverse Eigenvalue Method in the original solver. This way, with the local matrix and the given eigenvalues `labdast(i)`, the dissipation coefficients could be computed, as explained in the previous chapters.
This solver uses, apart from the regular coefficients which define the type of flow, four given eigenvalues to compute the dissipation coefficients with. In this first test, the following values were taken:

`labdast(1)` $= -2.0 * 10^{-2}$
`labdast(2)` $= -4.0 * 10^{-2}$
`labdast(3)` $= -6.0 * 10^{-2}$
`labdast(4)` $= -8.0 * 10^{-2}$

I chose these specific values because tests with the Inverse Eigenvalue Method as used here, showed that the convergence performance of this method reduced significantly when the chosen values were (relatively) too close to each other. This means for example that the eigenvalues can not be equal to each other, say `labdast(i)` $= -2.0 * 10^{-2}$. However, another restriction still had to be satisfied as well. The eigenvalues could not be too negative, as it would mean that too much dissipation would be added.
The solver `nsIEM` performed 1000 iterations before using the IEM subroutine and computing the dissipation coefficients. These coefficients were stored in `diss3`, so that they could be used by the solver which was to compute the solution with this new dissipation (`danteil`).
Having computed these coefficients, they are used by the solver `danteil`. Also, the regular solver, `dantei`, was used to obtain the regular solution. Both solvers were iterated 5000 times, which should be enough to get a good impression about the convergence rate.

4.2 Convergence History in the Regular and the New Case

In figure A.1, we see a comparison between the convergence history of the original method (dashed line) and the method which uses the coefficients computed by the IEM (solid line). From this comparison, we see that the method which uses the regular artificial dissipation converges smoothly. The plotted function here, ${}^{10}\log\Delta\rho$, is almost linear, with the small 'bumps' caused by the imaginary part of the eigenvalues. The method using the artificial dissipation computed from the IEM shows different behaviour. Until about 700 iterations, it keeps up with the original method quite well. From there, however, it maintains the level of about 10^{-4}, which indicates that it does not converge. And although it does not really diverge either, it shows us that the regular method performs better than the new method. The reason this method maintains this level and will not converge further, will be explained later. Let us first consider the differences between the two methods.

4.3 Comparison Between Old and New Method

Of course, a more detailed look at these two methods is desired. To get an impression of the differences, three main criteria were considered. First, the *added artificial dissipation* over the field is shown. Furthermore, I will show $\Delta\rho$ *over the flow field*, to see the relationship between possible differences in added dissipation and convergence and stability. Finally, the *pressure distribution* is shown, both over the field and around the airfoil. This way, we can see whether the solution obtained with the new method is really 'less dissipative' then the regular method. Hopefully, a better estimate of the real solution will be given with a less dissipative method, which was the ultimate goal of this research after all.

4.3.1 Comparing Added Dissipation and $\Delta\rho$ in the Two Cases

The first thing we are interested in is to see the difference in the amount of added dissipation between the two methods, and to see where in the field this difference is largest. If we compare this with $\Delta\rho$ over the flow field, we can find out where the influence of the added dissipation plays an important role in the convergence and the stabilization of the method. Figure A.3 shows the amount of dissipation after 1000 iterations, where the Inverse Eigenvalue Method is used to compute the dissipation coefficients. We can distinguish three important regions where artificial dissipation is added, I will refer to them as regions A, B and C (see figure 4.1).
Most dissipation is added in the regions where large gradients might occur, A and B. Al-

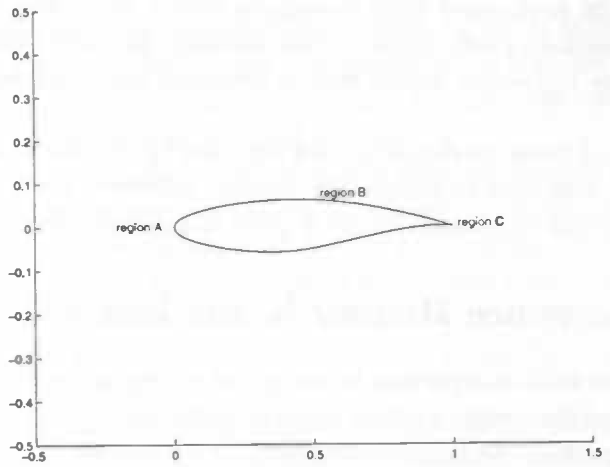


Figure 4.1: The three important regions near the airfoil

though in this case we do not really have a shock in region B, as will be shown later, this is obviously some 'reaction' of the solver to the fact that there is a *tendency* of a shock. Near the trailing edge of the airfoil, also some dissipation is added. It is important to note that this latter dissipation is mainly of fourth order, i.e. to damp out oscillations caused by the numerical scheme. The dissipation in the regions A and B is of second order, to damp out oscillations due to large gradients. It is exactly this fourth order dissipation that we tried to optimize here, so the difference in added dissipation can easily be explained. We have to see whether this optimization really has a positive effect, which will be done later. If we take a look at the added dissipation computed by the regular method as shown in figure A.2, we see some differences. The most important difference is that in this case, a lot more dissipation is added near the trailing edge of the airfoil. The amount of dissipation added in the other two regions is more or less the same. It is important to note that the artificial dissipation added in region C is mainly of fourth order, i.e. to damp out oscillations caused by the domination of the convective terms. The dissipation in the regions A and B is of second order, to damp out oscillations due to large gradients. It is precisely the fourth order dissipation that we tried to optimize here, which means of course that less dissipation is added in the new case. This easily explains the difference in added dissipation. We still have to see, however, whether this optimization really has

a positive effect, which will be done later.
The influence of the 'extra' dissipation in region C on the convergence of the solver can be shown if we compare $\Delta\rho$ over the flow field after 1000 iterations in both cases (figures A.4 and A.5).
The first difference is that the graph in the 'regular' case looks much smoother overall than the graph in the case IEM was applied. But what matters more is the fact that at the trailing edge of the airfoil, we see that $\Delta\rho$ in the new case is much larger than $\Delta\rho$ in the regular case. From this, we might conclude that the fact that too little dissipation is added in the new case, causes this method to show irregularities in that region. How we expect this relationship to be, will be explained later.
If we look at the graphs of the added dissipation after 3000 iterations in both cases, we see some interesting differences and similarities. First, we note that the amount of dissipation in the regular case (figure A.6) has not changed significantly. The dissipation above the airfoil (region B) has shifted a little towards the front end, which can be explained by the fact that in the course of 'time', the shock is also moving slowly towards the front end of the wing.
Secondly, if we look at the added dissipation in the new case (figure A.7), we see that the added dissipation in region B has been decreased significantly. The dissipation in region A is about the same, while the added dissipation in region C has been somewhat changed. The effects these changes have on the convergence can be seen when we look at the graphs of $\Delta\rho$ in the field for the two cases.
In figure A.8, we see an overall smooth graph, with a clear peak at region B, although it is important to note that the order of magnitude here is 10^{-5}, while after 1000 iterations it was of order 10^{-4}. So the method does converge (which we saw from the convergence history already, of course), but is still having some trouble with the 'shock wave'.
If we consider the graph of $\Delta\rho$ in the new case, figure A.9, we immediately see where the trouble occurs. In most regions, we see some relatively small irregularities. In region C however, a peak is visible and between regions B and C, we also see a lot of extra wiggles. It is quite clear that these irregularities cause the method not to converge, but at this stage it is still unclear how and why this peak occurs. In the section on the velocity field, more is explained about this matter.
Figures A.10 and A.11 finally, show us the amount of dissipation added after 5000 iterations. In the regular case, nothing much has changed. Still most dissipation is added at the trailing edge of the airfoil. In regions A and B, we also see that about the same amount of dissipation has been added as before. The amount of dissipation in the new case has again changed somewhat, particularly at the trailing edge, where a little extra dissipation has been added. The effects of this extra dissipation is visible if we consider $\Delta\rho$ in the field.
In figure A.12, we see $\Delta\rho$ over the flow field. Note that the order of magnitude has already reduced to 10^{-6}, which corresponds with the convergence history as shown in figure A.1. This is also the explanation for the wiggles in the flow field. These are caused by the fact that the machine accuracy is around 7 figures, 'enough' to cause irregularities on this level. If we do not take these wiggles into account, nothing much has changed compared to the situation after 3000 iterations. Again, the largest $\Delta\rho$ occurs in region B.
Figure A.13 shows a much more different result. Region C is still very unstable, the order of magnitude being even 10^{-3}. The impression is that these peaks are more or less ran-

domly 'wiggling' around 10^{-3}, as the plot of the convergence history in max $\Delta\rho$ shows an irregular course as well, but it is more or less constant at this level.
In region B we still see a large bump and some smaller ones, and in region A the peak appears to be somewhat larger than in the previous case. Finally, we see one larger and two smaller bumps at the lower side of the airfoil. Our main interest, however, is focused on the trailing edge of the airfoil, region C, because the most relevant irregularities were found there.
Let us first check what the pressure profiles in the flow field and around the airfoil look like.

4.3.2 Comparing the Resulting Pressure Distribution

Above, the relationship between the addition of artificial dissipation and convergence of the two methods was shown. What we are now interested in is to see whether the solution obtained after 5000 iterations from the new method is really less dissipative and whether it gives a solution which is closer to the real solution.
Let us first take a look at the distribution of the pressure around the airfoil. In figure A.14 and A.15, we see the pressure distribution around the airfoil in the regular and the new case, respectively. A few conclusions can be made from these figures. First, it is obvious that the drag has been greatly reduced in the new case. There is not much of a shock visible, and we see that already at $x = 0.3$, the amount of pressure above the airfoil approaches the amount of pressure below the airfoil. It should be clear that this has a bad influence on the drag capacities of the wing.
In figures A.16 and A.17, an impression is given on how the behaviour of the pressure is in the flow field around the airfoil. We recognize the pressure distribution around the airfoil as shown above.
In the regular case, the shock is sharper. Until $x = 0.5$, the level of the pressure above the airfoil is significantly lower than that below the airfoil. Thus, we may conclude that the regular solution seems to be a better estimate of the real case than the 'new' solution.
The question whether the new solution looks 'less dissipative' is harder to answer. The first difficulty is that we do not have a proper definition of what 'less dissipative' means. Furthermore, we have to distinguish between a solution which is less dissipative, and a solution which approaches the desired solution best, the first does not automatically imply the second.
We could just say that a solution looks less dissipative if it shows more wiggles and other irregularities than in a case in which more dissipation is added. In that case, the new solution considered here is indeed less dissipative than the regular solution. But this still does not account for the fact that the shock is less apparent in the new case. If in a method less dissipation has been added, we expect this method to have less dissipative behaviour, of course. And the fact that the shock tends to disappear is precisely the opposite behaviour. Thus, we might need the information given by the velocity field as well.

4.3.3 The Velocity Field

Having seen the relationships between the added dissipation, the convergence and the resulting pressure distribution, we suspect that the problems occur between the trailing

edge of the airfoil (region C) and the region where the shock may be expected (region B). Comparing this region in both cases shows a very interesting phenomenon.
In the regular case, nothing much happens. The velocity field above and below the airfoil is laminar and does not show any irregularities. In the new case, however, we see a region of separation at the trailing edge of the airfoil, between regions B and C. And although this seems to be quite a normal flow, it is not a desirable effect. What exactly happens here? First of all, because of the separation, the flow becomes unstable and tends to become turbulent, which means that we have an unsteady, or time dependant, solution. And that is related to the following problems:

1. There is no turbulence model built in the Navier-Stokes solver. This means, of course, that turbulent phenomena cannot be captured in the right way.
2. More or less related to the first point: this solver cannot solve unsteady flows, as it uses time to iterate towards a steady state solution.

Let us focus on the second problem. The solver assumes an eventual steady solution: 'in the course of time' the solution will converge. In this case, however, we do not have a steady state solution. This means that the solution cannot converge further than the amplitude of the (time dependant) solution, which is about 10^{-4}, as can be seen from figure A.1.
Moreover, we would like to know how this unsteady solution can occur. Because of the 'lack' of dissipation, the flow is under more influence of convective terms (i.e., transport caused by the flow itself) than of viscous terms (transport caused by differences in concentration). Generally speaking, the dissipative terms are of the same kind as viscous terms, i.e. they increase the viscosity of the fluid. So, for example, it means that highly dissipative flows will not easily show separation phenomena. In this case, however, there was too little dissipation, which means that the flow was able to show a region of separation. Another consequence is that the boundary layer near the trailing edge becomes much thicker, thus significantly influencing the effective airfoil shape. A result is that the shock moves forward and becomes much weaker.

4.4 A New Test Case

If our expectation is true, we should encounter less difficulties if the added dissipation is increased a little. To achieve this, the given eigenvalues for the Inverse Eigenvalue Method were set to:

`labdast(1)` $= -6.0 * 10^{-2}$,
`labdast(2)` $= -1.2 * 10^{-1}$,
`labdast(3)` $= -1.8 * 10^{-1}$,
`labdast(4)` $= -2.4 * 10^{-1}$.

With these 'more negative' eigenvalues, we may expect that the added dissipation will increase. As explained before, an increase in the addition of artificial dissipation causes the eigenvalues to become more negative, so it can be expected that if the given eigenvalues are more negative, there will be more added dissipation.

The result of this 'adapted' new method is shown in figure A.20. As we can see, the method with coefficients obtained from this 'new IEM' converges better than the previous method. From this we can already draw the conclusion that probably less disturbances occur than in the previous case. The pressure distribution around the airfoil with this adapted new method is shown in figure A.21. Not surprisingly, this solution looks much better than the first one with the IEM. The difference between the pressure distributions of the regular method and the adapted new method is barely visible.
If we compare the velocity fields from the two methods, there is again only a little difference. Figure A.22 shows the velocity field in the case the 'new IEM' was applied. At the very end of the wing we see a little more 'hesitation' in this case than in the regular case. And in the boundary layer, the difference is also visible somewhat towards the front end, but generally, the performance here is much better than with the previous IEM.
Finally, we can consider the graphs of the added dissipation and $\Delta\rho$. In figures A.23 and A.24, a perfect match with our theory appears. More dissipation at the trailing edge, no real wiggles for $\Delta\rho$, so we may conclude that the Inverse Eigenvalue Method with these prescribed eigenvalues gives satisfying results.

4.5 Using a Blending of Both Methods

Although the first Inverse Eigenvalue Method did not give satisfying results near the trailing edge, the amount of added dissipation was enough elsewhere. This brought us to the idea of 'blending' the Inverse Eigenvalue Method (with the coefficients as used in the first case) and the original method. Near the trailing edge, the original method was used and in the other regions, I used the Inverse Eigenvalue Method with the eigenvalues from the first case.
Let us first take a look at the convergence history of this last method compared to the convergence history of the original solver. Figure A.25 shows this comparison, with the solid line giving the convergence rate of the mixed method and the dash dotted line that of the original method. From this graph, we may conclude that the solver using the 'mix' of both methods performs only a little less than the original method. As we see, both lines run almost equal until about 3000 iterations. From there, the mixed method is converging slightly less rapid, but still almost linearly decreasing, until it reaches a level of about 10^{-6}. This can be explained by the fact that the solution might still be a little unsteady, so that on this level oscillations occur. Later we will see that this assumption is justified by the velocity field.
Although this is an encouraging result, we have to check whether the results concerning the amount of added dissipation and the resulting pressure distribution also satisfy our expectations. First, let us take a look at the amount of dissipation added in the flow field. If our expectations are satisfied, we should see an amount of artificial dissipation near the trailing edge similar to that in the original case. In other regions, we should see a situation similar to what we saw in the case the first IEM was applied. If we compare the graph of added dissipation after 5000 iterations (figure A.26) with figures A.10 and A.11, we see that we have indeed obtained a mixture of these two cases. The wiggle in region B is caused by the fact that a sharper shock occurs in this method than in the method in which only the IEM was applied. This sharper shock causes the second order dissipation

to increase.
The effect this change in addition of artificial dissipation has on the convergence of the method is visible in the graph of the average $\Delta\rho$ over the flow field (figure A.27). After 5000 iterations, this has been reduced to order 10^{-6}, which is much better than the first method with the IEM, and about the same as the original method. Of course this could already be concluded from the convergence history plot.
If we now compare the pressure distribution around the airfoil (figure A.28), we see that there are almost no differences between the solution obtained from the original method and the solution from this new method. Differences between these two methods are too small to be significant.
Finally, the velocity field near the trailing edge can tell us something more on how the solution behaves in difficult regions. In figure A.29, we see how this part of the solution behaves. Although the situation is again slightly different in this case compared to the original case, there is no significant change here as well. The differences occurring between the original and the present case are caused by the fact that less dissipation is added in region B, which causes the velocity to increase in this region. The pressure becomes slightly lower in this region, making the flow more prone to separation. This is to be seen in the same figure, where a beginning of a region of separation is visible.
We may conclude from these results that the last method, of blending the original solver and the IEM, gives the most satisfying results. The obtained solution shows irregularities when too little dissipation is added. These irregularities are caused by normal physical phenomena, and they tell us that we have to add more artificial dissipation. In regions with smooth graphs however, like the region below the airfoil, it should be possible to add even less artificial dissipation. The pressure graph shows a smooth line here, which means that enough dissipation is added.
Furthermore, this method combines a more 'subtle' addition of artificial dissipation (i.e. *less* artificial dissipation) with an apparent consistency. This means that the method seems to depend not very much on a variation of the prescribed eigenvalues.
Nevertheless, we may question whether the results are satisfying enough to consider applying this blending method in future solvers. In computing speed e.g., no significant difference was to be seen. And, after all, we were looking for a 'less dissipative' result. Although it is hard to define whether a solution is less dissipative, the solutions from both methods seem to be similar. This indicates that the results of the adaptions on the solver are marginal. More on this discussion is given in the next chapter.

Chapter 5

Conclusions

We saw that if too little dissipation is added, i.e. the given eigenvalues are too close to zero, the solution is unsteady. This implies that the numerical solution does not converge. And although the velocity field does not look very unusual, this solution is too less dissipative to be acceptable.
If the artificial dissipation is increased a little, the solution converges much better. And comparing the amounts of dissipation between the original method and this new method, we clearly see a decrease in the addition of artificial dissipation when this new method is applied. This is a hopeful result, but on the other hand one might question if the method is consistent enough to be a real helpful tool in future cases. In other words, for a slight change of the prescribed eigenvalues, will the solution change in about the same order? And if not, from what point is its behaviour not acceptable anymore?
In trying to improve the consistency of the method, another case was tested. The idea was to use the original amount of dissipation near the trailing edge (region C), thus assuring a good convergence behaviour. In the other regions, we assumed that the amount of artificial dissipation computed by the Inverse Eigenvalue Method should be enough to stabilize the numerical method. This turned out to work quite well, in the sense that the velocity and the pressure graphs looked much like the original situation. And according to the graphs showing the amounts of added dissipation, we add less artificial dissipation in this case compared to the original case. In short, we may conclude that this last method shows promising results.
Another question that comes to mind is whether the 'mixed' solver is really faster? Of course there is something won by the fact that no computations for the fourth order dissipation need to be done anymore in regions where the IEM is applied. Performing 5000 iterations in both cases, however, did not show many differences in CPU-time. It might be interesting to see whether it is possible to improve this part of the solver as well.
For future research there are some points of attention I would like to make here:

- Some research could still be done on the last method to find out whether this method is really consistent enough to deal with (small) changes in the given eigenvalues. If so, the results on convergence, pressure distribution and the velocity field should differ only slightly when the eigenvalues are changed a little.

- There are no results available yet for turbulent and/or supersonic flows, as this solver did not have a turbulence model. It could be interesting to see how the Inverse

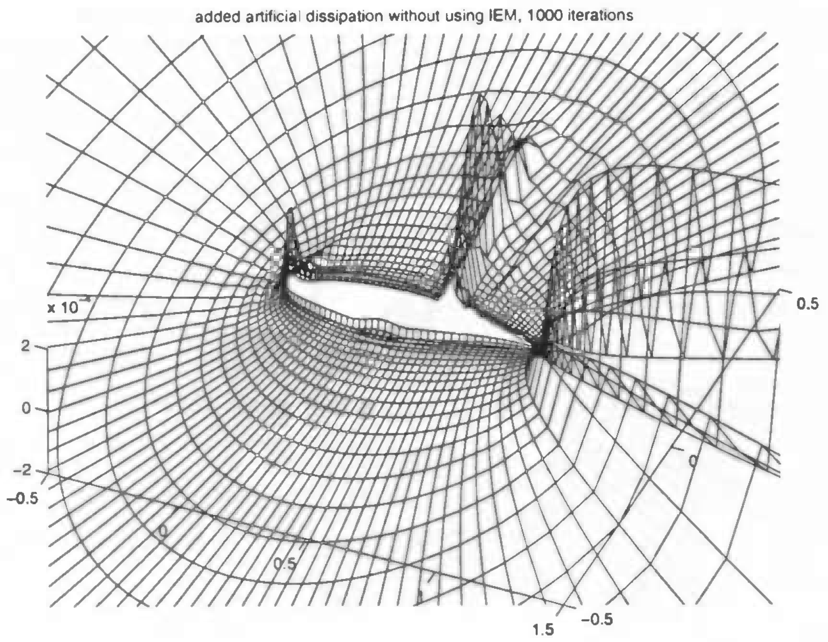


Figure A.2: Added artificial dissipation computed by the regular method, in the flow field after 1000 iterations

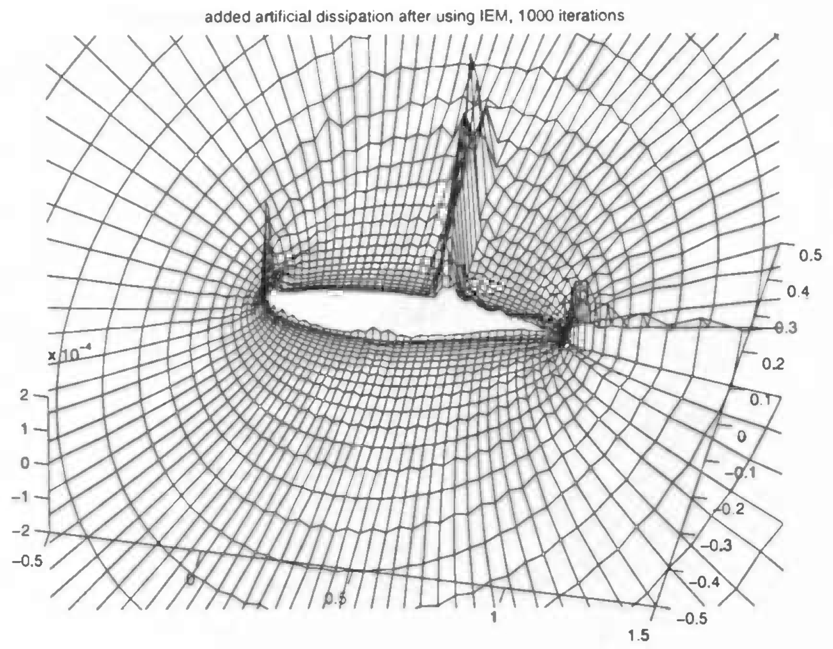


Figure A.3: Added artificial dissipation computed by the IEM, in the flow field after 1000 iterations

Appendix A

Figures Obtained by the Various Solvers

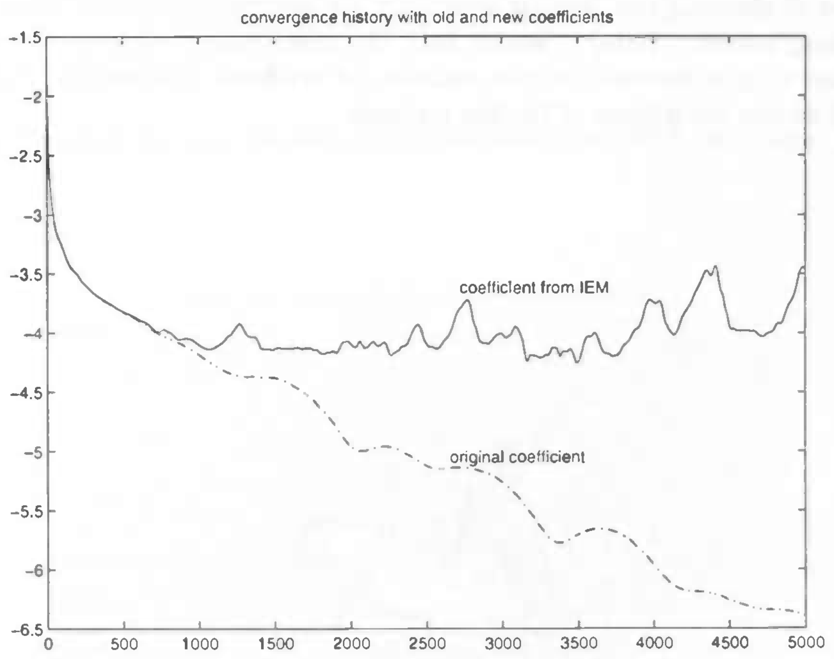


Figure A.1: Comparison between average $\Delta\rho$ from old method (dashed line) and new method (solid line)

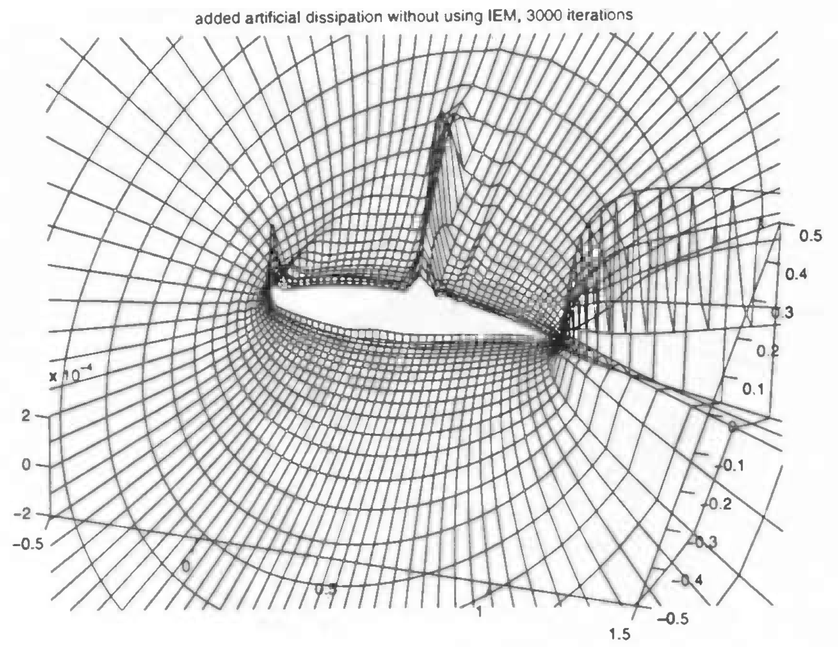


Figure A.6: Added artificial dissipation computed by the regular method, in the flow field after 3000 iterations

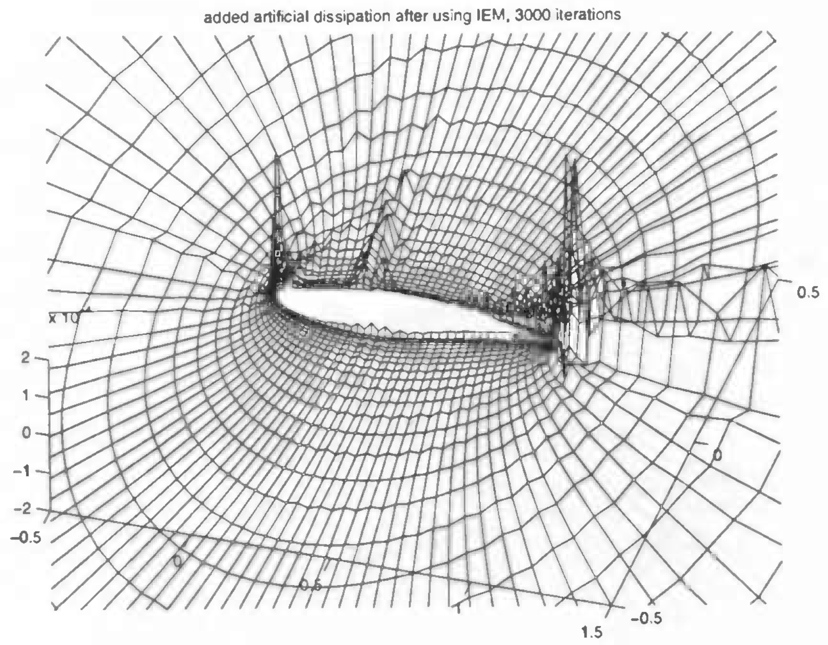


Figure A.7: Added artificial dissipation computed by the IEM, in the flow field after 3000 iterations

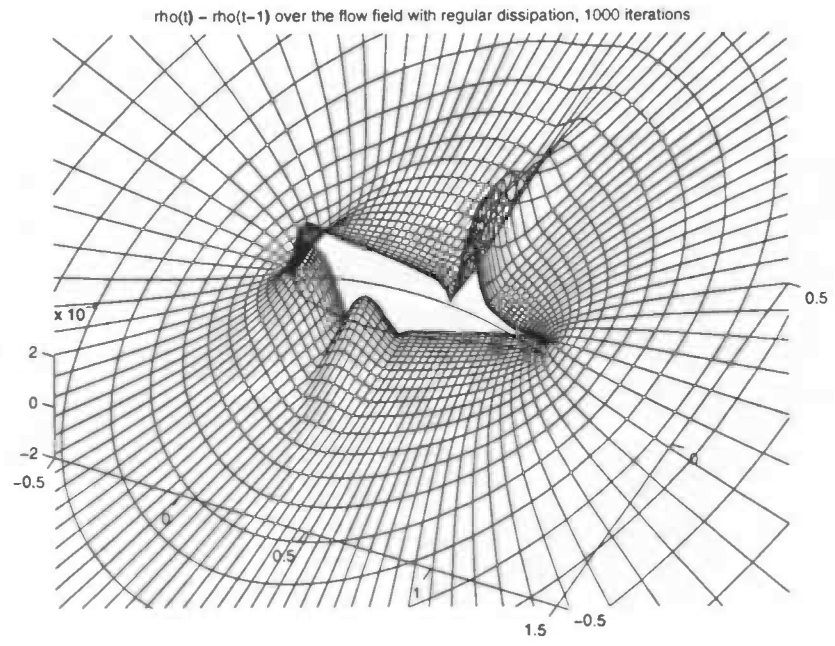


Figure A.4: Graph of $\Delta\rho$ over the flow field, computed after 1000 iterations, with regular dissipation

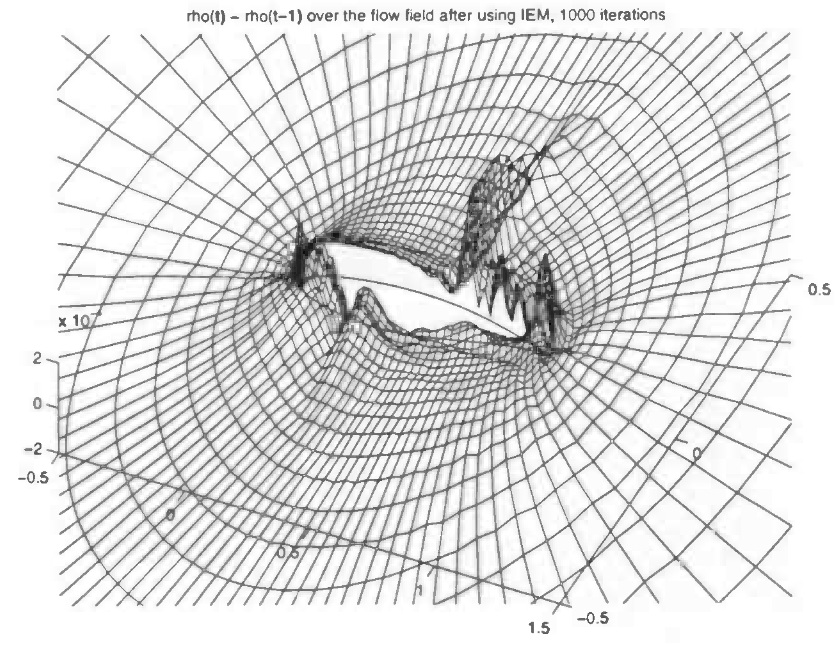


Figure A.5: Graph of $\Delta\rho$ over the flow field, computed after 1000 iterations, in the case IEM is applied

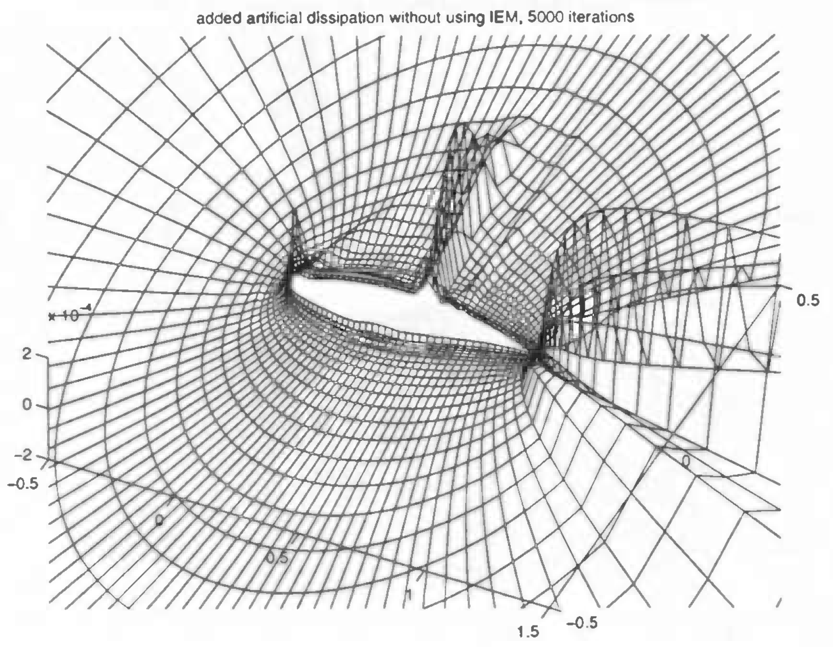


Figure A.10: Added artificial dissipation computed by the regular method, in the flow field after 5000 iterations

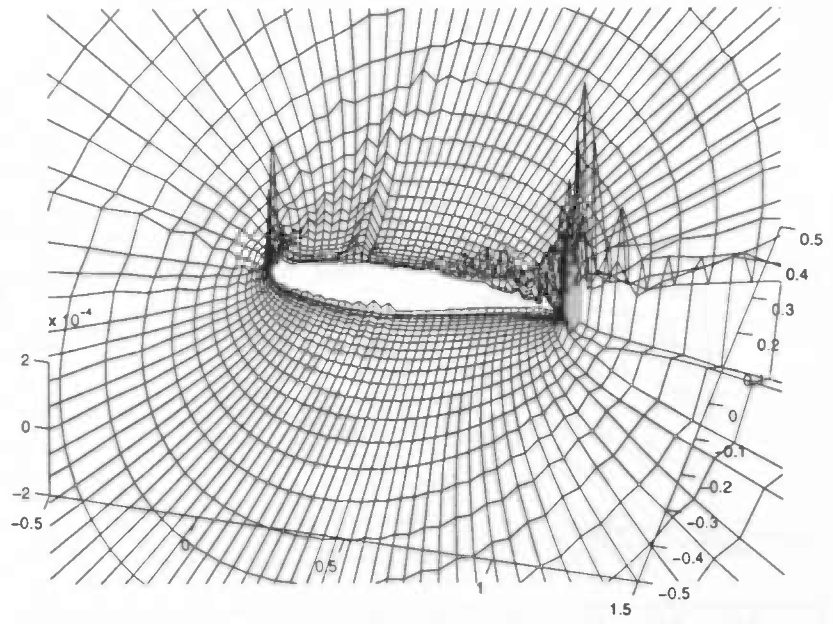


Figure A.11: Added artificial dissipation computed by the IEM, in the flow field after 5000 iterations

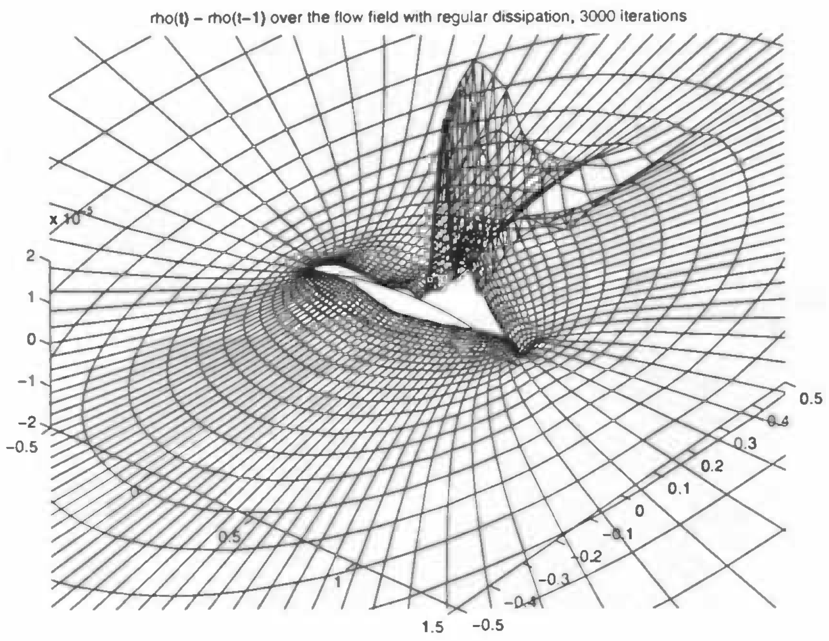


Figure A.8: Graph of $\Delta\rho$ over the flow field, computed after 3000 iterations, with regular dissipation

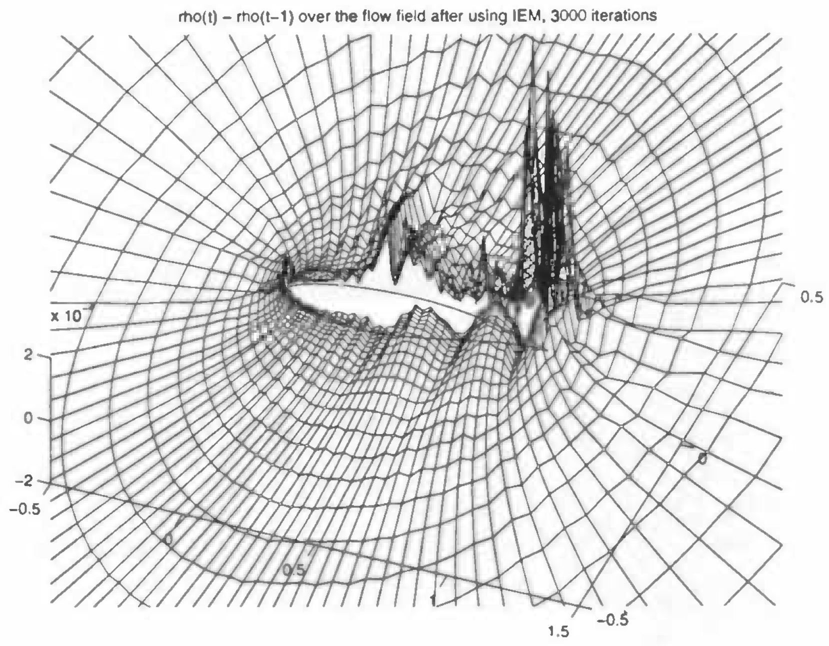


Figure A.9: Graph of $\Delta\rho$ over the flow field, computed after 3000 iterations, in the case IEM is applied

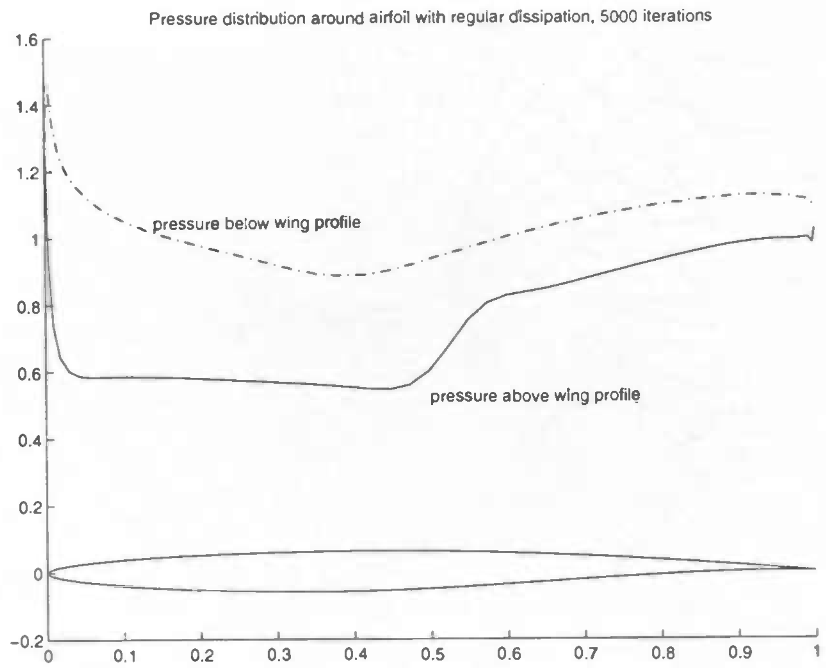


Figure A.14: Pressure distribution around an RAE2822 airfoil after 5000 iterations, with regular artificial dissipation

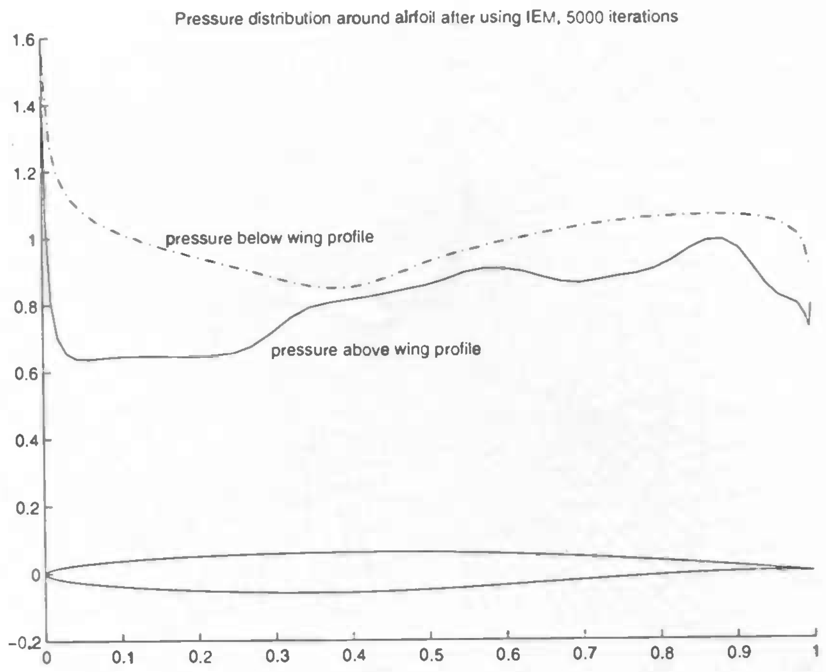


Figure A.15: Pressure distribution around an RAE2822 airfoil after 5000 iterations, with artificial dissipation computed by the IEM

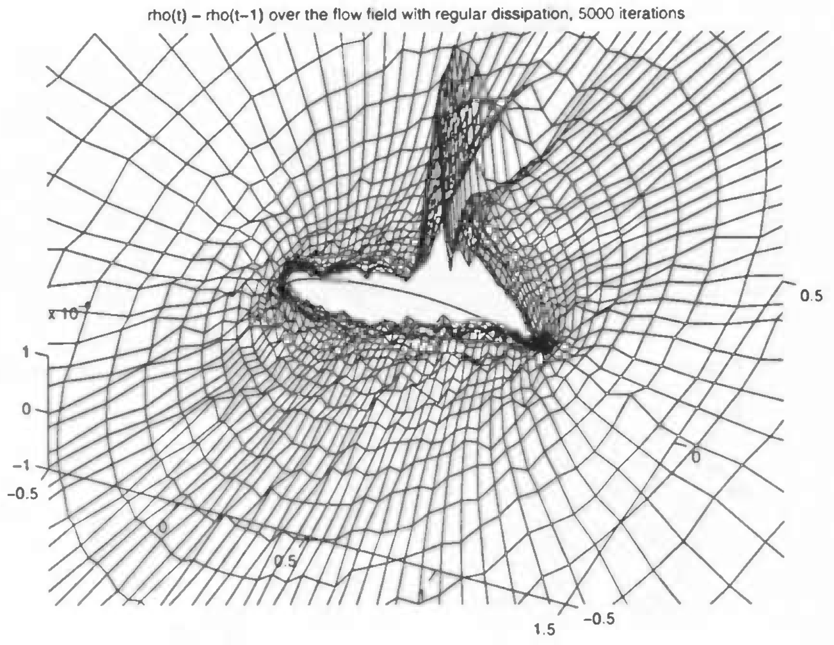


Figure A.12: Graph of $\Delta\rho$ over the flow field, computed after 5000 iterations, with regular dissipation

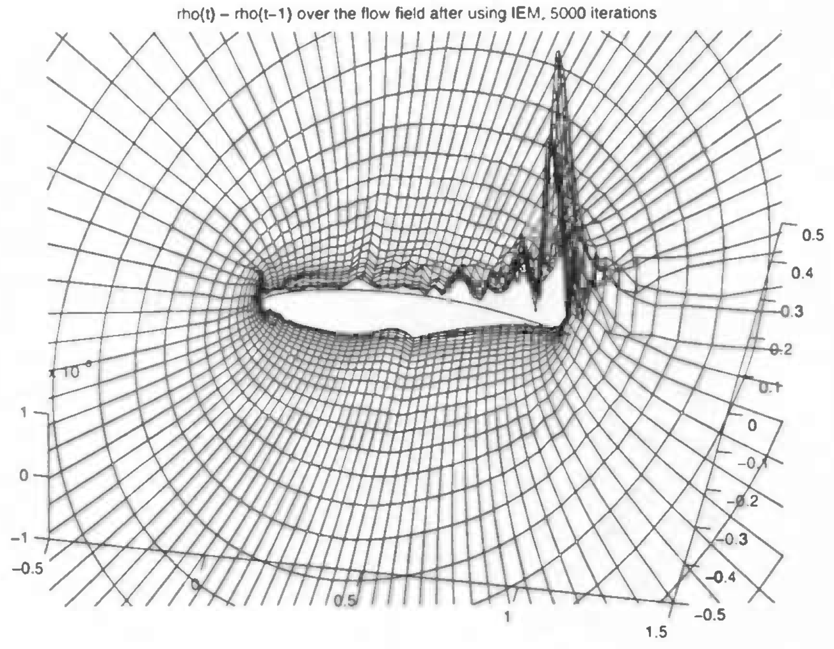


Figure A.13: Graph of $\Delta\rho$ over the flow field, computed after 5000 iterations, in the case IEM is applied

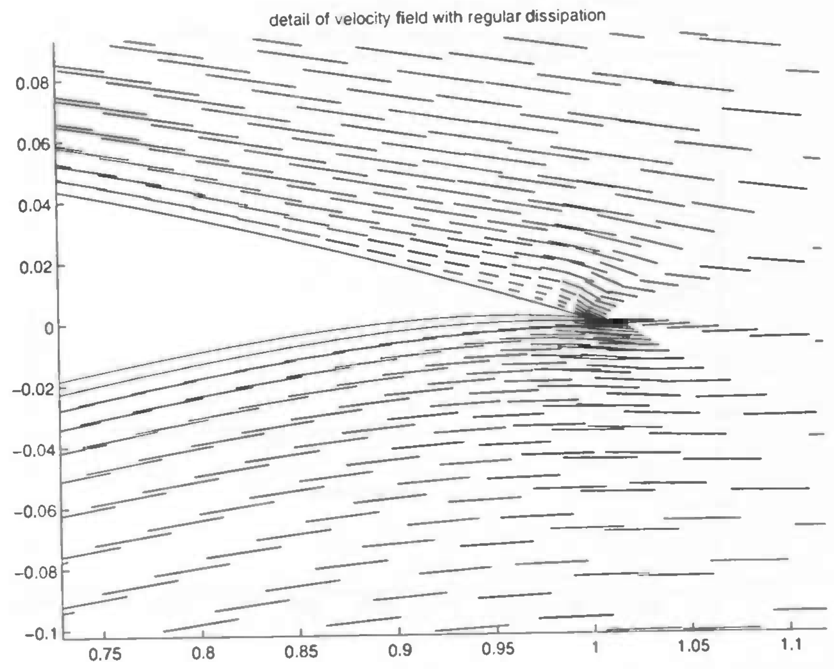


Figure A.18: Velocity field around the trailing edge, computed with regular dissipation

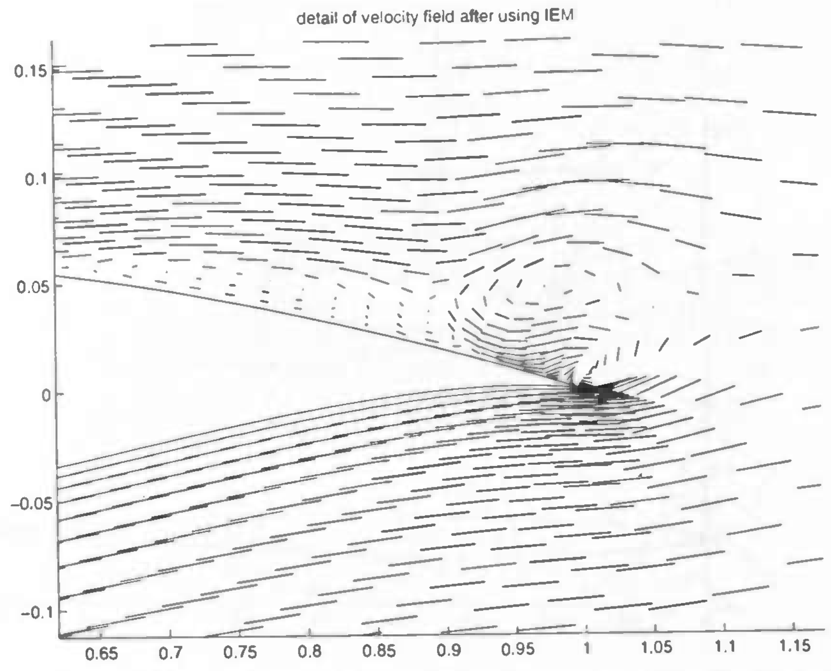


Figure A.19: Velocity field around the trailing edge, computed with dissipation from IEM

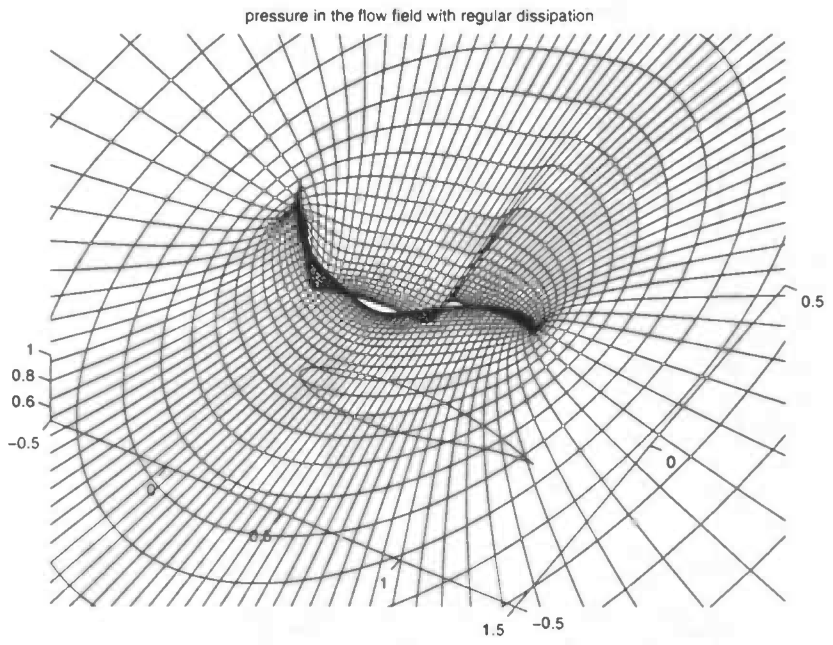


Figure A.16: Pressure distribution in the flow field after 5000 iterations, with regular artificial dissipation

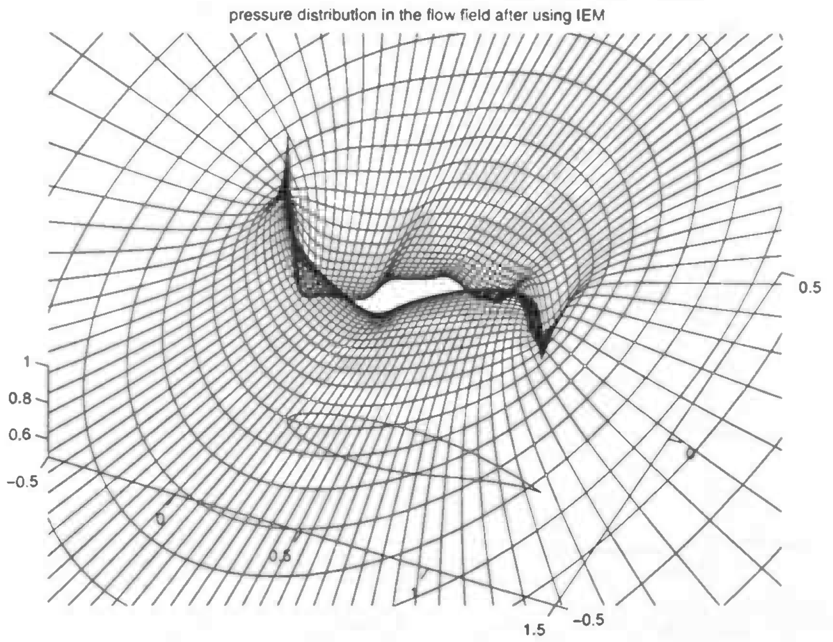


Figure A.17: Pressure distribution in the flow field after 5000 iterations, with artificial dissipation computed by the IEM

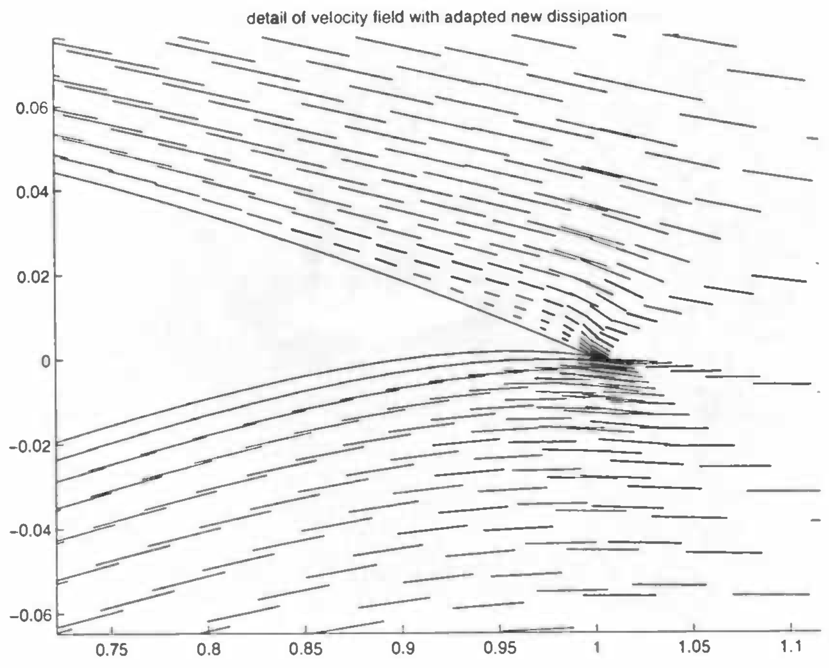


Figure A.22: Velocity field around the trailing edge, computed with adapted dissipation coefficients

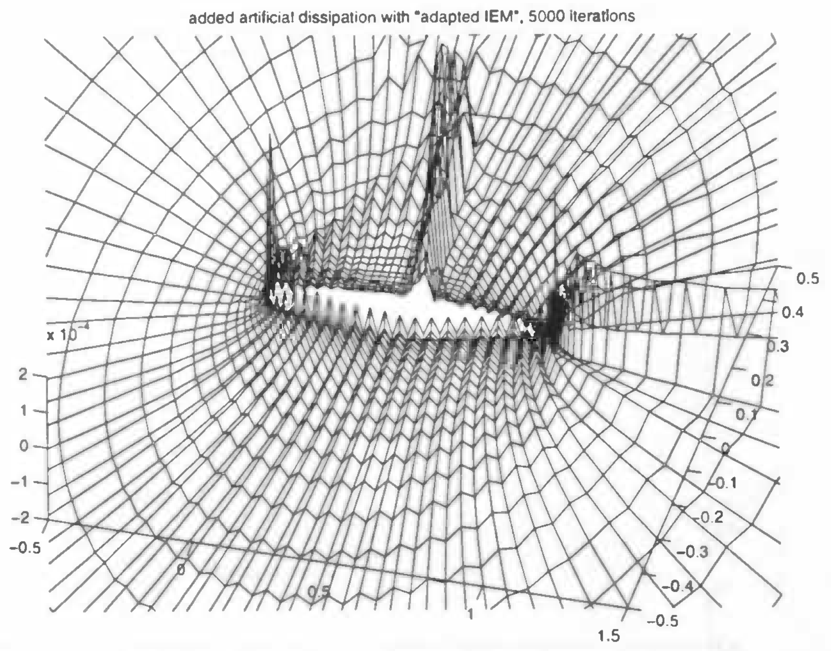


Figure A.23: Graph of the added artificial dissipation computed after 5000 iterations, with adapted dissipation coefficients

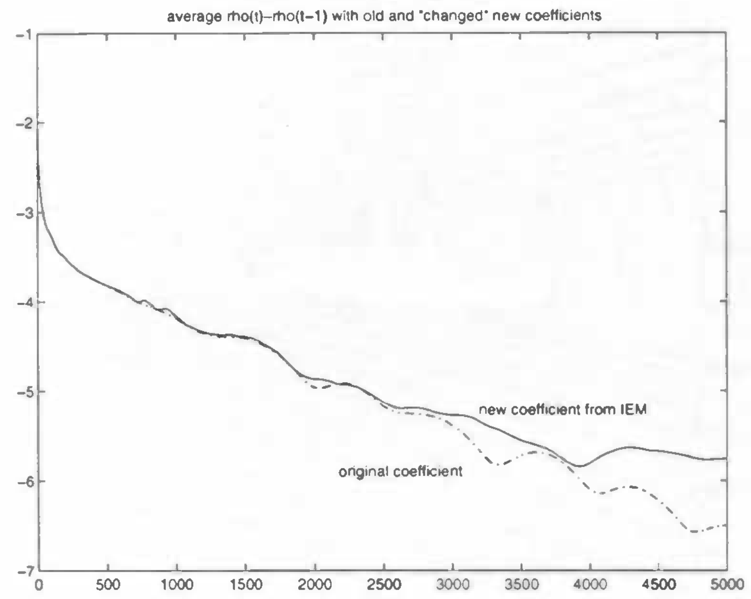


Figure A.20: Convergence history of the regular solver and the 'adapted' new solver

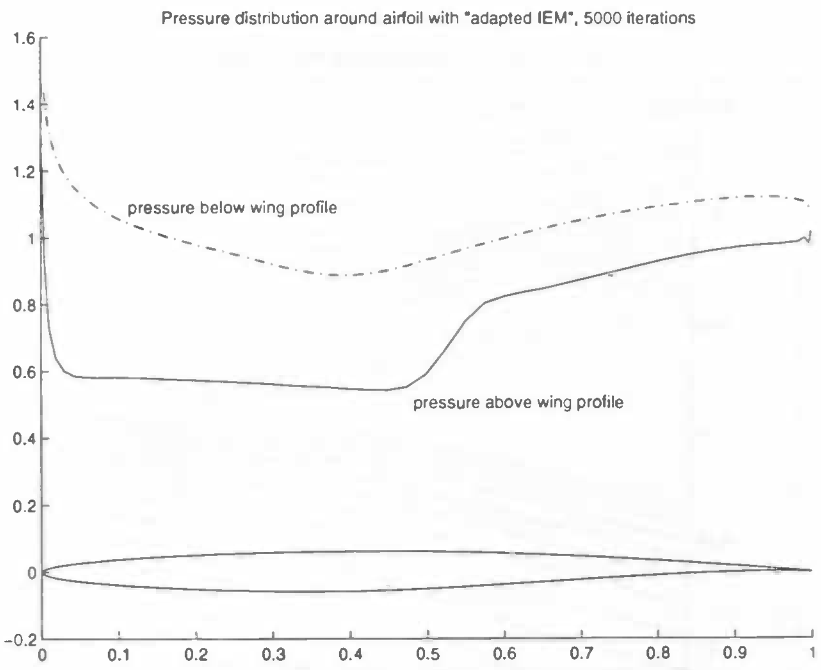


Figure A.21: Pressure distribution around the airfoil with the dissipation computed by the adapted new method

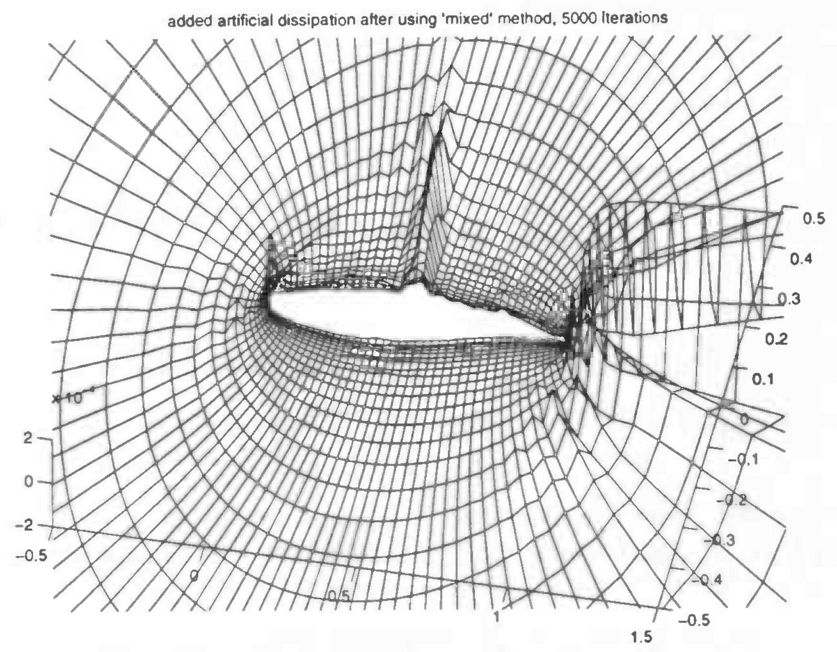


Figure A.26: Graph of the added artificial dissipation computed by the mixed method, after 5000 iterations

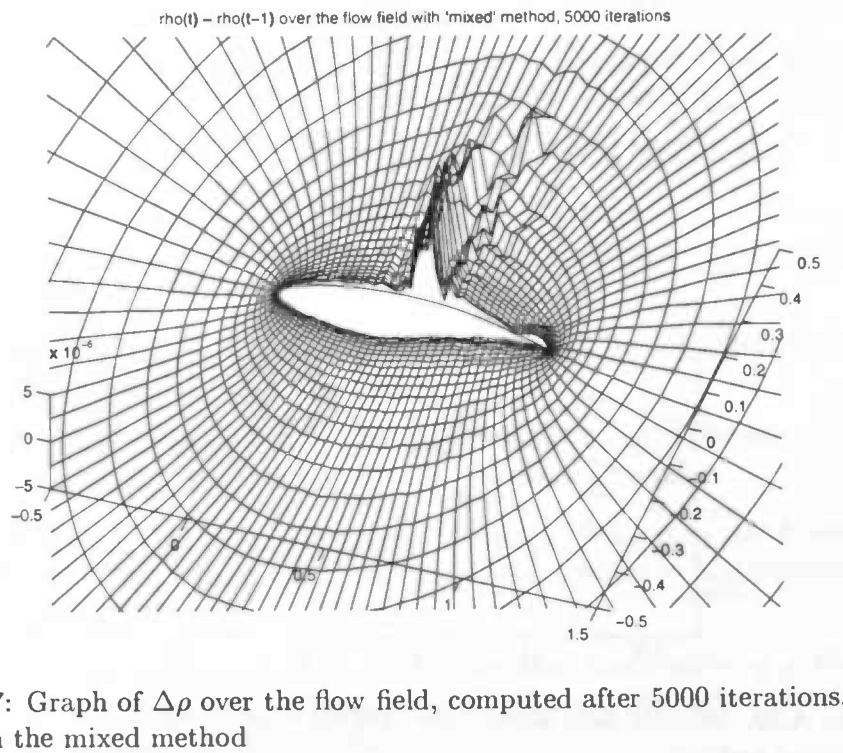


Figure A.27: Graph of $\Delta\rho$ over the flow field, computed after 5000 iterations, with dissipation from the mixed method

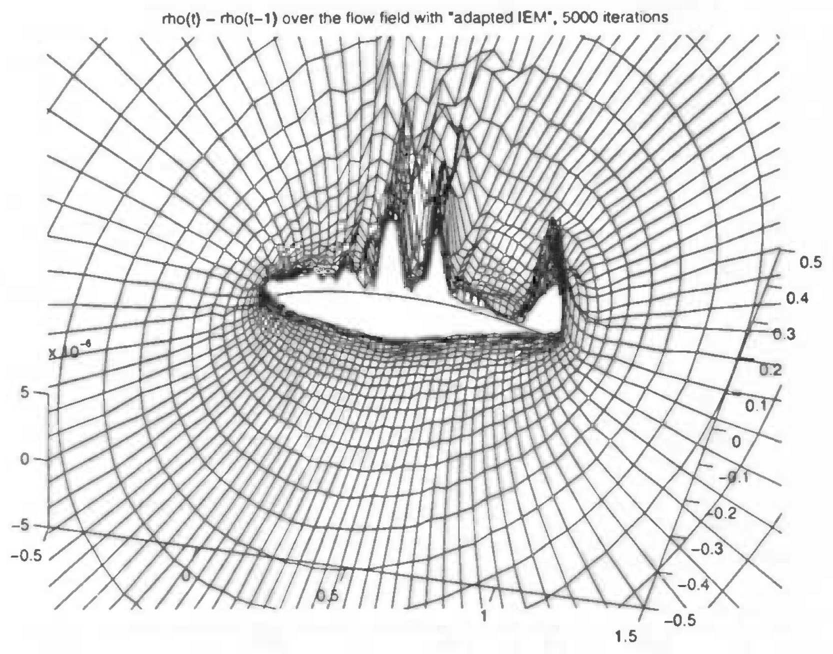


Figure A.24: Graph of $\Delta\rho$ over the flow field, computed after 5000 iterations, with adapted dissipation coefficients

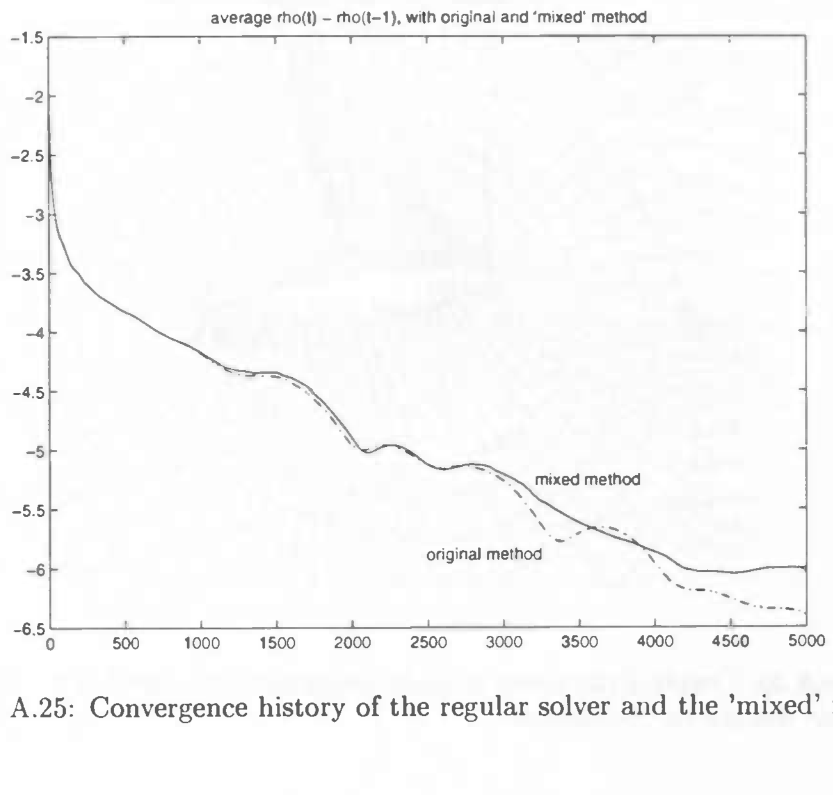


Figure A.25: Convergence history of the regular solver and the 'mixed' method

Bibliography

[1] E. Alimin, *Ph.D. Thesis*

[2] V. Barcilon, *Inverse Eigenvalue Problems*, Lecture Notes in Mathematics Vol. **1225**, Springer-Verlag, 1986

[3] K.A. Cliffe, T.J. Garratt and A. Spence, *Eigenvalues of Block Matrices Arising from Problems in Fluid Mechanics*, SIAM J. of Matr. Anal. Appl. **15** no. 4 pp. 1310-1318 (1994)

[4] Dantje K. Natakusumah, *Ph.D. Thesis*

[5] Shmuel Friedland, *Inverse Eigenvalue Problems*, Linear Algebra and its Applications **17**, 15-51 (1977)

[6] S. Friedland, J. Nocedal and M.L. Overton, *The Formulation of Numerical Methods for Inverse Eigenvalue Problems*, SIAM J. of Num. Anal. **24** no. 3, pp. 634-667 (1987)

[7] C. Hirsch, *Numerical Computation of Internal and External Flows*, Vol. 2: Computational methods for inviscid and viscous flows, Wiley, 1990

[8] H.W. Hoogstraten, *Stromingsleer*, Rijksuniversiteit Groningen (1992)

[9] A. Jameson, W. Schmidt and E. Turkel, *Numerical Solutions of the Euler Equations by a Finite Volume Method using Runge-Kutta Time-Stepping Schemes*, AIAA Paper 81-1259, 1981

[10] A. Jameson, *Transonic Flow Calculations for Aircraft*, Lecture Notes in Mathematics Vol. **1127**, Springer-Verlag, 1983

[11] Harvard Lomax, Thomas H. Pulliam and David W. Zingg, *Fundamentals of Computational Fluid Dynamics*, September 1996, Internet document

[12] M.R. Osborne, *On the Inverse Eigenvalue Problem for Matrices and Related Problems for Difference and Differential Equations*, Lecture Notes in Mathematics Vol. **228**, Springer-Verlag 1971

[13] Luis C. Santos, *Conceptual Ideas for Eigensystem Acceleration of a Finite Volume Scheme for the Navier-Stokes Equations*, Aerothermal Methods and Analysis, Rolls Royce Inc., Atlanta GA, 30339-3769, 1994

[14] Luis C. Santos, paper, 1993

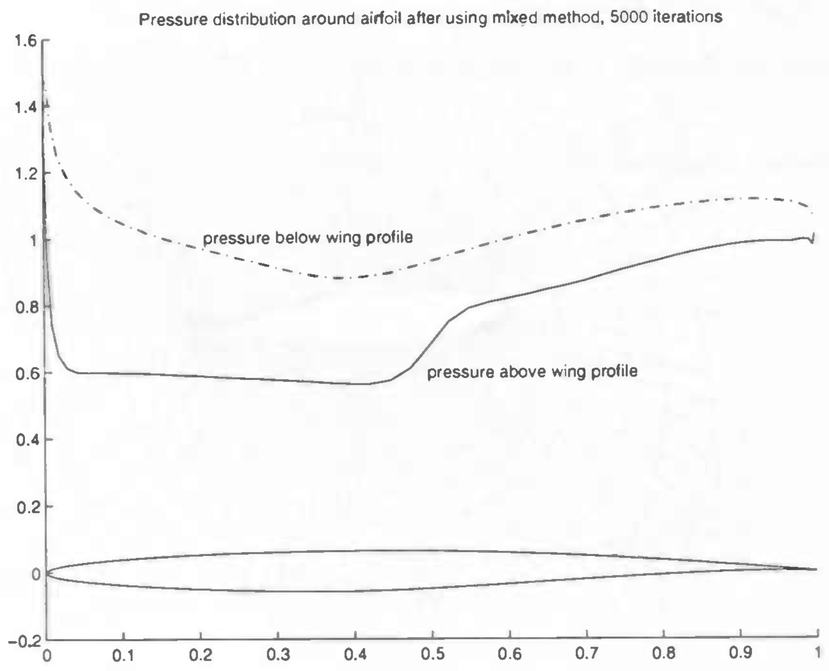


Figure A.28: Pressure distribution around the airfoil with the dissipation computed by the mixed method

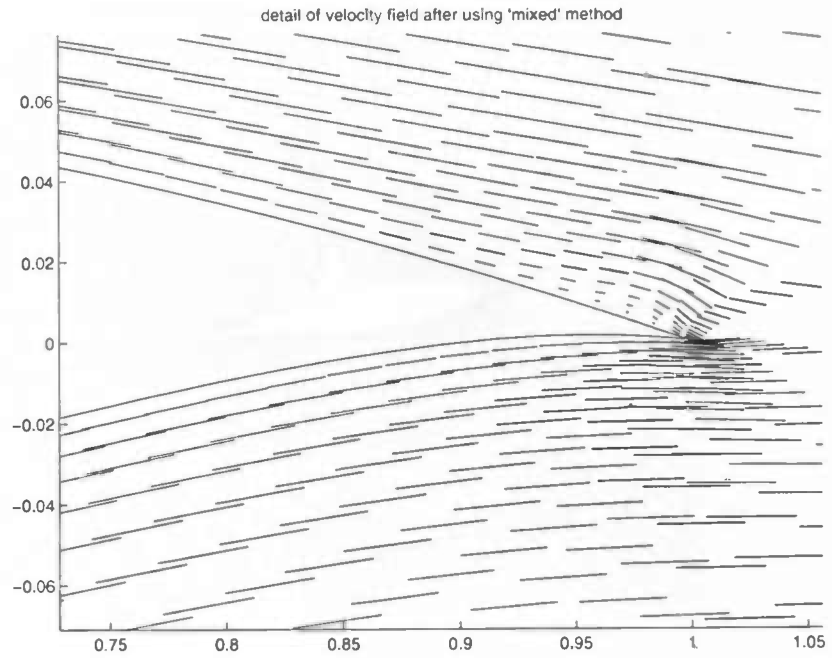


Figure A.29: Velocity field around the trailing edge, computed with the dissipation from the mixed method

[15] A.E.P. Veldman, *Numerieke Stromingsleer*, Rijksuniversiteit Groningen (1993)

[16] A.E.P. Veldman, *Partiële Differentiaalvergelijkingen*, Rijksuniversiteit Groningen (1993)

[17] J.H. Wilkinson, *The algebraic eigenvalue problem*, Monographs on numerical analysis, Clarendon Press, Oxford (1965)