Disturbance evolution in the near-wake behind a flat plate

Diplomarbeit
von
cand. aer. Janis Mühlratzer

durchgeführt am
Institut für Aerodynamik und Gasdynamik
der Universität Stuttgart
und an der
Cardiff University
School of Mathematics

Stuttgart / Cardiff, im März 2007

Universität Stuttgart

INSTITUT FÜR AERODYNAMIK UND GASDYNAMIK
DIREKTOR: PROF. DR.-ING. EWALD KRÄMER
IAG

Pfaffenwaldring 21 · 70550 Stuttgart Apl. Prof. Dr.-Ing. Ulrich Rist

Diplomarbeit für Herrn cand. aer. Janis Mühlratzer:

"Disturbance evolution in the near-wake behind a flat plate"

A number of recent theoretical studies[1,2,3] have considered the behaviour of disturbances in a two-dimensional thin wake formed behind a streamlined body. The absolute or convective nature of the linear instability, together with some effects of nonlinearity, have been investigated using various asymptotic approximations for the flow at high Reynolds numbers. For example, Smith, Bowles and Li[3] showed that in the near-wake region just aft of the trailing edge, nonlinearity could provoke the upstream propagation of inviscid forms of disturbance.

For the present work the velocity-vorticity method developed by Davies and Carpenter[4,5] is made available to carry out direct numerical simulations. The main intention is to further investigate, at finite Reynolds numbers, the evolution of two-dimensional disturbances in a two-dimensional near-wake behind a streamlined body. In particular, model problems involving the spatiotemporal development of initially localized free disturbances in the near-wake behind an aligned flat plate will be studied. The computation of the basic state in the vicinity of the trailing edge of the plate will first be validated by making careful comparisons with the structures obtained from asymptotic analysis. For this the influence of the Reynolds number is to be shown.

In a second part of the work a stability analysis of the obtained velocity profiles is to be performed and compared to the results for analytical velocity profiles of the $sech^2(y)$-type. An appropriate scheme to solve the Orr-Sommerfeld equation will be provided.

Literatur

1. Woodley B. M. and Peake N.: *Global linear stability analysis of thin aerofoil wakes,* J. Fluid Mech. **339** (1997) 239-260.
2. Turkyilmazaglu M., Gajjar J. S. B. and Ruban A. I.: *The Absolute Instability of Thin Wakes in an Incompressible/Compressible Fluid,* Theoret. Comput. Fluid Dynamics **13** (1999) 91-114.
3. Smith F. T., Bowles R. G. A. and Li L.: *Nonlinear effects in absolute and convective instabilities of a near-wake,* Eur. J. Mech. B - Fluids **19** (2000) 173-211.
4. Davies C. and Carpenter P. W.: *A Novel Velocity-Vorticity Formulation of the Navier-Stokes Equations with Applications for Boundary Layer Disturbance Evolution,* J. Comp. Phys. **172** 119-165.
5. Bowles R. I., Davies C. and Smith F. T.: *On the spiking stages in deep transition and unsteady separation,* J. Eng. Math. (2003) in press.

Betreuer: Ulrich Rist Stuttgart,
Christopher Davies, School of Mathematics,
Cardiff University

Ausgabedatum: 1.10.2006

Abgabedatum: 31.3.2007 apl. Prof. Dr.-Ing. Ulrich Rist

Übersicht

In dieser Arbeit wird die Untersuchung der Störungsausbreitung in der Nachlaufströmung einer ebenen Platte über zwei Wege angegangen. Direkte numerische Simulation der Strömung ist eine Methode, die jedoch nicht zum Abschluß gebracht werden konnte, da ein gänzlich stabiler Lauf des Codes, der für diese Simulationen zur Verfügung stand nicht erreicht werden konnte. Die zweite Variante ist, die ORR-SOMMERFELD-Gleichung, sowohl für eine Nachlaufströmung aus der numerischen Simulation, als auch für einen analytisch definierten sech^2-förmigen Nachlauf, zu lösen.

Nach der Vorstellung der grundlegenden Eigenschaften der Strömung und der zugrundeliegenden Gleichungen werden die numerischen Methoden und das Lösungsverfahren des Codes aufgezeigt. Weiterhin werden Änderungen und Erweiterungen des Codes und neu erstellte Programme für die Datenaufbereitung dargestellt. Anschließend wird dargelegt, welche Parametereinstellungen für die Simulationen gewählt wurden, um die Suche nach der Ursache der künstlichen Instabilität zu unterstützen.

Ebenfalls wird die Grundströmung aus der Simulation mit den Vorhersagen der asymptotischen Theorie (Dreidecktheorie, welche kurz erläutert wird) für den Bereich um die Hinterkante der Platte verglichen, um die Qualität und Glaubwürdigkeit der vom Code erzeugten Strömung zu überprüfen.

Schließlich wird das Verfahren zur Lösung des Eigenwertproblems mit einem linearen Löser für die zwei Grundströmungen umrissen. Die Ergebnisse dieser Eigenwertsuche werden dargestellt, die Stabilität der zwei Strömungstypen verglichen und die Abhängigkeit der Stabilitätseigenschaften von verschiedenen Parametern diskutiert.

Abstract

The investigation of the disturbance evolution in the near-wake behind a flat plate is approached via two ways in this thesis. The direct numerical simulation of the flow is one method, which could not be brought to an end because the code available for this simulation could not be made to run entirely stable. The second technique is to solve the Orr-Sommerfeld equation for both a wake flow obtained from numerical simulation and a wake flow defined by an analytical function of the sech^2-type.

After presenting the basic flow properties and the governing equations for the flow, the numerical approach and the solution scheme of the code are reviewed. Furthermore, changes and extensions to this code as well as separate programmes written for post-processing are presented. Subsequently it is shown under which settings the simulations were actually carried out to support the search for the root of the artificial instability.

The basic state from the simulation is compared with what asymptotical theory (triple-deck theory, which is reviewed in short) predicts for the region around the trailing edge of the plate to examine the quality and credibility of the flow the code produces.

Finally, the approach to solving the eigenvalue problem for the two basic flow types using a linear solver is outlined. The results of the eigenvalue solving are presented, the stability of the two wake flow types is compared and dependence of the stability properties from several parameters is discussed.

Contents

List of Figures

List of Tables

Nomenclature

The dimension is given only for variables also appearing in dimensional form in this thesis. The dimension only applies to the dimensional usage of a variable, which will be denoted by an asterisk extending the symbol.

A	[-]	General variable
A	[-]	Slip velocity
a	[-]	General variable
A_f	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	Forcing amplitude
b	[-]	General variable
c	[-]	Phase velocity
c_{ph}	[-]	Real phase velocity
d	[-]	General variable
e	[-]	Parameter of the buffer ramp-down function
$\mathcal{F}$	[-]	sin transform
f	[-]	BLASIUS function
f_b	[-]	Body force
g	[-]	Parameter of the buffer ramp-down function
h	[-]	Interpolation function
i	[-]	Imaginary number
i	[-]	Iteration step index
j	[-]	Wall-normal step index
k	[-]	Streamwise step index
ℓ	[m]	Length
l	[-]	Time step index
l	[m]	Stretching parameter
l_B	[-]	Buffer length
l_{xf}	[-]	Forcing x-scaling
l_{yf}	[-]	Forcing y-scaling
m	[-]	Buffer ramp-down function
m	[-]	CHEBYSHEV coefficient index
N	[-]	Accumulative right-hand side quantity of vorticity transport equation
n	[-]	General index
N_{it}	[-]	Number of iterations

N_x	[-]	Number of streamwise grid points
N_y	[-]	Number of wall-normal grid points / CHEBYSHEV order
$\mathcal{O}$	[-]	Order of
P	[Pa]	Base flow pressure
p	[-]	Profile parameter
p	[Pa]	Perturbation pressure
Q	[-]	Profile velocity ratio
Re	[-]	REYNOLDS number
r_t	[-]	Forcing scaling function in t
r_x	[-]	Forcing scaling function in x
r_y	[-]	Forcing scaling function in y
T	[-]	CHEBYSHEV coefficient
T	[s]	Total time / periodic time
t	[s]	Time
$\vec{U}$	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Base flow velocity vector
$\vec{u}$	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Perturbation velocity vector
U	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Streamwise base flow velocity
u	[m]	Streamwise perturbation velocity
V	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Normal base flow velocity
v	$[\frac{\mathrm{m}}{\mathrm{s}}]$	Normal perturbation velocity
x	[m]	Streamwise dimension
x_{end}	[-]	Outflow x-location
x_f	[-]	Forcing x-location
x_{TE}	[-]	Trailing edge x-location
y	[m]	Normal dimension
α	[-]	Wavenumber
β	[-]	General multiple
χ	[-]	Triple-deck region length
δ	[m]	Characteristic length: displacement thickness or wake half-width
ϵ	[-]	Triple-deck constant
ϕ	[-]	Eigenfunction
φ	[-]	Phase shift
γ	[-]	EUCLIDEAN distance of the eigenvalues
η	[-]	BLASIUS variable
η	$[\frac{\mathrm{kg}}{\mathrm{m\,s}}]$	Dynamic viscosity
λ	[-]	Wavelength
ν	$[\frac{\mathrm{m}^2}{\mathrm{s}}]$	Kinematic viscosity
ψ	[-]	Perturbation streamfunction
Ψ	[-]	Base flow streamfunction
ρ	$[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	Density
ϱ	[-]	Relaxation parameter

ϑ	[-]	Intermediate wall-normal coordinate
ω	$[\frac{\mathrm{rad}}{\mathrm{s}}]$	(Angular) frequency
ω_z	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	Perturbation vorticity
Ω_z	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	Base flow vorticity
ζ	[-]	Mapped wall-normal coordinate
∞	[-]	Free stream variable
c	[-]	Complex
r	[-]	Real
i	[-]	Imaginary
$pert$	[-]	Perturbation value
$\_$	[-]	Lower half of domain
$[y]$	[-]	In y-coordinates
$[\eta]$	[-]	In η-coordinates
$*$	[-]	Dimensional variable
t	[-]	Total variable
$\hat{}$	[-]	Symmetric part
$\check{}$	[-]	Antisymmetric part
$'$	[-]	First total derivative with respect to y
$\tilde{}$	[-]	Predictor value
$\bar{}$	[-]	Upper half of domain
$\breve{}$	[-]	Iterated value
L	[-]	Lower deck perturbation variable
M	[-]	Main deck perturbation variable
U	[-]	Upper deck perturbation variable

Part I.

Introduction

The Subject

Studies of disturbance evolution in fluid flow over and behind streamlined bodies are of high importance to many domains of fluid dynamics. Practical phenomena like transition and turbulence, which are crucially influencing the flow characteristics of a given flow configuration, are determined by the stability properties of the flow under the given conditions. Accordingly, qualities like load distribution, manoeuvrability and drag of air, ground and water vehicles are related to the question of hydrodynamic stability.

Besides laboratory experiments and several types of theoretical analysis, numerical simulation is a competitive and powerful way to investigate disturbance evolution. Numerical simulations are not only relatively cheaper, quicker to set up and better repeatable than physical experiments, but also allow to conduct "experiments" in laminar flow at REYNOLDS numbers that naturally would come along with turbulent flow. The adaptability to nearly any arbitrary geometry as well as discretionary inclusion or elimination of three-dimensional effects and of non-linear effects are other advantages of computational fluid dynamics.

The basic idea to the numerical examination of disturbance evolution like it should be pursued in this thesis is to simulate a stable basic wake flow of a flat plate at a fixed REYNOLDS number and to introduce small oscillating disturbances at a certain point in space, with defined frequency and amplitude. Repeating this with varying parameters within a range of values yields the quantitative correlation of stability behaviour with each of the parameters.

Objective of This Thesis

The original scope of this thesis was two-fold: first, a code developed at Cardiff University School of Mathematics, proven and used for boundary-layer flow at several configurations and adapted to allow the inclusion of a trailing edge, shall be tested and validated to simulate disturbance evolution in the wake of a flat plate properly. Second, the stability properties of the resulting wake flow shall be examined in detail and compared on the one hand with known results from literature and on the other hand with the stability behaviour of a prescribed mean flow – given by an analytic function reproducing the wake flow velocity profile shape, for which reference stability properties can even be derived analytically and are available in literature.

During the efforts to test the code provided it turned out that with any disturbance introduced the simulation becomes numerically instable and quickly leads to unbounded values of the flow variables for any circumstances but the undisturbed base flow. Though intense and exhausting search for the reason for this instability was done, it was not possible to find the cause. Reconstruction and detailed analysis of every part of this complex code is a task on its own for a separate study.

Unfortunately, much of the work done in preparation for the numerical stability research could not be used subsequently due the fact that no stable simulation could be obtained. Nevertheless these considerable efforts shall be presented here, because they represent the major part of the work done and can serve as a helpful basis for a new attempt to use the code for the stability analysis once numerical stability is achieved. And yet it remains a scope of this thesis to present the general approach to the problem, the mathematical foundations of the code used and the basics of stability analysis.

As an alternative way to achieve insight into the stability of the flow under consideration, in spite of the unavailability of the direct numerical simulation, we will go via solving the underlying eigenvalue problem. For this, a versatile numerical solver, in the configuration used here actually solving the ORR-SOMMERFELD equation and provided by Institut für Aerodynamik und Gasdynamik der Universität Stuttgart, is used. The solver delivers – after adaptation and parameter tuning – the eigenvalues for the given flow configuration, using the mean flow obtained from running the original code as well as the prescribed mean flow. With the results from this eigenvalue analysis an important and helpful basis of information is at hand for the assessment of the stability behaviour of the direct numerical simulation. When this can be performed one day, a reference of the stability properties for the flow under consideration is available from a set-up clear of numerical simulation hitches.

Part II.

Main Part

1. Theoretical Background

This chapter shall present the basic flow properties, the governing equations for the flow and basics of fluid flow stability. These basics apply and are relevant independently of the type of the further approach to the stability analysis (direct numerical simulation or eigenvalue solving).

1.1. Basic Flow Properties

Our simulation "arena" (FIG. 1.1 (p. 6)) is the flow (i) over an aligned flat plate and (ii) in the wake of this plate with a physical domain infinitely stretching to either side of the plate / centreline. The domain is limited to a settable length in the direction along the plate. The flow is assumed to be two-dimensional and nonlinear effects shall be resolved by the calculations.

We aim for direct numerical simulations of the flow of a viscid incompressible fluid over a range of REYNOLDS numbers in this domain.

1.1.1. Flow Decomposition and Terminology

The total flow can be considered as a composition of base flow and perturbation. As we assume parallel base flow, it can be written as $\vec{U} = (U, 0)$, while the perturbations have components in both dimensions, $\vec{u} = (u, v)$. The total field then is

$$U^t = U + u \tag{1.1}$$
$$V^t = v \tag{1.2}$$
$$P^t = P + p \tag{1.3}$$
$$\Omega_z^t = \Omega_z + \omega_z \ . \tag{1.4}$$

Since parallel base flow does not mean that the whole flow must be parallel, any departure of the basic flow from parallelism is also represented by the perturbation variables. In differentiation to the parallel *base flow*, a flow including a possible perturbation from non-parallelism (not from introduction of the external disturbance i. e. the body forcing, cf. 2.3) shall be called *mean flow* in this thesis. In summary:

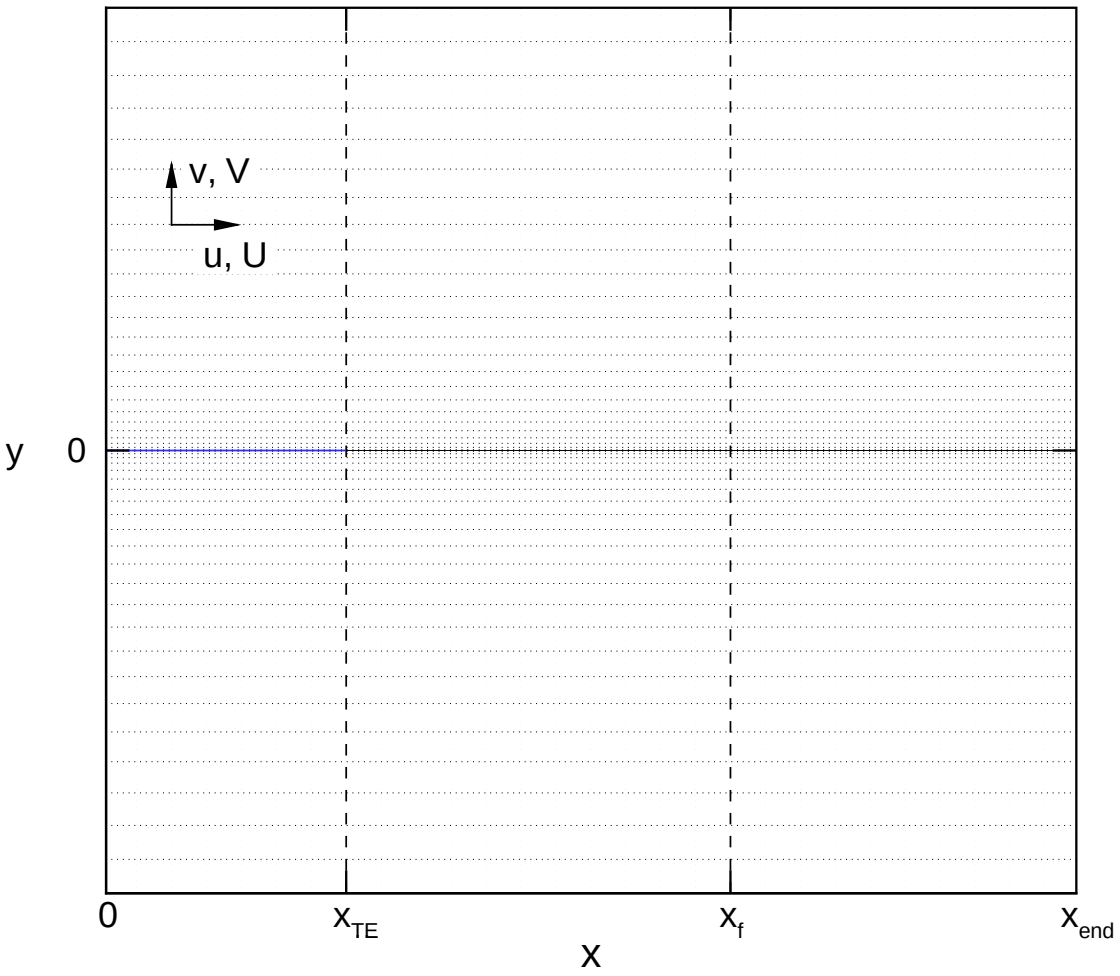

FIGURE 1.1. – *The domain of the simulations including the flat plate.*

- $\vec{U}$ – base flow
- $\vec{u}$ – perturbation
- $\vec{U} + \vec{u}|_{A_f=0}$ – mean flow

The definition of vorticity used here is

$$\Omega_z := \frac{\partial U}{\partial y} - \frac{\partial V}{\partial x} \tag{1.5}$$

$$\omega_z := \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \, . \tag{1.6}$$

1.1.2. Non-dimensionalisation of Variables

In order to work with practical values and to get universally valid results that can be rescaled to any given physical problem, we divide all dimensional variables (denoted by an asterisk) by appropriate variables and thereby get non-dimensional variables (without asterisk). All non-dimensionalisations hold for perturbation, base flow and total flow variables.

Wall-normal coordinate

We use the Blasius variable to scale and non-dimensionalise the wall-normal coordinate:

$$\eta := \frac{y^*}{\sqrt{\frac{2x^*\nu^*}{U_\infty^*}}} \,. \tag{1.7}$$

Lengths

Length variables are brought to a non-dimensional form by scaling them with a characteristic length δ^*. In the case of the flat plate this length is the boundary layer thickness (cf. 1.4), for the wake simulation it is the wake half-width (cf. 1.7).

$$x = \frac{x^*}{\delta^*} \tag{1.8}$$

$$y = \frac{y^*}{\delta^*} \tag{1.9}$$

$$\ell = \frac{\ell^*}{\delta^*} \tag{1.10}$$

Velocities and Pressure

Velocities are non-dimensionalised by scaling them with the undisturbed free-stream velocity; the pressure is divided by twice the dynamic pressure:

$$u = \frac{u^*}{U_\infty^*} \qquad U = \frac{U^*}{U_\infty^*} \qquad U^t = \frac{U^{*t}}{U_\infty^*} \tag{1.11}$$

$$v = \frac{v^*}{U_\infty^*} \qquad V = \frac{V^*}{U_\infty^*} \qquad V^t = \frac{V^{*t}}{U_\infty^*} \tag{1.12}$$

$$p = \frac{p^*}{\rho U_\infty^{*2}} \qquad P = \frac{P^*}{\rho U_\infty^{*2}} \qquad P^t = \frac{P^{*t}}{\rho U_\infty^{*2}} \tag{1.13}$$

Time and Frequency

A characteristic time derived from the variables already used is δ^*/U_∞^* and therefore:

$$t = \frac{t^* U_\infty^*}{\delta^*} \,, \tag{1.14}$$

$$T = \frac{T^* U_\infty^*}{\delta^*} \,. \tag{1.15}$$

Using

$$\omega^* = \frac{2\pi}{T^*} \tag{1.16}$$

$$= \frac{2\pi U_\infty^*}{T\delta^*} \tag{1.17}$$

and

$$\omega = \frac{2\pi}{T} \,, \tag{1.18}$$

we get for the non-dimensionalisation of the frequency

$$\omega = \frac{\omega^* \delta^*}{U_\infty^*} \,. \tag{1.19}$$

1.1.3. Reynolds Number

Unless denoted by a subscript giving the respective length, we use the Reynolds Number based on the boundary layer thickness throughout this thesis:

$$Re := Re_{\delta^*} = \frac{U_\infty^* \delta^*}{\nu^*} \,. \tag{1.20}$$

For further relations involving the Reynolds Number cf. 1.5.

1.2. Governing Equations

1.2.1. Navier-Stokes Equations in Velocity-Vorticity Formulation

According to our premises of sec. 1.1, the appropriate basic equations for this two-dimensional problem are the full Navier-Stokes equations for viscous incompressible flow: the momentum equation

$$\frac{\partial \vec{U}^{*t}}{\partial t} + \left(\vec{U}^{*t} \cdot \nabla\right) \vec{U}^{*t} = -\frac{1}{\rho^*} \nabla P^{*t} + \nu^* \nabla^2 \vec{U}^{*t} \,, \tag{1.21}$$

and the continuity equation

$$\nabla \cdot \vec{U}^{*t} = 0 \,, \tag{1.22}$$

together with appropriate boundary conditions to close the system (cf. 1.3).

Division by U_∞^{*2}/δ^* and application of the aforementioned non-dimensionalisations leads to

$$\frac{\partial \vec{U}^t}{\partial t} + \left(\vec{U}^t \cdot \nabla\right) \vec{U}^t = -\nabla P^t + \frac{1}{Re} \nabla^2 \vec{U}^t \tag{1.23}$$

and

$$\nabla \cdot \vec{U}^t = 0 \,. \tag{1.24}$$

We will derive the governing equations for perturbations rather then for the total flow as having the perturbation values readily at hand is more convenient for the stability analysis.

Davies and Carpenter (2001) gave an equivalent novel velocity-vorticity formulation of the Navier-Stokes equations which entails several advantages compared to the classical formulation in primitive variables. First advantage is the removal of the pressure and a reduction in the number of variables and governing equations, which reduces demand of computational resources.

The derivation of this formulation is as follows: The variables of the momentum equation are decomposed according to (1.1)–(1.3) and the x- and y-components of the momentum equation are derived with respect to y and x respectively, and then subtracted from each other. Using the definition of vorticity (1.5) and continuity (1.24) as well as the premise of a steady parallel basic flow (i. e. $V = U_t = U_x = 0$) one gets eventually[1]

$$\frac{\partial \omega_z}{\partial t} + U\frac{\partial \omega_z}{\partial x} + u\frac{\partial \omega_z}{\partial x} + v\frac{\partial \omega_z}{\partial y} + v\frac{\mathrm{d}^2 U}{\mathrm{d}y^2} = \frac{1}{Re}\left(\frac{\partial^2 \omega_z}{\partial x^2} + \frac{\partial^2 \omega_z}{\partial y^2}\right) \,. \tag{1.25}$$

This vorticity transport equation is complemented by the continuity equation (conservation of mass for an incompressible fluid), combined, again, with the definition of vorticity, which leads to the Poisson equation

$$\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = -\frac{\partial \omega_z}{\partial x} \,. \tag{1.26}$$

[1] The term $+\frac{1}{Re}\frac{\mathrm{d}^3 U}{\mathrm{d}y^3}$ in principle appearing on the right-hand side of the equation must be zero because otherwise, even in the absence of any initial perturbation, it would produce a non-zero right-hand side and thereby generate a non-parallel base flow.

We now have one differential equation (1.25) for the vorticity ω_z and one equation (1.26) to derive the velocity component v from that. The x-wise component u is obtained by solving the integrated definition of vorticity which stems from (1.5)

$$u = -\int_y^\infty \left(\frac{\partial v}{\partial x} + \omega_z \right) \mathrm{d}y \; . \tag{1.27}$$

This implies, for the vanishing integral $\int_\infty^\infty \ldots$, that the perturbation u vanishes at infinity:

$$u|_\infty = 0 \; . \tag{1.28}$$

When of interest, the pressure can be derived by integrating the ordinary y-momentum equation:

$$p = \int_y^\infty \left(\frac{\partial v}{\partial t} + (U+u)\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} - \frac{1}{Re}\nabla^2 v \right) \mathrm{d}y \; . \tag{1.29}$$

As u and p can be expressed in terms of the other variables and therefore be removed from consideration entirely, they are called the "secondary" variables; ω_z and v are referred to as "primary" variables consequently.

It is worth noting that the parallel basic flow $\vec{U} = \big(U(y), 0\big)$ was not subjected to the NAVIER-STOKES equations in this derivation, therefore it does not have to be an exact solution of these. The only constraint is

$$\frac{\mathrm{d}^3 U}{\mathrm{d}y^3} = 0 \tag{1.30}$$

as mentioned in footnote 1. It can be shown that the effects of omitting that term are much weaker (i. e. asymptotically smaller) than the effects that can be described in terms of the triple deck structure in the near wake around the trailing edge and the associated strongly non-parallel flow, which will be discussed in chapter 4.

1.2.2. Equivalence with the NAVIER-STOKES Equations in Primitive Variables Formulation

With some operations it can be shown conversely that (1.25) and (1.26) are equivalent to the original form of the NAVIER-STOKES equations (1.23) and (1.24) – the principal idea of this will be shown below because it provides some conditions that have to be fulfilled in order to keep the equivalence.

The comprehension of the y-momentum equation is directly obvious because the pressure was defined via this equation.

Differentiating (1.27) with respect to x and substituting for ω_z by means of (1.26) yields (minding the rules for improper integrals)

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = \left.\frac{\partial v}{\partial y}\right|_{\infty} , \tag{1.31}$$

so for satisfying the continuity equation (1.24), it must be

$$\left.\frac{\partial v}{\partial y}\right|_{\infty} = 0 . \tag{1.32}$$

Deriving the conditions stemming from satisfying the x-momentum equation is slightly more laborious. In order to do this we differentiate (1.29) with respect to x and cancel out respective terms using the continuity equation (1.24), which gives

$$\frac{\partial p}{\partial x} = \int_y^{\infty} \left(\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x}\right) + (U+u)\frac{\partial^2 v}{\partial x^2} + v\frac{\partial^2 v}{\partial x\,\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\frac{\partial v}{\partial x} \right) \mathrm{d}y . \tag{1.33a}$$

Now we use $\frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} - \omega_z$ from (1.5):

$$\frac{\partial p}{\partial x} = \int_y^{\infty} \left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \right) \left(\frac{\partial u}{\partial y} - \omega_z \right) \mathrm{d}y . \tag{1.33b}$$

When subtracting the vorticity transport equation (1.25) from this, the result is (with $'$ denoting the total derivative $\frac{\mathrm{d}}{\mathrm{d}y}$)

$$\frac{\partial p}{\partial x} = \int_y^{\infty} \left(\left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \right) \frac{\partial u}{\partial y} + vU'' \right) \mathrm{d}y . \tag{1.33c}$$

We then apply the partial derivative $\frac{\partial}{\partial y}$ on the product of the term in the inner brackets and u, rather than multiplying the term in the inner brackets with the partial derivative of u. This requires equalising the terms arising from the product rule by subtracting them subsequently:

$$\begin{aligned} \frac{\partial p}{\partial x} = \int_y^{\infty} \Bigg(& \frac{\partial}{\partial y}\left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \right) u - \\ & - \left(U' + \frac{\partial u}{\partial y} \right)\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\frac{\partial u}{\partial y} + vU'' \Bigg) \mathrm{d}y , \end{aligned} \tag{1.33d}$$

where continuity equation, again, cancels some terms so that

$$\frac{\partial p}{\partial x} = \int_y^\infty \left(\frac{\partial}{\partial y} \left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right) u + U'\frac{\partial v}{\partial y} + vU'' \right) \mathrm{d}y \,. \tag{1.33e}$$

Now we can simply rearrange and evaluate the integral:

$$\frac{\partial p}{\partial x} = \int_y^\infty \frac{\partial}{\partial y} \left(\left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right) u + vU' \right) \mathrm{d}y \tag{1.33f}$$

$$= \left[\left(\frac{\partial}{\partial t} + (U+u)\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right) u + vU' \right]_y^\infty \,. \tag{1.33g}$$

So, to be able to recover the x-momentum equation, which is the lower limit of the definite integral above:

$$-\frac{\partial p}{\partial x} = \frac{\partial u}{\partial t} + (U+u)\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + vU' - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \,, \tag{1.34}$$

the upper limit of this integral must vanish. Since $u|_\infty = 0$ (1.28), this amounts to requiring that

$$\left(v\left(U' + \frac{\partial u}{\partial y} \right) - \frac{1}{Re}\frac{\partial^2 u}{\partial y^2} \right)\Bigg|_\infty = 0 \tag{1.35a}$$

or

$$\left(v\left(U' + \frac{\partial u}{\partial y} \right) - \frac{1}{Re}\frac{\partial}{\partial y}\left(\omega_z + \frac{\partial v}{\partial x} \right) \right)\Bigg|_\infty = 0 \,, \tag{1.35b}$$

where (1.5) was used. This equation is satisfied if

$$v|_\infty = 0 \tag{1.36}$$

and

$$\left.\frac{\partial \omega_z}{\partial y}\right|_{y\to\infty} = 0 \,. \tag{1.37}$$

To sum it up, the velocity-vorticity formulation of the Navier-Stokes equation is equivalent to the "classical" form in primitive variables if

$$\lim_{y\to\infty} v = \lim_{y\to\infty} \frac{\partial v}{\partial y} = \lim_{y\to\infty} \frac{\partial \omega_z}{\partial y} = 0 \,.$$

As to how these conditions can be applied in the numerical representation the discussion will be held in 2.1.2.

1.3. Boundary Conditions and Symmetry

Solving the differential equations (1.5) and (1.26) requires appropriate boundary conditions. In order to specify them in a manner that allows for efficient numerical calculations, we use the concept of symmetric and antisymmetric decomposition of the flow field.

1.3.1. Symmetric and Antisymmetric Decomposition

The two half-planes of the computational domain ($\bar{\cdot}$ denotes upper half, $\underline{\cdot}$ lower half) are readily represented by a decomposition of all variables in the field into a symmetric ($\check{\cdot}$) and an antisymmetric ($\hat{\cdot}$) part according to

$$\bar{a} = \check{a} + \hat{a} \tag{1.38}$$

$$\underline{a} = \check{a} - \hat{a} \; . \tag{1.39}$$

Splitting the field into symmetric and antisymmetric parts also gives the advantage of storing and manipulating the values in the numerical simulation easily, as it will be discussed in 2.4.

The upper and lower half-plane can both be assigned an own coordinate system as

$$\bar{y} = y \tag{1.40}$$

$$\underline{y} = -y \; . \tag{1.41}$$

which maps them onto two overlaid semi-infinite domains with positive coordinate values.

Still, these half-planes match at $y = \bar{y} = \underline{y} = 0$ and the condition of uniqueness (i. e. one single value for one physical point) imposes a constraint at this line, and the symmetric and antisymmetric part of every variable get different parity from this:

$$\bar{u}|_{\bar{y}=0} = \underline{u}|_{\underline{y}=0} \quad \Rightarrow \quad \check{u}|_{y=0} = 0 \tag{1.42}$$

$$\bar{v}|_{\bar{y}=0} = -\underline{v}|_{\underline{y}=0} \quad \Rightarrow \quad \hat{v}|_{y=0} = 0 \tag{1.43}$$

$$\bar{\omega}_z|_{\bar{y}=0} = -\underline{\omega}_z|_{\underline{y}=0} \quad \Rightarrow \quad \hat{\omega}_z|_{y=0} = 0 \tag{1.44}$$

(This can be derived formally by using the definition of vorticity (1.5) and the results for u and v above.)

$$\bar{p}|_{\bar{y}=0} = \underline{p}|_{\underline{y}=0} \quad \Rightarrow \quad \check{p}|_{y=0} = 0 \tag{1.45}$$

$$\left.\frac{\partial \bar{u}}{\partial x}\right|_{\bar{y}=0} = \left.\frac{\partial \underline{u}}{\partial x}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial \check{u}}{\partial x}\right|_{y=0} = 0 \tag{1.46}$$

$$\left.\frac{\partial \bar{v}}{\partial x}\right|_{\bar{y}=0} = -\left.\frac{\partial \underline{v}}{\partial x}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial \hat{v}}{\partial x}\right|_{y=0} = 0 \tag{1.47}$$

$$\left.\frac{\partial\bar{\omega}_z}{\partial x}\right|_{\bar{y}=0} = -\left.\frac{\partial\underline{\omega}_z}{\partial x}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\hat{\omega}_z}{\partial x}\right|_{y=0} = 0 \tag{1.48}$$

$$\left.\frac{\partial\bar{p}}{\partial x}\right|_{\bar{y}=0} = \left.\frac{\partial\underline{p}}{\partial x}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\check{p}}{\partial x}\right|_{y=0} = 0 \tag{1.49}$$

$$\left.\frac{\partial\bar{u}}{\partial y}\right|_{\bar{y}=0} = -\left.\frac{\partial\underline{u}}{\partial y}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\hat{u}}{\partial y}\right|_{y=0} = 0 \tag{1.50}$$

$$\left.\frac{\partial\bar{v}}{\partial y}\right|_{\bar{y}=0} = \left.\frac{\partial\underline{v}}{\partial y}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\check{v}}{\partial y}\right|_{y=0} = 0 \tag{1.51}$$

$$\left.\frac{\partial\bar{\omega}_z}{\partial y}\right|_{\bar{y}=0} = \left.\frac{\partial\underline{\omega}_z}{\partial y}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\check{\omega}_z}{\partial y}\right|_{y=0} = 0 \tag{1.52}$$

$$\left.\frac{\partial\bar{p}}{\partial y}\right|_{\bar{y}=0} = \left.\frac{\partial\underline{p}}{\partial y}\right|_{\underline{y}=0} \quad \Rightarrow \quad \left.\frac{\partial\check{p}}{\partial y}\right|_{y=0} = 0 \tag{1.53}$$

$$\vdots$$

These basic results, of course, may not be imposed on the flow field simultaneously but they can be employed to give two boundary conditions for the two primary variables – as will be shown below.

The property that the product of an arbitrary symmetric variable and an arbitrary antisymmetric variable is antisymmetric, the product of any two symmetric variables is symmetric and the product of any two antisymmetric variables also is symmetric:

$$\hat{a}\,\check{b} = \check{d}\,, \tag{1.54a}$$

$$\hat{a}\,\hat{b} = \hat{d} \text{ and} \tag{1.54b}$$

$$\check{a}\,\check{b} = \hat{d}\,. \tag{1.54c}$$

will be used in the following.

1.3.2. Derivation of the Boundary Conditions

On the Rigid Wall

The no-slip boundary condition at a rigid wall, i. e. over the flat plate, is $U^t(y=0)=0$ and $V^t(y=0)=0$. With a base flow that is subjected to viscous adhesion ($U(y=0)=0$ and $V(y=0)=0$) – insofar now imposing a condition on the base flow – this also results

in

$$u|_{y=0} = 0 \quad \Rightarrow \quad \hat{u}|_{y=0} = 0 \quad \wedge \quad \check{u}|_{y=0} = 0 \tag{1.55}$$
$$v|_{y=0} = 0 \quad \Rightarrow \quad \hat{v}|_{y=0} = 0 \quad \wedge \quad \check{v}|_{y=0} = 0 \quad . \tag{1.56}$$

As we operate with the primary variables, the condition for u has to be translated into a condition involving ω_z, which we get when we substitute the results of (1.55) into (1.27):

$$\int_0^\infty \left(\hat{\omega}_z + \frac{\partial \hat{v}}{\partial x}\right) \mathrm{d}y = 0 \qquad \wedge \qquad \int_0^\infty \left(\check{\omega}_z + \frac{\partial \check{v}}{\partial x}\right) \mathrm{d}y = 0 \quad . \tag{1.57}$$

On the Wake Centreline

As there is no no-slip condition in the wake, we need different boundary conditions there. Firstly we can derive a condition for the symmetric parts as the condition of uniqueness holds for the total flow – here we use the total vorticity in particular – as well:

$$\bar{\Omega}_z|_{\bar{y}=0} = -\underline{\Omega}_z|_{\underline{y}=0} \Rightarrow \hat{\Omega}_z|_{y=0} = 0 \tag{1.58a}$$
$$\left(\hat{\omega}_z + \frac{\partial \hat{U}}{\partial y} - \underbrace{\frac{\partial \hat{V}}{\partial x}}_{=0}\right)\Bigg|_{y=0} = 0 \tag{1.58b}$$

and therefore

$$\left(\hat{\omega}_z + \frac{\mathrm{d}\hat{U}}{\mathrm{d}y}\right)\Bigg|_{y=0} = 0 \, . \tag{1.58c}$$

For the antisymmetric set we employ a condition on the pressure as we substitute $\check{p}|_{y=0} = 0$ (1.45) into (1.29). Respecting (1.54a) the antisymmetric part of this pressure equation for $y = 0$ gives

$$0 = \int_0^\infty \left(\frac{\partial \check{v}}{\partial t} + \left(\hat{U} + \hat{u}\right)\frac{\partial \check{v}}{\partial x} + \check{u}\frac{\partial \hat{v}}{\partial x} + \hat{v}\frac{\partial \check{v}}{\partial y} + \check{v}\frac{\partial \hat{v}}{\partial y} + \frac{1}{Re}\frac{\partial \check{\omega}_z}{\partial x}\right) \mathrm{d}y \, . \tag{1.59}$$

We made use of the assumption that the base flow U was symmetric, hence $\check{U} = 0$, and replaced the last term as $-\nabla^2 v = \frac{\partial \omega_z}{\partial x}$, as the Poisson equation states. The product rule allows to separately integrate two terms:

$$\int_0^\infty \left(\hat{v}\frac{\partial \check{v}}{\partial y} + \check{v}\frac{\partial \hat{v}}{\partial y}\right) \mathrm{d}y = \left[\hat{v}\check{v}\right]_0^\infty \tag{1.60}$$
$$= 0 \, , \tag{1.61}$$

since $\hat{v}|_\infty = \check{v}|_\infty = 0$ (1.36) and $\hat{v}|_{y=0} = 0$ (1.43). Subsequently, our condition on the wake flow, derived from the pressure condition finally is

$$0 = \int_0^\infty \left(\frac{\partial \check{v}}{\partial t} + \left(\hat{U} + \hat{u} \right) \frac{\partial \check{v}}{\partial x} + \check{u} \frac{\partial \hat{v}}{\partial x} + \frac{1}{Re} \frac{\partial \check{\omega}_z}{\partial x} \right) \mathrm{d}y \, . \tag{1.62}$$

The interesting point is that this condition is linear in the antisymmetric part of the field $\{\check{u}, \check{v}, \check{\omega}_z\}$.

Alternative Antisymmetric Boundary Condition An alternative boundary condition for the antisymmetric part can be found using $\frac{\partial \check{v}}{\partial y}|_{y=0} = 0$ and $\frac{\partial \check{\omega}_z}{\partial y}|_{y=0} = 0$. Taking into account (1.54a)–(1.54c) and assuming that the conditions for the validity of (1.34) hold, we get from (1.34):

$$\frac{\partial \check{p}}{\partial x} = \left(\frac{\partial}{\partial t} + \left(\hat{U} + \hat{u} \right) \frac{\partial}{\partial x} + \hat{v} \frac{\partial}{\partial y} - \frac{1}{Re} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right) \check{u} + \check{u} \frac{\partial \hat{u}}{\partial x} + \check{v} \frac{\partial \hat{u}}{\partial y} + \check{v} \hat{U}' \, . \tag{1.63}$$

Using the implication from continuity $\frac{\partial \check{v}}{\partial y}|_{y=0} = 0 \Rightarrow \frac{\partial \check{u}}{\partial x}|_{y=0} = 0$ this leads to

$$\left. \frac{\partial \check{p}}{\partial x} \right|_{y=0} = \left. \left(\frac{\partial \check{u}}{\partial t} - \frac{1}{Re} \frac{\partial^2 \check{u}}{\partial y^2} + \check{u} \frac{\partial \hat{u}}{\partial x} \right) \right|_{y=0} , \tag{1.64}$$

where replacing

$$\left. \left(\check{v} \frac{\partial \hat{u}}{\partial y} + \check{v} \hat{U}' \right) \right|_{y=0} = \left. \left(\check{v} \left(\frac{\partial \hat{u}}{\partial y} + \hat{U}' \right) \right) \right|_{y=0} = 0 \, , \tag{1.65}$$

because of (1.58c) and (1.43).

Using (1.51) and (1.27), this equals to

$$\left. \frac{\partial \check{p}}{\partial x} \right|_{y=0} = \left. \left(\frac{\partial \check{u}}{\partial t} - \frac{1}{Re} \frac{\partial \check{\omega}_z}{\partial y} + \check{u} \frac{\partial \hat{u}}{\partial x} \right) \right|_{y=0} . \tag{1.66}$$

So with $\check{u}|_{y=0} = 0$ and $\frac{\partial \check{\omega}_z}{\partial y} = 0|_{y=0} = 0$ we would eventually get

$$\left. \frac{\partial \check{p}}{\partial x} \right|_{y=0} = 0 \, , \tag{1.67}$$

but $\frac{\partial \check{v}}{\partial y}|_{y=0} = \frac{\partial \check{\omega}_z}{\partial y}|_{y=0} = 0$ only specifies

$$\left.\frac{\partial \check{p}}{\partial x}\right|_{y=0} = \left.\left(\frac{\partial \check{u}}{\partial t} + \check{u}\frac{\partial \hat{u}}{\partial x}\right)\right|_{y=0} , \tag{1.68}$$

as $\frac{\partial \check{v}}{\partial y}|_{y=0} = 0$ only delivers $\check{u}|_{y=0} = \text{const}$. This means that we could end up with a varying $\frac{\partial \check{p}}{\partial x}$ (contrary to 1.45) if $\check{u}|_{y=0} = 0$ is not satisified.

1.4. Boundary Layer Thickness

From conservation of mass (equation of continuity) it follows

$$U_\infty^* \delta^* = \int_0^\infty \left(U_\infty^* - U^*\right) \, \mathrm{d}y^* \tag{1.69}$$

and subsequently

$$\delta^* = \int_0^\infty \left(1 - \frac{U^*}{U_\infty^*}\right) \, \mathrm{d}y^* = \int_0^\infty (1 - U) \, \mathrm{d}y^* , \tag{1.70}$$

the displacement thickness, which is a function of x^*.

In analogy to the definition of the Blasius variable we can say

$$\delta_{[\eta]} = \frac{\delta^*}{\sqrt{\frac{2x^*\nu}{U_\infty^*}}} \tag{1.71}$$

and with

$$\frac{\mathrm{d}y^*}{\mathrm{d}\eta} = \sqrt{\frac{U_\infty^*}{2x^*\nu}} \tag{1.72}$$

we get

$$\delta^* = \int_0^\infty (1 - U) \sqrt{\frac{2x^*\nu}{U_\infty^*}} \, \mathrm{d}\eta \tag{1.73}$$

and

$$\delta_{[\eta]} = \int_0^\infty (1 - U) \, \mathrm{d}\eta , \tag{1.74}$$

the dimensionless displacement thickness in η-coordinates (while $\delta_{[y]} \equiv 1$, the displacement thickness in "classical" y-coordinates is equal to 1 because of the non-dimensionalisation chosen).

Introducing the streamfunction

$$\Psi := \sqrt{2x^*\nu U_\infty^*} f(\eta) \tag{1.75}$$

with f being a function that satisfies the BLASIUS formulation of boundary layer equations $ff''+f'''=0$ subject to the boundary conditions $f(0)=f'(0)=0$ and $\lim_{\eta\to\infty} f'(\eta) = 1$ we get, using

$$U^* = \frac{\partial \Psi}{\partial y} = U_\infty^* f'(\eta)\ , \tag{1.76}$$

$$U = f'(\eta)\ . \tag{1.77}$$

Thus we can write

$$\delta_{[\eta]} = \int_0^\infty \left(1 - f'\right)\, \mathrm{d}\eta\ , \tag{1.78}$$

and integrating this, we get

$$\delta_{[\eta]} = \lim_{\eta\to\infty} \left(\eta - f(\eta)\right) \approx 1.21678\ , \tag{1.79}$$

which is independent of x^*.

Therefore

$$\delta^*(x^*) \approx 1.72079\sqrt{\frac{x^*\nu}{U_\infty^*}} \tag{1.80}$$

and

$$\eta \approx 1.21678 y\ . \tag{1.81}$$

1.5. REYNOLDS Number Relations

$$Re_{x^*}(x^*) = \frac{U_\infty^* x^*}{\nu} \tag{1.82}$$

$$Re_{\delta^*}(x^*) = \frac{U_\infty^* \delta^*(x^*)}{\nu} = \frac{U_\infty^* \delta_{[\eta]}}{\nu}\sqrt{\frac{2x^*\nu}{U_\infty^*}} \tag{1.83}$$

$$= \sqrt{2}\,\delta_{[\eta]}\sqrt{Re_{x^*}} \tag{1.84}$$

Since we assume parallelism of the flow, the boundary layer thickness is to be considered constant over the whole flow and the REYNOLDS Number prescribed gives implicitly a distance between the point of consideration and the leading edge of the plate:

$$Re_{x^*}(x^*) = \frac{U^*_\infty x^*}{\nu} \tag{1.85}$$

$$Re_{\delta^*}(x^*) = \frac{U^*_\infty \delta^*(x^*)}{\nu} = \frac{U^*_\infty \delta_{[\eta]}}{\nu}\sqrt{\frac{2x^*\nu}{U^*_\infty}} \tag{1.86}$$

$$= \sqrt{2}\,\delta_{[\eta]}\sqrt{Re_{x^*}}\ . \tag{1.87}$$

1.6. Base Flow Calculation

In this study the base flow, as noted, can be of any type – provided it is parallel – and thus provisions are made for examining the stability behaviour of different basic flow types. To state it clearly, not the whole flow itself must be parallel but only the base flow; any non-parallel part can be coped with by the code via the perturbation variables (as this is the case for the trailing edge flow) – the result is the mean flow, cf. 1.1.1. The base flow can either be calculated in a previous step or specified explicitly.

The code used here provides a module that independently computes the base flow for the flat plate by solving the BLASIUS equations, which is the solution to PRANDTL's boundary layer equations with no streamwise pressure gradient. FIG. 1.2 (p. 21) gives the velocity profile over the flat plate produced by the `PCNAVWAKEBD` code. Likewise using the same interface to the main code, a new module, `PRESCMF`, was written that allowed to use any flow defined by an analytical function as the basic state.

1.7. Wake Flow and Prescribed Mean Flow

The altered boundary condition at the trailing edge of the plate (if it is within the domain) introduces a non-vanishing vorticity to the vorticity transport equation for positions downstream of the trailing edge and thereby provokes a "perturbation" flow. When adding this to the base flow, one gets the mean flow which equals the total flow in the wake of the plate, as it is depicted also in FIG. 1.2 (p. 21).

Alternatively to this, the `PRESCMF` wakeflow module (cf. 2.6) allows to remove the plate completely and have the whole computational domain represent the wake where the basic flow is chosen freely. This shall be called the prescribed mean flow further on (in our terminology it could also be called a prescribed base flow but obviously it will not be combined with a trailing edge).

Several analytic representations of the wake flow are discussed in the literature, like the

function $U(y) = 1 - Q\left(1 + \sinh^{2p}\left(y \sinh^{-1}(1)\right)\right)^{-1}$ proposed by Monkewitz (1988) or $U(y) = 1 - Q\,\mathrm{sech}^2(y)$ as used e. g. by Criminale, Jackson, and Joslin (2003). The profile parameter p determines the shape of the velocity profile and the velocity ratio Q the ratio between U^*_∞ and $U(y = 0)$. The latter profile could be considered a "standard" form and as both differ relatively little, we will concentrate on the sech^2 shaped representation for this thesis. FIG. 1.2 (p. 21) also shows its velocity profile. Its stability characteristics are discussed in 5.5.

The wake half-width, which is the y-coordinate where $U(\delta^*) = \frac{1}{2}\left(U^*_\infty + U(0)\right)$ is the non-dimensionalisation parameter δ^* for the variables, replacing the displacement thickness.

1.8. Stability of Fluid Flow

Stability of a dynamical system can be defined generally as the "ability of a dynamical system to be immune to small disturbances" (Criminale et al., 2003). More concrete, in the field of hydrodynamics it is the response of a laminar flow to a disturbance of small amplitude and there is the distinction: "if the flow returns to its original laminar state one defines the flow as stable, whereas if the disturbance grows and causes the laminar flow to change into a different state, one defines the flow as unstable" (Schmid & Henningson, 2001).

1.8.1. Modal Expansion

The classical approach to stability analysis of a flow is to make a FOURIER analysis of the disturbances, therefore to suppose the existence of harmonic partial oscillations, the normal modes, with the streamfunctions

$$\psi_n(x, y, t) = \phi_n(y)\, \mathrm{e}^{\mathrm{i}(\alpha_n x - \omega_n t)} \in \mathbb{C}\,, \tag{1.88}$$

which form a spectrum of modes and add together to the full disturbance via a FOURIER integral.

Variables of a wave used here are the phase velocity c, the wavenumber α, and the frequency ω, that are associated with each other and with the wavelength λ via the relations

$$c = \frac{\omega}{\alpha} \tag{1.89}$$

and

$$\alpha = \frac{2\pi}{\lambda}\,. \tag{1.90}$$

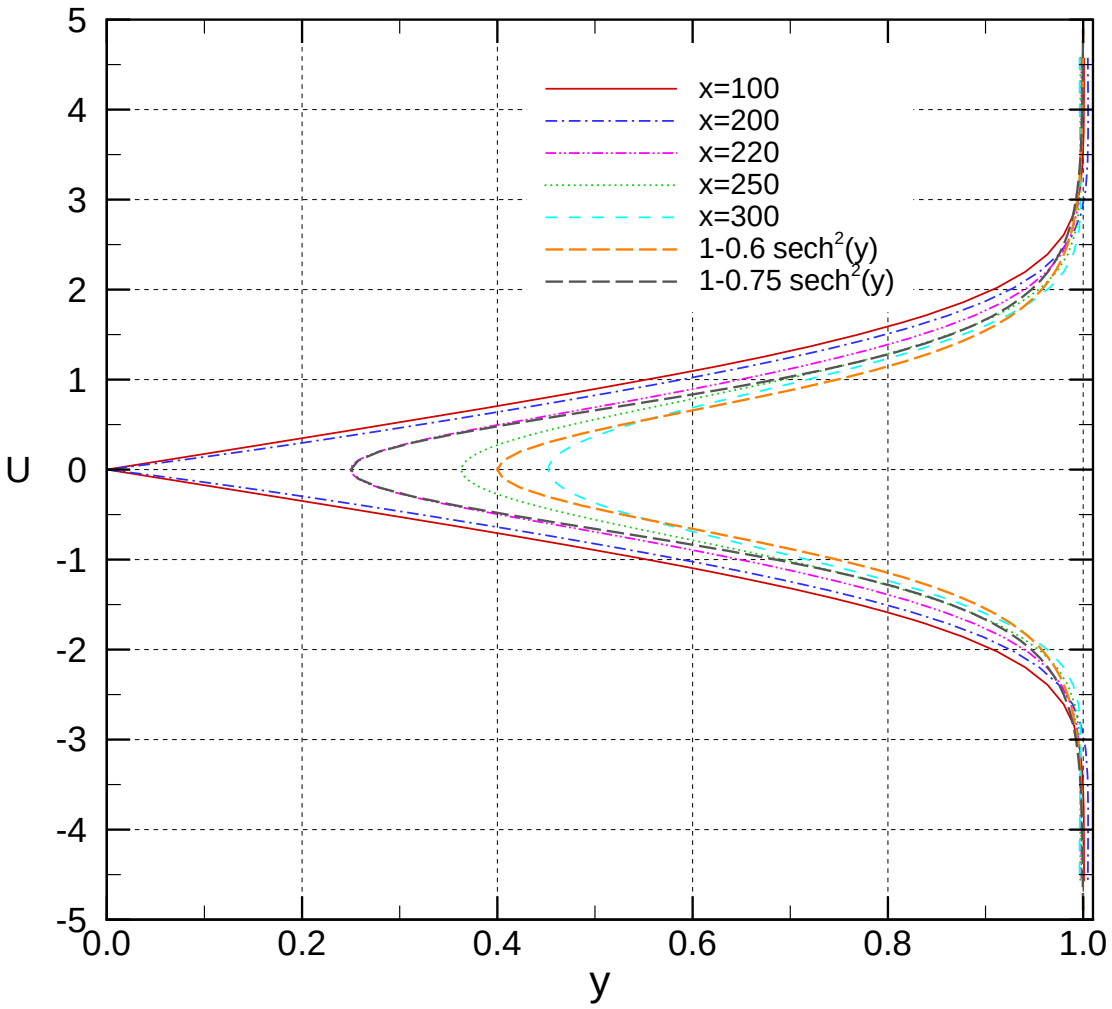

FIGURE 1.2. – *u-velocity profiles at various x-stations as resulting from simulation at* $Re = 1000$ *and with a trailing edge position of* $x_{TE} = 200$*, and as obtained with prescribed analytic mean flow.*

Furthermore, all these variables are complex

$$c = c_r + \mathrm{i}c_i \tag{1.91}$$

$$\alpha = \alpha_r + \mathrm{i}\alpha_i \tag{1.92}$$

$$\omega = \omega_r + \mathrm{i}\omega_i \ . \tag{1.93}$$

The complex eigenfunction $\phi_n(y)$ characterises the distribution of fluctuations in the y-direction. For spatial stability analysis (cf. 1.8.3) with $\omega_i = 0$ we can derive

$$c_r = \frac{\alpha_r \omega_r}{\alpha_r^2 + \alpha_i^2} \tag{1.94}$$

$$c_i = -\frac{\alpha_i \omega_r}{\alpha_r^2 + \alpha_i^2} \,, \tag{1.95}$$

and for the temporal stability problem ($\alpha_i = 0$)

$$c_r = \frac{\alpha_i \omega_i}{\alpha_r^2 + \alpha_i^2} \tag{1.96}$$

$$c_i = \frac{\alpha_r \omega_i}{\alpha_r^2 + \alpha_i^2} \,. \tag{1.97}$$

It should be noticed that there is a difference to the definition of the entirely real phase velocity

$$c_{ph} = \frac{\omega_r}{\alpha_r} \,. \tag{1.98}$$

1.8.2. Stability Analysis

When stating the existence of a streamfunction in this incompressible two-dimensional flow, applying its definition on (1.88) yields the complex forms (the subscript n is dropped for convenience):

$$u_c = \frac{\partial \psi}{\partial y} = \phi' \, \mathrm{e}^{\mathrm{i}(\alpha x - \omega t)} \tag{1.99}$$

$$v_c = -\frac{\partial \psi}{\partial x} = -\mathrm{i}\alpha\phi \, \mathrm{e}^{\mathrm{i}(\alpha x - \omega t)} \tag{1.100}$$

$$\omega_{z\,c} = \frac{\partial u_c}{\partial y} - \frac{\partial v_c}{\partial x} = -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\psi \tag{1.101}$$

$$= -\left(\phi'' - \alpha^2 \phi\right) \mathrm{e}^{\mathrm{i}(\alpha x - \omega t)} \,. \tag{1.102}$$

When seeking for an answer to whether a given disturbance exerts a destabilising effect on a fluid flow of given Reynolds number, the idea of the concept of normal mode disturbance forms becomes clear by solely considering the real, physically meaningful part of the perturbations as written above:

$$u = \mathrm{Re}\{u_c\} = \phi' \, \mathrm{e}^{-(\alpha_i x - \omega_i t)} \cos(\alpha_r x - \omega_r t) \tag{1.103}$$

$$v = \mathrm{Re}\{v_c\} = -\alpha\phi \, \mathrm{e}^{-(\alpha_i x - \omega_i t)} \sin(\alpha_r x - \omega_r t) \tag{1.104}$$

$$\omega_z = \mathrm{Re}\{\omega_{z\,c}\} = -\left(\phi'' - \alpha^2 \phi\right) \mathrm{e}^{-(\alpha_i x - \omega_i t)} \cos(\alpha_r x - \omega_r t) \tag{1.105}$$

It is evident that in any case where the disturbance exhibits an imaginary wave number $\alpha_i < 0$ and / or an imaginary frequency $\omega_i > 0$, the disturbance grows exponentially and inevitably leads to unbounded amplitudes of velocity and vorticity.

One approach to hydrodynamic stability analysis thus is to search for the neutral wave number $\alpha_i = 0$, respectively the neutral imaginary frequency $\omega_i = 0$, which constitute

the boundary between stable and unstable disturbances for a flow of given parameters.

One way to do this is to substitute the oscillation form of the disturbances (1.99)–(1.101) into the NAVIER-STOKES equations (1.23). With neglecting the nonlinear terms (products of small quantities) and eliminating the pressure this gives

$$(U-c)\left(\phi''-\alpha^2\right)-U''\phi=-\frac{\mathrm{i}}{\alpha Re}\left(\phi''''-2\alpha^2\phi''+\alpha^4\phi\right)\ , \tag{1.106}$$

the ORR-SOMMERFELD equation, the fundamental differential equation to hydrodynamic stability analysis. Boundary conditions for a symmetric free flow as the wake are

$$\phi'(y=0)=\phi'''(y=0)=0 \qquad \text{or} \qquad \phi(y=0)=\phi''(y=0)=0 \tag{1.107}$$

and

$$\lim_{y\to\infty}\phi(y)=\lim_{y\to\infty}\phi'(y)=0\ . \tag{1.108}$$

The solution of the ORR-SOMMERFELD equation, which is the governing equation for every normal mode, is an eigenvalue problem, providing an eigenfunction ϕ for each complex pair of α and ω for a given Re (ω replaces, together with the initially unknown α, c according to (1.89)).

1.8.3. Spatial and Temporal Stability

Two principal points of view can be taken when examining the stability properties of a fluid flow: (a) prescribing an initial streamwise homogeneous perturbation of wavenumber α with $\alpha_i = 0$ and inspecting the development of the response in time for any point, or (b) prescribing a temporally constant disturbance of frequency ω with $\omega_i = 0$ in a fixed location and considering the evolution of the disturbance travelling downstream from this location. These cases correspond to temporal and spatial stability analysis respectively.

Historically, stability analysis focused mostly on temporal stability as the analytical approach to this case is notably less complicated. Yet what is observed physically in experiment conducted with fluid flow is mostly spatial stability or rather instability. This equally holds for experiments conducted in the synthetic environment of a numerical simulation and thus the temporal stability analysis originally was the primary concern in this thesis.

2. Numerical Approach

In the following sections, the numerical approach and the solution scheme of the `PCNAVWAKEBD` *code – for the case it is used for the simulation of the flow – are briefly reviewed. Furthermore, changes and extensions to this code as well as separate programmes written for post-processing the data obtained from running* `PCNAVWAKEBD` *are presented.*

2.1. Discretisation

2.1.1. Streamwise Discretisation and Timestepping

In order to run the numerical simulation, a constant discretisation of the time as Δt is applied. The streamwise direction also is discretised equidistantly with a step width of Δx.

2.1.2. Wall-Normal Discretisation

Coordinate Mapping

In the wall-normal direction y a coordinate mapping

$$\zeta = \frac{l}{l+y} \,, \tag{2.1}$$

involving the free stretching parameter l and mapping the range $[0, \infty)$ on $(0, 1]$ is applied. This mapping gives both the advantage of a non-equidistant grid with highest resolution near the wall / centreline, where gradients are highest, and provides a pseudo-infinite domain with the outmost grid point arbitrarily far out. This is particularly interesting because the upper deck flow (cf. chapter 4) shall be resolved and this structure lies in the potential flow region well above the boundary layer. Moreover, the grid points are spaced via a cos function

$$\vartheta_j = \pi \frac{j-1}{2N_y} \tag{2.2}$$

$$\zeta = \cos\vartheta \,, \tag{2.3}$$

so that in result

$$y_j = \frac{l}{\cos\left(\pi \frac{j-1}{2N_y}\right)} - l\,. \tag{2.4}$$

Another advantage of the coordinate mapping lies within in the behaviour of the wall-normal derivative of this mapped coordinate:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial \zeta}\frac{\partial \zeta}{\partial y} \tag{2.5a}$$

$$= -\frac{\partial f}{\partial \zeta}\frac{l}{(l+y)^2} \tag{2.5b}$$

$$= -\frac{\partial f}{\partial \zeta}\frac{\zeta^2}{l}\,, \tag{2.5c}$$

and as $\zeta \to 0$ when $y \to \infty$, this assures

$$\lim_{y\to\infty}\left\{-\frac{\partial f}{\partial \zeta}\frac{\zeta^2}{l}\right\} = 0 \tag{2.6}$$

in the mapped domain, as long as $\lim_{y\to\infty} f$ is bounded in the physical domain. This ensures the global satisfaction of (1.38).

CHEBYSHEV Representation

The discretisation of the wall-normal coordinate is achieved via odd (primary variables) or even (secondary variable) CHEBYSHEV polynomials representing the values at discrete points. The use of only the odd or even terms respectively is because only half the CHEBYSHEV interval is "used" by the semi-infinite physical domain. The discretisation reads like

$$v(x,y,t) = \sum_{m=1}^{N_y} v_m(x,t) T_{2m-1}(\zeta) \tag{2.7}$$

$$\omega_z(x,y,t) = \sum_{m=1}^{N_y} \omega_{z\,m}(x,t) T_{2m-1}(\zeta) \tag{2.8}$$

$$u(x,y,t) = \sum_{m=1}^{N_y} u_m(x,t)\left(T_{2m}(\zeta) - T_{2m}(0)\right)\,, \tag{2.9}$$

where the last term, derived from the zeroth-order coefficient, is there to ensure that u vanishes for $y \to \infty$

N_y is the truncation length of the series and thus the order of the CHEBYSHEV polynomial.

When substituting the CHEBYSHEV expansions into the definition for the secondary variable (1.27), a relationship between the coefficients of the primary and the secondary variables is obtained and thus for determining u it is not necessary any longer to go via the physical values.

2.2. Solution Scheme of the PCNAVWAKEBD Code

For the numerical solution of the vorticity transport equation (1.25), it is split into the time and second order y-derivative of the vorticity on the one hand and an accumulative quantity for the rest on the other hand.

$$\frac{\partial \omega_z}{\partial t} - \frac{1}{Re}\frac{\partial^2 \omega_z}{\partial y^2} = N \ , \tag{2.10}$$

where

$$N = -\left(U+u\right)\frac{\partial \omega_z}{\partial x} - v\left(\frac{\partial \omega_z}{\partial y} + \frac{\mathrm{d}^2 U}{\mathrm{d}y^2}\right) + \frac{1}{Re}\frac{\partial^2 \omega_z}{\partial x^2} + \frac{\partial f_b}{\partial x} \tag{2.11}$$

In the time loop over the index l from $t = 0$ to $t = l\Delta t$, a predictor-corrector scheme in time is employed for the left hand side of (2.10) and the right hand side N is treated explicitly with a backward difference scheme. The inclusion of the x-diffusion term in the explicitly solved part is due to higher robustness of this method without the price of high computational demand as it would be the case for the y-diffusion term because the latter is discretised in a CHEBYSHEV series; refer to Davies and Carpenter (2001) for further details.

2.2.1. Predictor Step

For the predictor step, a three-point second order backward difference scheme for the time derivative is used

$$\left(\frac{\partial \omega_z}{\partial t}\right)^l = \frac{1}{2\Delta t}\left(3\tilde{\omega}_z^l - 4\omega_z^{l-1} + \omega_z^{l-2}\right) \tag{2.12}$$

and this, together with the explicit approximation terms for N, turns (2.10) into

$$\left(\frac{3}{2\Delta t} - \frac{1}{Re}\frac{\partial^2}{\partial y^2}\right)\tilde{\omega}_z^l = 2N^{l-1} - N^{l-2} + \frac{1}{2\Delta t}\left(4\omega_z^{l-1} - \omega_z^{l-2}\right) \ , \tag{2.13}$$

a second order ODE to be solved in a spatial loop for every x-wise point k subject either to the symmetry condition $\tilde{\omega}_z^l|_{y=0} = -U'|_{y=0}$ (1.58c) or to the no-slip condition (1.57), which itself is approximated using a backward difference scheme:

$$\int_0^\infty \tilde{\omega}_z^l \,\mathrm{d}y = -\frac{\partial}{\partial x} \int_0^\infty 2v^{l-1} - v^{l-2} \,\mathrm{d}y \;. \tag{2.14}$$

To this we reapply the no-slip condition in terms of previous values $\int_0^\infty \tilde{\omega}_z^{l-n} \,\mathrm{d}y = -\frac{\partial}{\partial x} \int_0^\infty v^{l-n} \,\mathrm{d}y$ and get

$$\int_0^\infty \tilde{\omega}_z^l \,\mathrm{d}y = \int_0^\infty 2\omega_z^{l-1} - \omega_z^{l-2} \,\mathrm{d}y \;. \tag{2.15}$$

2.2.2. Solution of the Poisson Equation

The predictor value is used to solve the Poisson equation (1.26)

$$\frac{\partial^2 v^l}{\partial x^2} + \frac{\partial^2 v^l}{\partial y^2} = -\frac{\partial \tilde{\omega}_z^l}{\partial x} \;, \tag{2.16}$$

and a second order central second difference scheme approximates the second derivative in space

$$\frac{v_{k+1} - 2v_k + v_{k-1}}{(\Delta x)^2} + \frac{\partial^2 v^l}{\partial y^2} = -\frac{\partial \tilde{\omega}_z^l}{\partial x} \;. \tag{2.17}$$

For the solution of this equation, two different approaches can be envisaged, either an iterative scheme or a direct solution involving a sin transform.

Iterative Scheme

The iteration in (2.17) over i is performed with an "inner" spatial loop over k, therefore using the unknown value of v on the downstream position $k+1$ from the previous iteration step:

$$\frac{\partial^2 \breve{v}_k^{l,i}}{\partial y^2} - \frac{2\breve{v}_k^{l,i}}{(\Delta x)^2} = -\frac{v_{k+1}^{l,i-1} + v_{k-1}^{l,i}}{(\Delta x)^2} - \frac{\partial \tilde{\omega}_{z\,k}^l}{\partial x} \;. \tag{2.18}$$

The result $\breve{v}_k^{l,i}$ may be used for the next iteration and marching step ($v_k^{l,i} = \breve{v}_k^{l,i}$), but alternatively, an extrapolation using the previous value

$$v_k^{l,i} = v_k^{l,i-1} + \varrho \left(\breve{v}_k^{l,i} - v_k^{l,i-1} \right) \tag{2.19}$$

can be applied to get an improved value for subsequent calculation steps. Typically, useful values for ϱ lie between 0.5 and 1.5, where $\varrho < 1$ is called under-relaxation and $\varrho > 1$ over-relaxation.

The influence of the number of iterations per timestep N_{it} will be discussed in 3.1.4.

Direct Solution

Alternatively to the iteration procedure, a direct solution of equation (2.17) can be achieved via a sin transformation that is defined as

$$\bar{v}_n := \mathcal{F}\{v_n\} = \sum_{k=1}^{N_x-1} v_k \sin\left(\frac{kn\pi}{N_x}\right) \ . \tag{2.20}$$

For the inclusion of the boundary points $k = 0$ and $k = N_x - 1$ (whose contributions to the sum above vanish) particular provisions have to be made, details on that can be found in Heaney (2007).

The transform of the complete discretisation term then readily reduces, by use of periodicity and sum formulae, to a multiple of the centre value

$$\mathcal{F}\left\{\frac{v_{k+1} - 2v_k + v_{k-1}}{(\Delta x)^2}\right\} = \beta_n \mathcal{F}\{v_k\} \ , \tag{2.21}$$

which turns the Poisson equation into a second order ODE in terms of the transformed variable:

$$\frac{\partial^2}{\partial y^2}\bar{v}_n + \beta_n \bar{v}_n = \mathcal{F}\left\{-\frac{\partial \tilde{\omega}_{z\,k}^l}{\partial x}\right\} \ . \tag{2.22}$$

After solution for $\bar{v}_n$, an inverse transform gives v_k.

Without further precautions – which have not been considered – this direct solution cannot be applied to the case of a flow that comprises at the same time a finite flat plate and a trailing edge plus wake flow because the series form directly involves every streamwise point and requires homogenous boundary conditions over the whole domain.

2.2.3. Corrector Step

The corrector step for ω_z^l, again looped over k, is then accomplished by using the values of $\tilde{\omega}_z^l$ and v^l obtained in the preceding steps

$$\frac{1}{2\Delta t}\left(3\omega_z^l - 4\omega_z^{l-1} + \omega_z^{l-2}\right) - \frac{1}{Re}\frac{\partial^2 \omega_z^l}{\partial y^2} = N^l\Big|_{\tilde{\omega}_z^l, v^l} \ , \tag{2.23}$$

which then can be solved subject to

$$\int_0^\infty \omega_z^l \,\mathrm{d}y = -\frac{\partial}{\partial x} \int_0^\infty v^l \,\mathrm{d}y \tag{2.24}$$

or

$$\omega_z^l|_{y=0} = -U'|_{y=0} \tag{2.25}$$

respectively.

2.3. Introduction of Disturbances

The introduction of disturbances is done by a body force exerted on the antisymmetric part of the flow in the y-direction. While the `PCNAVWAKE` code in principle allows for any type of disturbances, it is chosen to be periodic in this work. As an abrupt rise in disturbance amplitude would introduce a wide range of frequencies (the FOURIER series representation of a rectangular signal having an infinite number of terms), the disturbance is "faded" in and out over time and space. Time-wise the forcing ramps up with

$$r_t(t) = 1 - \mathrm{e}^{-\frac{1}{2}\omega t} \,, \tag{2.26}$$

and over streamwise space it is multiplied by

$$r_x(x) = \frac{x - x_f}{l_{xf}} \,\mathrm{e}^{-(\frac{x-x_f}{l_{xf}})^2} \,, \tag{2.27}$$

resulting in a GAUSSIAN curve shaped distribution over x, centred around x_f. The scaling factor l_{xf} is introduced to adapt the width of this bell curve to be able to remove possible spurious effects.

The wall normal distribution is chosen such that the integral of the forcing over y vanishes for every x and, for the simulations carried out here, the forcing position was left symmetrically on the centreline:

$$r_y(y) = 2\left(1 - 2\left(\frac{y}{l_{yf}}\right)^2\right) \mathrm{e}^{-(\frac{y}{l_{yf}})^2} \,. \tag{2.28}$$

The forcing amplitude has the dimension of a force per unit mass ($[\frac{\mathrm{N}}{\mathrm{kg}}] = [\frac{\mathrm{m}}{\mathrm{s}^2}]$) and is non-dimensionalised according to

$$A_f = \frac{A_f^* \delta^*}{U_\infty^{*2}} \,. \tag{2.29}$$

In sum, the forcing is

$$f_b(x, y, t) = A_f \, r_x(x) \, r_y(y) \, r_t(t) \sin(\omega t) \; . \tag{2.30}$$

The body force f_b would be simply added to the right hand side of the y-momentum equation. For the velocity-vorticity form employed here, the curl of this forcing is taken und this results in adding $\frac{\partial f_b}{\partial x}$ as an antisymmetric part to (1.25).

2.4. Symmetric and Antisymmetric Decomposition

The symmetric and antisymmetric components are conveniently stored as the real and imaginary part of complex variables. The advantage is that the whole domain is represented by one semi-infinite domain carrying two sets of variables for each point. Additions can then be made straightforwardly, multiplications have to be done separately for real and imaginary part, and we have two separate and independent sets of boundary conditions as discussed in 1.3.

2.5. Inflow and Outflow Conditions

Apart from the boundary conditions for $y = 0$ and $y \to \infty$, stemming directly from the physical boundaries of the flow and having been discussed above, the fact that the numerical simulation takes place within a domain bounded also in x-direction necessitates boundary conditions at $x = 0$ and $x = x_{end}$.

The velocity-vorticity formulation of the NAVIER-STOKES equation does not require any special condition for the upstream boundary and the perturbations under consideration are expected to only weakly propagate upstream, thus the inflow condition simply is $u(x = 0) = v(x = 0) = \omega_z(x = 0) = 0$.

The downstream side is more delicate because the artificial value specified here can be convected upstream and cause spurious phenomena, e. g. upstream reflection of the disturbances travelling downstream at the "fixed end" of the domain. For the outflow condition there are several alternatives, three of which are implemented in the code and could be used (for further details on these conditions see Kloker, Konzelmann, and Fasel (1993) and Fasel, Rist, and Konzelmann (1990)):

- **Second order oscillatory outflow condition**
  The perturbations arriving at the downstream boundary can be expected to be of oscillatory form ($v, \omega_z \propto \mathrm{e}^{\mathrm{i}\alpha x}$) as the disturbances introduced are oscillating – at least as long as the disturbance frequency is not too high above the frequency for the most unstable mode. Therefore, a second order wave condition, obtained by

deriving that form two times with respect to x, lets the disturbance waves flow out without them being reflected:

$$\frac{\partial^2 v}{\partial x^2} = -\alpha^2 v \tag{2.31}$$

$$\frac{\partial^2 \omega_z}{\partial x^2} = -\alpha^2 \omega_z \, . \tag{2.32}$$

- **First order wave equation**
  Alternatively, a complete first order wave equation

  $$\frac{\partial v}{\partial t} + U\frac{\partial v}{\partial x} = 0 \tag{2.33}$$

  can likewise provide a convecting downstream boundary.

- **Buffer domain**
  Another possibility to avoid artificial distortions caused by the outflow is to "smooth out" the variables over a finite buffer domain of a certain length. Within this buffer between $x = x_{end} - l_B$ and $x = x_{end}$ the value of ω_z is replaced at the end of each time step as

  $$\omega_z \mapsto m(x)\,\omega_z \tag{2.34}$$

  with

  $$m(x) = \frac{1}{2}\left(1 - \tanh\left(\sinh(e\frac{x - x_{end}}{l_B} + g)\right)\right) \in [0, 1] \, , \tag{2.35}$$

  ramping down from ω_z to 0. e and g are parameters to adjust the distribution of m. Values of 8 and 4 respectively, together with $l_B = 80$, showed to give a reasonable transition with the significant decrease over a centred half of the buffer length and margin to both buffer boundaries, see FIG. 2.1 (p. 32).

2.6. PRESCMF Wakeflow Module

A module complementing the PCNAVWAKE code was written to allow for prescribing arbitrary analytic wake flow profiles. Any kind of function giving the y-distribution of U, the x-velocity component, can be specified. The velocity is set to have no x-wise variation and, consistently with the global setup, parallel flow is assumed ($V = 0$).

The code of this module can be found in A.1.

The PRESCMF wakeflow module is docked on the main code via the latter's interface originally created for retrieving the BLASIUS boundary layer base flow. Variables handed

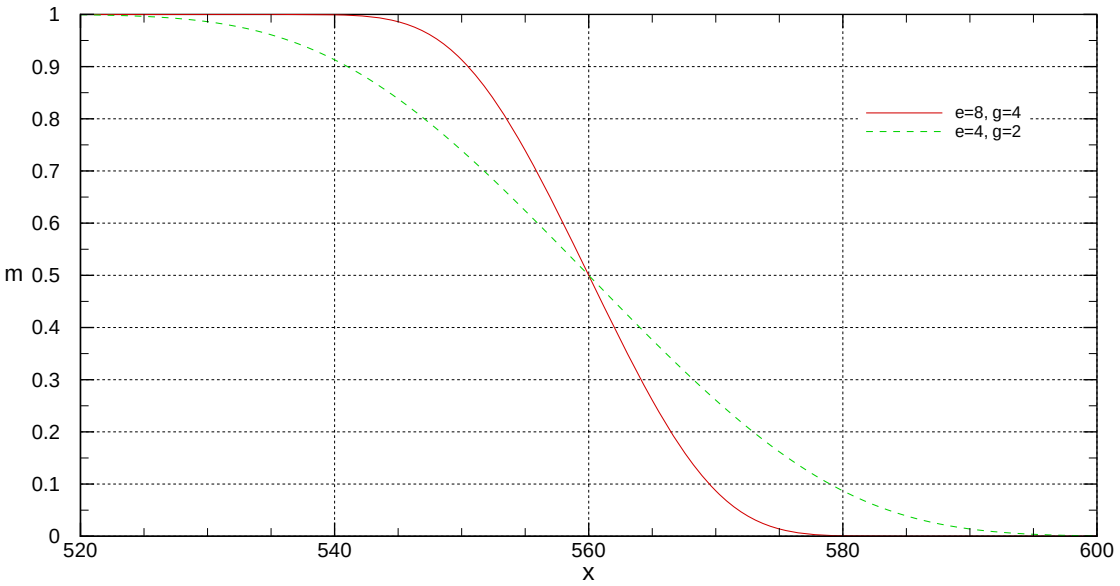

FIGURE 2.1. – *Distribution of the buffer function $m(x)$ for two sets of adjusting parameters at $l_B = 80$ and $x_{end} = 600$.*

over to the subroutine were the mapping constant l^*, the number of y-wise points N_y and the profile parameter p, as well as the velocity ratio Q. The data returned by the module are U, $\frac{\mathrm{d}^2U}{\mathrm{d}y^2}$, $\frac{\mathrm{d}U}{\mathrm{d}y}|_{y=0}$ and l (therefore implicitly δ^* because $l = \frac{l^*}{\delta^*}$).

The module code had to be written, tested and the problem-free integration and interaction with the main code had to be verified.

2.7. Other Extensions and Changes to the PCNAVWAKE Programme Code

A wide number of changes and amendments were applied to the `PCNAVWAKE` code to optimise and adapt its behaviour at input and output and the platform compatibility. All changes were written, tested and verified for problem-free integration and interaction with the main code.

In detail, the following work was achieved:

- The whole data output interface was rewritten to have a more structured set of output files and a clearer data arrangement in these files at hand. The original subroutines `outputdata` and `timehist` are replaced by the newly written subroutines `outputall` and `outputcent`. Module `outputall` writes the values for the symmetric and asymmetric part of the field to the file `sym-asym.dat` and for the

total values to the file `total.dat`. Variables written to `sym-asym.dat` are (in that order) x, $1-\zeta$, y, $\hat{u}$, $\hat{v}$, $\hat{\omega}_z$, $\check{u}$, $\check{v}$, $\check{\omega}_z$, $\hat{u}_{pert}$, $\hat{v}_{pert}$, $\hat{\omega}_{z\,pert}$, $\check{u}_{pert}$, $\check{v}_{pert}$, $\check{\omega}_{z\,pert}$, $\hat{p}$, $\check{p}$ and to `total.dat` x, $1-\zeta$, y, u, v, ω_z, u_{pert}, v_{pert}, $\omega_{z\,pert}$, U^t, $\frac{\mathrm{d}^2U^t}{\mathrm{d}y^2}$, p. $\cdot_{pert}$ means the difference to the base flow:

$$a_{pert} = a - A\,. \tag{2.36}$$

The printout of snapshots of the flow for subsequent points in time was added with the subroutine `outputcent`. It writes the values of x, $|u|$, $|v|$, $|\omega_z|$, u, v, ω_z on the centreline ($y = 0$) to files called `centvt`*nnnn*`.dat` with *nnnn* the current time t, when needed preceded by zeros to fill up four digits. An adapted command calls this module at every integer time step. Together with an appropriate visualisation tool (e. g. the respective command of `Tecplot`), animations could be generated to illustrate the development of the flow in a short film sequence.

The code of the new output module is listed in A.3.1.

- It was considered useful to also allow for the visualisation of the data with the programme `Tecplot` in addition to the `gnuplot` compatibility foreseen by the original `PCNAVWAKE` code, therefore the output data format and all file headers were adapted such that they can be used for either one of these programmes. Via the new start-up parameter `TP`, which is 1 for `Tecplot` compatible output format and 0 for `gnuplot` compatibility, the desired format can be chosen before running the simulations. The routine that reads in the start-up parameters from `input.dat` was amended accordingly. All output routines, also the ones obsolete when using the new output interface mentioned above, were rewritten to read this parameter and to format and write data either in `Tecplot` or `gnuplot` format.

- All occurrences of the `print*,`... commands were replaced by `write(...,*) '...'` because piping of screen output to a file specified on the command line at programme call is not allowed on the Condor cluster used for running the simulations (cf. 3.2).

- The data input interface was completely rearranged to make it more structured, integrate the above change towards `write` commands and make the status messaging more handy. A file called `outputparam.dat` is created by the code, where all used input parameters are summed up and printed. Furthermore the need to specify the input file `input.dat` explicitly at programme start from the command line was superseded by an automatic integration. The subroutines `flowparams`, `miscdata` and `otherdata` were replaced altogether by a single subroutine `startupparams` for which the code is given in A.3.2.

2.8. Post-processing Programmes

Several smaller programmes were written to post-process the data obtained from simulations for further analysis and visualisation. The two most important ones are:

- For achieving the triple-deck scalings involving the REYNOLDS number, a comprehensive programme to process the data was written. This programme, called `POSTPROCESSING` and shown as source code in A.4, consists of several subroutines and modules that read in the flow parameters from the principal input file `input.dat` and the parameters of the scaling to be applied from the post-processing input file `ppinput.dat`. From the latter it reads in the parameters shown and explained in TAB. 2.1. The programme allows – amongst others – to set the origin, to specify the variables to be scaled and the exponents to the REYNOLDS number to be used for the scaling. It then calculates these scalings, prints out the results to `resc.dat` and creates a customised file for launching `gnuplot` with the appropriate parameters.
- The programme `GATHERING` assembles the `centvt`*nnnn*`.dat` files giving the centreline values at points in time into one file, thus creating a t-x-field spanning over the whole length and the complete simulation time. This field, when visualised, was considered to be useful for tracing the origin of the numerical instabilities observed. The programme code is listed in A.5.

```
1  0       ADDMF (Add base flow field to perturbation field)
2  1       ORIGIN (Set x-origin to: 0: inflow, 1: TE, 2: LE)
3  1       RESCX (Scale x on Reynolds number)
4  0       ETAZETASC (Scale on 0=unchanged, 1=y, 2=eta, 3=zeta)
5  0       PLOT (Plot via gnuplot)
6  1       RESCY (Scale zeta/y/eta on Reynolds number)
7  1       RESCUV (Scale u and v on Reynolds number)
8  -0.25   XSCE (Exponent to Reynolds number when scaling x)
9  0.25    YSCE (Exponent to Reynolds number when scaling y)
10 0.25    USCE (Exponent to Reynolds number when scaling u)
11 0.125   VSCE (Exponent to Reynolds number when scaling v)
12 0.5     VORSCE (Exponent to Reynolds number when scaling p)
```

TABLE 2.1. – *Exemplary* `ppinput.dat` *file.*

3. Realisation of Simulations

The following chapter shows under which settings the simulations were actually carried out. The individual influence of the numerical parameters would be of highest interest if a stable and meaningful simulation would be possible – however, as this is not achieved, they show which parameters are not at the root of the artificial instability. Additionally, a method how the wavenumber of the perturbation could have been extracted from the flow is presented, together with re-dimensionalisations that give an impression of what the physical values of the results would be.

3.1. Choice of Numerical Parameters

Numerical behaviour, in particular numerical stability of the solution (in contrast to the physical stability in the focus of this thesis), is a major concern when running computer simulations. The code used had evolved over long time and – in its latest evolution taken as a basis for this research – had been optimized for boundary layer flow over a fixed surface plus the particular case of flow around the trailing edge, where some simulations were made previously. However, for the case of wake flow, including the newly added feature of an arbitrary prescribed mean flow, no experience existed to rely on. Considerable efforts were therefore taken on the choice of numerical parameters of the code to gain usable results.

The obstacle to this was that global oscillations of the whole domain in the form of a standing wave with increasing wavelength occurred as soon as any minimal body forcing was applied – both for the trailing edge flow and the prescribed wake flow, at all meaningful Reynolds numbers and forcing frequencies. The supposed standing wave shape of this hinted to an oscillation emerging at the inflow and being reflected by the outflow "fixed end" of the domain. Its amplitude growth was such that it outweighed the physical phenomena within short time, in Fig. 3.1 (p. 37) through Fig. 3.3 (p. 39) these domain oscillations can be seen as wave "humps" for higher points in time[2].

To supplement the search for principal errors in the code, which was done as far as the

[2] The frequency of this oscillation was relatively insensitive to the parameters discussed below for lower amplitudes but became variable for higher amplitudes. To avoid that a difference in momentary amplitude depicted for a certain time is due to a difference in phase instead of a difference in maximum amplitude for different parameter settings, only time shots where the amplitude of this oscillation remained moderate are given in the following graphs.

insight into the code allowed, an empirical approach should help to find out whether any of the parameters determining the functioning or the code contributed to or was causing the instability. Accordingly, an immense set of simulations with several hundreds of permutations of the numerical parameters was performed to find the optimal configuration. Below, the individual contributions of each parameter involved are discussed.

3.1.1. Stream-wise Grid Step Size and Time Step Size

Stream-wise grid step size Δx and time step length Δt obviously had an influence on the stability of the simulation. With finer resolution of the numerical discretisation – while observing the Courant–Friedrichs–Lewy condition $\Delta x > \Delta t$, as the convective velocity is in the order of 1 – other spurious effects could be eliminated but, unfortunately, refining the simulation did not lead to complete stability and the global oscillations remained.

The optimum values, keeping computing time moderate, were $\Delta x = 1$ and $\Delta t = 0.5$.

3.1.2. Domain Length and Forcing Location

Additionally to the instability described above, a high-frequency oscillation forming directly behind the inflow, spanning over a length of about 50 and decaying then could be observed initially. In Fig. 3.1 (p. 37) the graphs for $t = 150$ reveal this oscillation. This was relatively simply resolved by placing the location where the body forcing is introduced sufficiently far downstream, together with a greater total domain length to minimize downstream effects. The amplitude of this noise then remained limited to about six magnitudes less than the actual flow effects and these disturbances posed no further concern.

3.1.3. Forcing Length Scale and Mapping Constant

Neither the scaling factor to the forcing in x-direction l_{xf} nor the one for y-direction l_{yf} showed any significance for the problem. They were set constantly to 10 and 0.5, respectively.

The same is true for l; the mapping constant used to map y onto ζ did not show to have an influence on the numerical instability. The value of l was fixed at 4 to get a good balance between domain width and resolution around the centreline.

3.1.4. Number of Iterations per Time Step

Interestingly, the flow seemed to be more susceptible for the domain oscillation mentioned above with a higher number of iterations per time step as could be seen in Fig. 3.1 (p.

37). There was a dependence from grid and time resolution but comparison revealed an optimum at the value of 5 when Δx and Δt chosen as above.

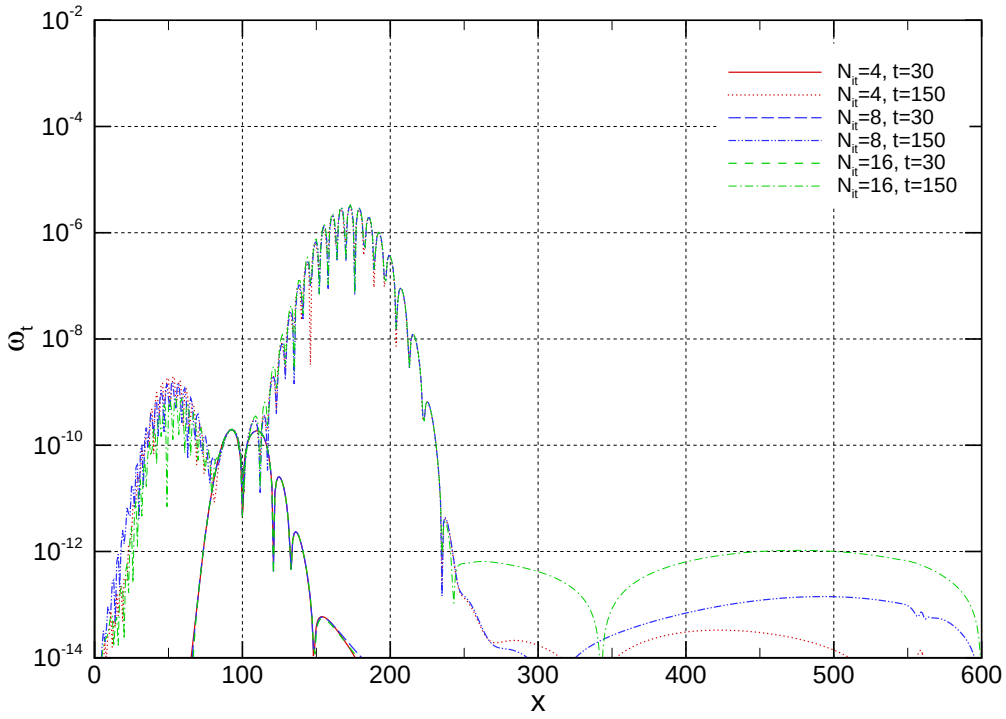

FIGURE 3.1. – *Absolute value of the perturbation vorticity versus streamwise coordinate at different times and for different numbers of iterations per time step for* $\Delta x = 1.0$, $\Delta t = 0.5$, $N_y = 64$, $Re = 1000$, $Q = 0.9$, $\omega = 0.4$.

3.1.5. Order of CHEBYSHEV Polynomials

Normal grid resolution, determined by the order of the CHEBYSHEV polynomial discretising the variables in y-direction, showed to have an ambiguous influence on the amplitude of the global domain oscillation.

From FIG. 3.2 (p. 38) it is obvious that an order of 32 leads to considerably higher parasitic oscillations than an order of 64. FIG. 3.3 (p. 39) in contrast, reveals an inverse relationship when a high number of iterations per time step are used. Since we decided for 5 iterations per time step, the higher normal grid resolution was chosen.

3.1.6. Outflow Condition

As the whole domain oscillation seemed like a problem involving the boundary conditions, trying several outflow conditions seemed to be helpful. For the downstream border of

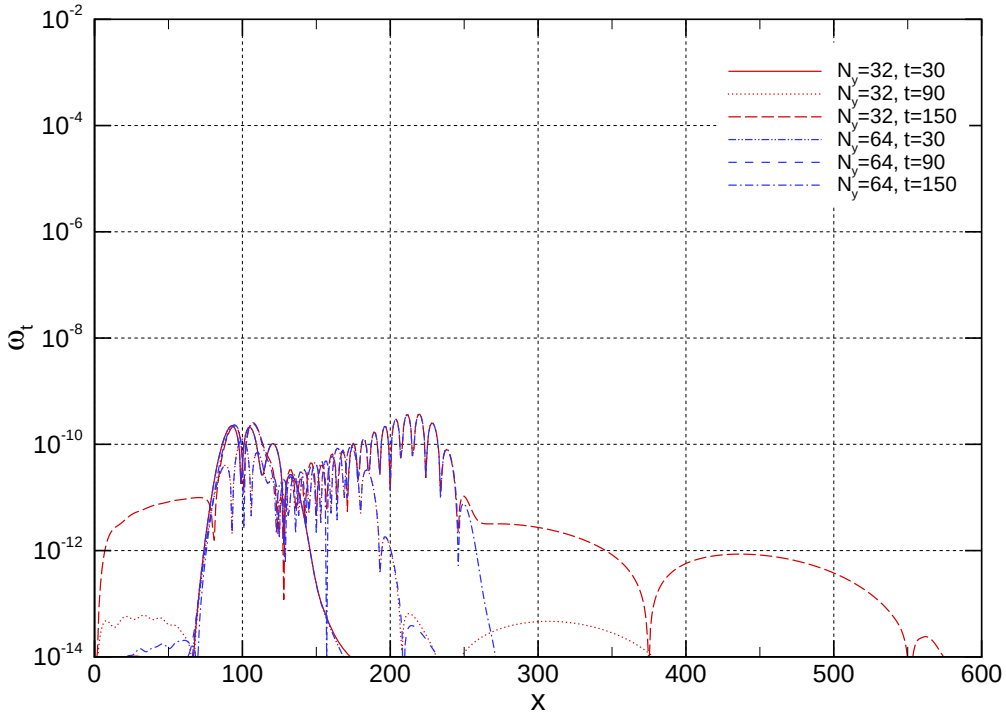

FIGURE 3.2. – *Absolute value of the perturbation vorticity versus streamwise coordinate at different times and for two orders of* CHEBYSHEV *polynomials for* $\Delta x = 1.0$, $\Delta t = 0.5$, $N_{it} = 4$, $Re = 1000$, $Q = 0.3$, $\omega = 0.4$.

the domain the two possibilities discussed in 2.5, together and without a buffer domain at outflow end were explored. Anyhow, no change to the problem could be found.

3.1.7. Relaxation Parameter and Initial Value for Next Time Step

If the iterative solution of the implicit central difference scheme (cf. 2.2.2) is chosen, then a starting value for $v_k^{l,0}$ can be derived from the previous time step. The parameter for the relaxation applied on the POISSON solver for the normal velocity is ϱ; it had no influence on the stability problem.

3.1.8. Forcing Amplitude

The forcing amplitude A_f is directly influencing the disturbance amplitude but the numerical instability was not dependent on this at all. For the simulations of which the results are shown in this chapter, a value of 10^{-10} was chosen.

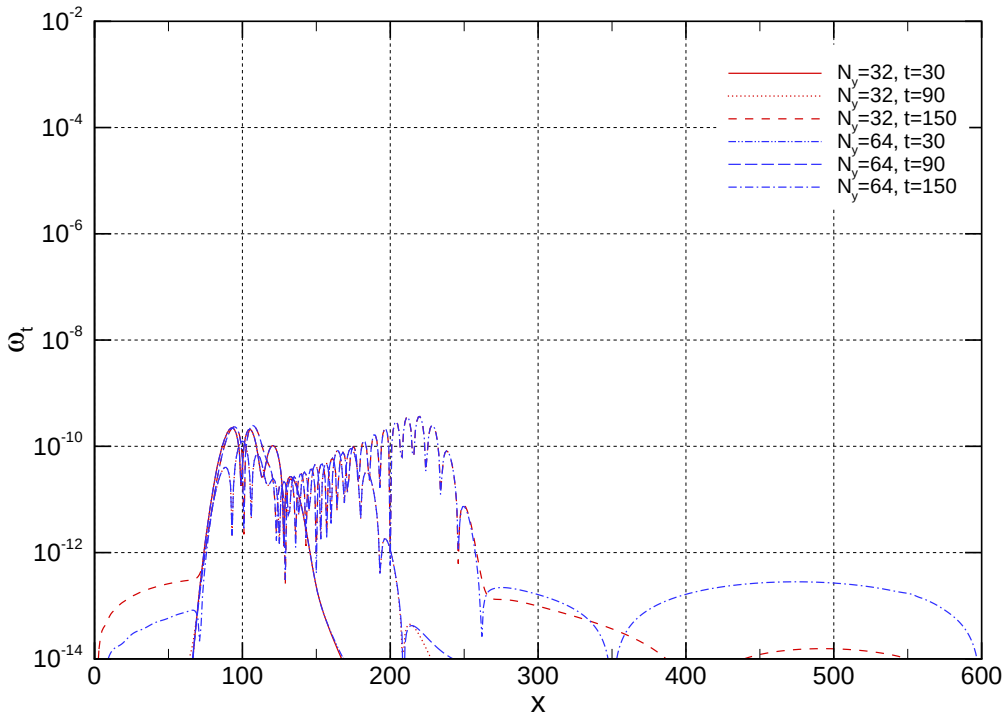

FIGURE 3.3. – *Absolute value of the perturbation vorticity versus streamwise coordinate at different times and for two orders of* CHEBYSHEV *polynomials for* $N_{it} = 16$*, other parameters as above.*

3.2. Implementation for Grid Computing

For the execution of the massive number of simulations, a computing grid of the Welsh e-Science Centre (WeSC) comprising four 8 processor SGI Origin 300 machines was at disposal. A load balancing and job distribution software called Condor could be charged with a batch of simulations to be performed independently and using the considerable computing resources of the cluster in coordination with jobs sent by other users. The gain of running the simulations on this WeSC Condor grid was a execution time reduced by about a factor of ten compared with a local Unix workstation.

Running the code on the grid system required some adaptations to the computing environment different from the one the code was used on previously. The compiler producing the 64 bit version executable required some specific input / output settings and prohibited some commands that had to be replaced by equivalents. For initiating the batch execution, a Unix shell script was written that writes for every set of parameters adjusted input files, creates directories named accordingly, copies necessary files into their respective directories and launches the Condor job via a tailored start-up file. An example of such a script is attached in A.2.

When the simulation runs stable, this adaptation to a high power computing environment allows to efficiently execute an enormous number of simulations in parallel and with that to carry out parametric studies of the disturbance evolution. The approach intended to be followed at the outset of this work was to trace the influence of e. g. the forcing frequency and amplitude, the REYNOLDS number, the profile velocity ratio etc. by changing these parameters individually or combined in small steps and to determine their influence on stability of the flow.

3.3. Identification of Wavenumber

Two values absolutely necessary for further steps of the stability analysis are the real and imaginary part of the wavenumber of the oscillating disturbances obtained from numerical simulation. Using the different α_i obtained from a high number of simulations – when they can be performed – with varying ω_r, Re and so on, a stability diagram giving the relationship α_i vs. ω_r can be drawn. Since the flow data from the simulation is at hand in the form of discrete values at x- and y-stations, this data has to be related to an analytical expression which then gives the wavenumber.

We chose to fit a pre-defined function with unknown parameters to the data using an implementation of the nonlinear least-squares (NLLS) MARQUARDT-LEVENBERG algorithm as it is provided by the "fit" function of the software package `gnuplot` (Crawford, 2004).

The function to represent the values is

$$h(x) = A\,\mathrm{e}^{-\alpha_i x}\cos(\alpha_r x + \varphi)\;, \tag{3.1}$$

which corresponds to the ansatz of the analytical stability analysis, extended by the phase shift φ to account for the prescribed origin of the perturbation i. e. the forcing location.

To exclude transitional effects during the development of the perturbation immediately downstream of the forcing, only data points with a distance of minimal 20 to the forcing location are taken into account.

As visible in FIG. 3.4 (p. 41), the interpolation function fits very well the data points in the segment of the highest amplitudes and therefore the retained imaginary and real wavenumber should correlate highly accurately with the true value in this section. Since a gap between amplitudes of the analytical function and the data points is perceivable for later segments of the oscillation while the phase is still well matched, it becomes clear that the imaginary wavenumber is not constant over the whole perturbation wave. Therefore only points up to a certain downstream position (in the example below this is 450) are taken into account for the data fit. Additionally, the least-squares characteristic of the algorithm assures that the data points with high amplitudes are the relevant ones for determining the value of α_i.

The print-out of results from the "fit" function for the data depicted in FIG. 3.4 (p. 41) reads as shown in TAB. 3.1 (p. 42).

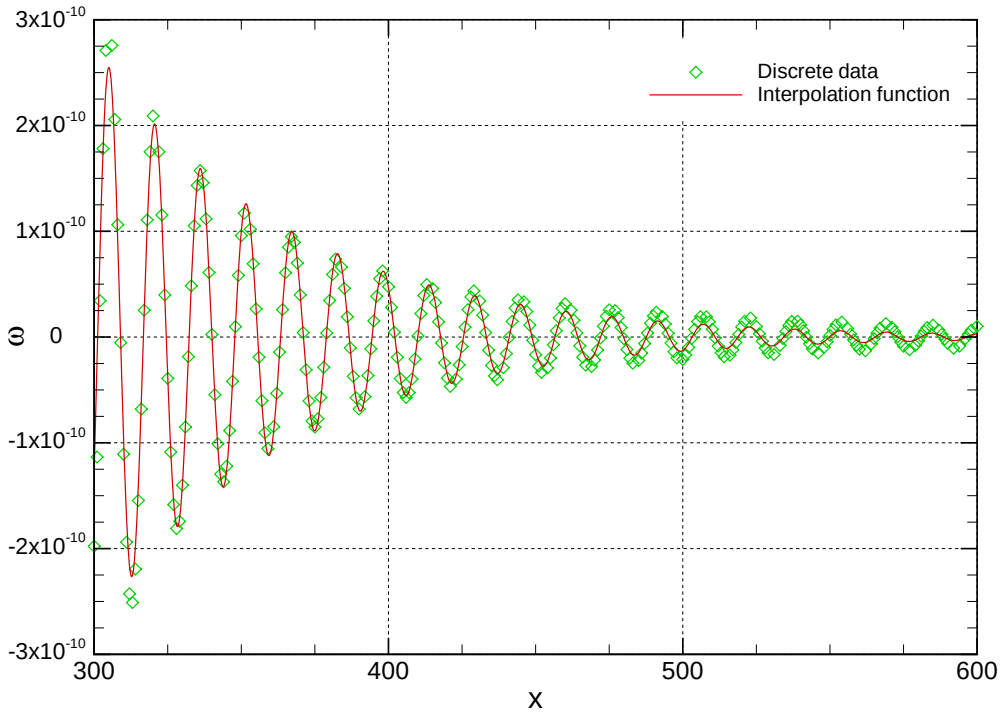

FIGURE 3.4. – *Example of an interpolation of data points for finding real and imaginary wavenumber. The data interpolated here is the vorticity ω_z from a generic stable configuration, obtained from the first time steps before the artificial instability overrides.*

3.4. Examples of Physical Interpretations

The variables involved are all non-dimensional and of global validity; however, to get a feeling for the physical and technical relevance of the values appearing in the stability analysis, a short excursus to dimensional values should be done here. Especially interesting is the dimensional value of the frequencies imposed. Some exemplary values are printed in TAB. 3.2 (p. 43).

```
FIT:    data read from 'centvt0555.dat' us 1:7
        #datapoints = 131
        residuals are weighted equally (unit weight)

function used for fitting: f(x)
fitted parameters and initial values from file: 1dim3.par

Iteration 0
WSSR        : 6.47444e-019      delta(WSSR)/WSSR   : 0
delta(WSSR) : 0                 limit for stopping : 1e-010
lambda    : 0.00868933

initial set of free parameter values
A               = 2e-010
alpha_i         = 0.01
alpha_r         = 0.42
phi             = 2.8

After 152 iterations the fit converged.
final sum of squares of residuals : 1.36424e-021
rel. change during last iteration : -1.7595e-011

degrees of freedom (ndf) : 127
rms of residuals      (stdfit) = sqrt(WSSR/ndf)      :
    3.27751e-012
variance of residuals (reduced chisquare) = WSSR/ndf : 1.0742
    e-023

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
A               = 2.55292e-008     +/- 1.285e-009   (5.035%)
alpha_i         = 0.0151035        +/- 0.0001456    (0.9642%)
alpha_r         = 0.404437         +/- 0.0001488
    (0.03679%)
phi             = 8.55782          +/- 0.05223      (0.6103%)

correlation matrix of the fit parameters:
                A      alpha_ alpha_ phi
A                1.000
alpha_i          0.995  1.000
alpha_r         -0.113 -0.179  1.000
phi              0.115  0.180 -0.997  1.000
```

TABLE 3.1. – *Print-out of results from the "fit" function for exemplary data.*

		η^* $[\frac{\mathrm{kg}}{\mathrm{m\,s}}]$	$17.3 \cdot 10^6$		η^* $[\frac{\mathrm{kg}}{\mathrm{m\,s}}]$	$1.00 \cdot 10^3$	
		ρ^* $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	1.225		ρ^* $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	1000	
		ν^* $[\frac{\mathrm{m}^2}{\mathrm{s}}]$	$14.1 \cdot 10^6$		ν^* $[\frac{\mathrm{m}^2}{\mathrm{s}}]$	$1.00 \cdot 10^6$	
U_∞^* $[\frac{\mathrm{m}}{\mathrm{s}}]$	10.0	Re $[-]$	ω^* $[\frac{\mathrm{rad}}{\mathrm{s}}]$	f^* [Hz]	Re $[-]$	ω^* $[\frac{\mathrm{rad}}{\mathrm{s}}]$	f^* [Hz]
ω $[-]$	1	1	$7.08 \cdot 10^6$	$1.13 \cdot 10^6$	1	$100 \cdot 10^6$	$15.9 \cdot 10^6$
		10	$708 \cdot 10^3$	$113 \cdot 10^3$	10	$10.0 \cdot 10^6$	$1.59 \cdot 10^6$
		100	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	100	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		1000	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	1000	$100 \cdot 10^3$	$15.9 \cdot 10^3$
		10000	708	113	10000	$10.0 \cdot 10^3$	$1.59 \cdot 10^3$
ω $[-]$	0.1	1	$708 \cdot 10^3$	$113 \cdot 10^3$	1	$10.0 \cdot 10^6$	$1.59 \cdot 10^6$
		10	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	10	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		100	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	100	$100 \cdot 10^3$	$15.9 \cdot 10^3$
		1000	708	113	1000	$10.0 \cdot 10^3$	$1.59 \cdot 10^3$
		10000	70.8	11.3	10000	$1.00 \cdot 10^3$	159
ω $[-]$	0.01	1	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	1	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		10	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	10	$100 \cdot 10^3$	$15.9 \cdot 10^3$
		100	708	113	100	$10.0 \cdot 10^3$	$1.59 \cdot 10^3$
		1000	70.8	11.3	1000	$1.00 \cdot 10^3$	159
		10000	7.08	1.13	10000	100	15.9
U_∞^* $[\frac{\mathrm{m}}{\mathrm{s}}]$	100.0	Re $[-]$	ω^* $[\frac{\mathrm{rad}}{\mathrm{s}}]$	f^* [Hz]	Re $[-]$	ω^* $[\frac{\mathrm{rad}}{\mathrm{s}}]$	f^* [Hz]
ω $[-]$	1	1	$70.8 \cdot 10^6$	$11.3 \cdot 10^6$	1	$1.00 \cdot 10^9$	$159 \cdot 10^6$
		10	$7.08 \cdot 10^6$	$1.13 \cdot 10^6$	10	$100 \cdot 10^6$	$15.9 \cdot 10^6$
		100	$708 \cdot 10^3$	$113 \cdot 10^3$	100	$10.0 \cdot 10^6$	$1.59 \cdot 10^6$
		1000	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	1000	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		10000	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	10000	$100 \cdot 10^3$	$15.9 \cdot 10^3$
ω $[-]$	0.1	1	$7.08 \cdot 10^6$	$1.13 \cdot 10^6$	1	$100 \cdot 10^6$	$15.9 \cdot 10^6$
		10	$708 \cdot 10^3$	$113 \cdot 10^3$	10	$10.0 \cdot 10^6$	$1.59 \cdot 10^6$
		100	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	100	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		1000	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	1000	$100 \cdot 10^3$	$15.9 \cdot 10^3$
		10000	708	113	10000	$10.0 \cdot 10^3$	$1.59 \cdot 10^3$
ω $[-]$	0.01	1	$708 \cdot 10^3$	$113 \cdot 10^3$	1	$10.0 \cdot 10^6$	$1.59 \cdot 10^6$
		10	$70.8 \cdot 10^3$	$11.3 \cdot 10^3$	10	$1.00 \cdot 10^6$	$159 \cdot 10^3$
		100	$7.08 \cdot 10^3$	$1.13 \cdot 10^3$	100	$100 \cdot 10^3$	$15.9 \cdot 10^3$
		1000	708	113	1000	$10.0 \cdot 10^3$	$1.59 \cdot 10^3$
		10000	71	11.3	10000	$1.00 \cdot 10^3$	159

TABLE 3.2. – *Dimensionalised values of the flow of air and of water at two different free-stream velocities, for two different disturbance frequencies and at five different Reynolds numbers.*

4. Trailing Edge Structure

The basic state without introduction of disturbances could be simulated without numerical instabilities and thus can be used to examine the quality and credibility of the flow the `PCNAVWAKEBD` *code produces. This will be done in this chapter by comparing the numerical results with what asymptotical theory (triple-deck theory, which will be reviewed in short) predicts for the delicate region around the trailing edge of the flat plate.*

4.1. Theoretical Reasoning

Classical boundary layer theory, as introduced by PRANDTL (Prandtl, 1904), holds for the viscid flow over a fixed surface and BLASIUS' solution (cf. 1.4) to the equations gives the velocity profile over the plate. GOLDSTEIN's theory (Goldstein, 1930) shows that the BLASIUS solution continues in the wake of the plate but it entrails the so-called GOLDSTEIN singularity immediately at the trailing edge: due to the discontinuity of the streamwise velocity u from no-slip to finite centreline value (therefrom also a discontinuity in the slope of the displacement thickness $\frac{\partial \delta^*}{\partial x}$), the transverse velocity v is singular in this point. The pressure, involved in the PRANDTL boundary layer equations, also becomes singular when approaching the trailing edge from upstream and this additionally leads to a discrepancy between computed and real drag coefficient.

To cope with this problem, a multi-structured asymptotic theory, that regards the dependency of the structure and the flow variables from the REYNOLDS number, was suggested mainly by MESSITER (Messiter, 1970) and STEWARTSON (Stewartson, 1969). This so-called triple-deck theory removes the trailing edge singularity by higher order expansions involving the REYNOLDS number and states the existence of three subsequent regions located above each other: In the lower deck (also called viscous sub-layer or wall layer) the classical boundary layer equations apply and with the altered boundary condition at the trailing edge it corresponds to GOLDSTEIN's inner viscous wake. The parabolic nature in this region inhibits upstream influence. The main deck corresponds to GOLDSTEIN's outer wake which to first order is the inviscid continuation of the Blasius boundary layer solution. In the additional inviscid upper (or outer) deck the displacement of the wake induces a pressure force, which in turn, acts on the lower deck – also in upstream direction since the flow in the upper deck is of elliptic type. Within the streamwise extent of the triple-deck region, the downstream wake influences

the upstream boundary layer on the plate via viscously-induced interaction with the potential flow upper deck. Accordingly, triple-deck theory is characterised not only by a different wall-normal segmentation of the flow domain but also by an interactive instead of hierarchical nature.

4.2. Triple-Deck Scalings

The expansions found by the triple-deck theory scale the geometry of the flow in terms of powers of the Reynolds number. Detailed discussion on that subject, that is reviewed briefly below, may be found in Schlichting and Gersten (2006); Rothmayer and Smith (1998); Smith (1982); Jobe and Burggraf (1974), e. g.

Examination of the pressure gradient around the trailing edge supposes an expansion in the order of $\varepsilon = Re_{x^*}^{-\frac{3}{8}}$.

As the problem is centred on the trailing edge, the origin for the local coordinate X will be shifted to this point.

The first assumption for finding the appropriate scalings is that the alteration of the velocity profile entirely takes place in the lower deck and that the slope of the Blasius profile in this deck can be approximated as constant, which means treating it as a uniform shear layer. With the boundary layer thickness $\delta^* = \mathcal{O}\left(Re_{x^*}^{\frac{1}{2}}\right)$ and the lower deck thickness stated as $Y^L = \mathcal{O}\left(\delta/\delta^*\right)$, with the parameter δ yet to be determined, then for the streamwise velocity in the lower deck it follows $U^L = \mathcal{O}\left(\delta\right)$ because of the uniform shear. The induced pressure must be $p = \mathcal{O}\left(\delta^2\right)$ for its gradient to balance the inertial forces of $\mathcal{O}\left(U^{L\,2}\right)$. If the length of the triple deck region is χ then the balance with the viscous forces requires $\delta = \mathcal{O}\left(\chi^{\frac{1}{3}}\right)$. The passive main deck only translates the displacement provoked by the trailing edge flow (which adds to the displacement of the boundary layer) to the upper deck, i. e. it is shifted by the lower deck thickness, while the pressure therefore is assumed to remain unaltered across the main deck. The pressure disturbance evolving from the displacement in the upper deck is of the order of the displacement slope, namely $\mathcal{O}\left(\frac{\delta/\delta^*}{\chi}\right)$. This external pressure should match the plate pressure $\mathcal{O}\left(\delta^2\right)$ in magnitude. When finally combining all these conditions, we get the basic triple-deck scalings $\chi = \mathcal{O}\left(\varepsilon\right)$, $X = \mathcal{O}\left(\varepsilon\right)$, $\delta = \mathcal{O}\left(\varepsilon^{\frac{1}{3}}\right)$, $U^L = \mathcal{O}\left(\varepsilon^{\frac{1}{3}}\right)$, and $p = \mathcal{O}\left(\varepsilon^{\frac{2}{3}}\right)$.

To satisfy the continuity equation it must be $\mathcal{O}\left(\frac{U^L}{\chi}\right) = \mathcal{O}\left(\frac{V^L}{\delta/\delta^*}\right)$ and therefore $V^L = \mathcal{O}\left(\varepsilon\right)$.

The displacement and the pressure disturbance exercised on the outer deck must be

linked to a normal velocity that must originate from the main deck and have the same order of magnitude, i.e. $V^M = \mathcal{O}(p) = \mathcal{O}\left(\varepsilon^{\frac{2}{3}}\right)$. With the height of the main deck identical to the boundary layer thickness δ^*, the streamwise velocity scaling is gained from continuity equation again: $\mathcal{O}\left(\frac{U^M}{\chi}\right) = \mathcal{O}\left(\frac{V^M}{\delta^*}\right)$ and therefore $U^M = \mathcal{O}\left(\varepsilon^{\frac{1}{3}}\right)$.

The same argument gives that $V^U = \mathcal{O}(p) = \mathcal{O}\left(\varepsilon^{\frac{2}{3}}\right)$. As the upper deck is part of the inviscid potential flow with the flow structure around the trailing edge having no direct influence to it, its scaling in x- and y-direction must be the same: $Y^U = \mathcal{O}(X)$. Consequently the streamwise velocity must be of same order as the normal velocity, which means $U^U = \mathcal{O}\left(\varepsilon^{\frac{2}{3}}\right)$.

When translating these estimates of orders of magnitude into actual variable scalings, the use of the Reynolds number based on the displacement thickness is practical for our simulations. This is done using equations (1.84) and (1.71) and making some further estimates of orders of magnitude for the basic flow variables; a constant factor of $\left(2\delta^2_{[\eta]}\right)^a$ appearing there is dropped because it is irrelevant for our qualitative examination. Eventually, the following scalings apply:

$$X^L = Re^{-\frac{1}{4}}(x - x_{TE}) \tag{4.1}$$
$$Y^L = Re^{\frac{1}{4}}y \tag{4.2}$$
$$U^L = Re^{\frac{1}{4}}u \tag{4.3}$$
$$V^L = Re^{\frac{1}{8}}v \tag{4.4}$$
$$P^L = Re^{\frac{1}{2}}p \tag{4.5}$$

$$X^M = Re^{-\frac{1}{4}}(x - x_{TE}) \tag{4.6}$$
$$Y^M = Re^{0}y \tag{4.7}$$
$$U^M = Re^{\frac{1}{4}}u \tag{4.8}$$
$$V^M = Re^{\frac{1}{2}}v \tag{4.9}$$
$$P^M = Re^{\frac{1}{2}}p \tag{4.10}$$

$$X^U = Re^{-\frac{1}{4}}(x - x_{TE}) \tag{4.11}$$
$$Y^U = Re^{-\frac{1}{4}}y \tag{4.12}$$
$$U^U = Re^{\frac{1}{2}}u \tag{4.13}$$
$$V^U = Re^{\frac{1}{2}}v \tag{4.14}$$
$$P^U = Re^{\frac{1}{2}}p \tag{4.15}$$

4.3. Comparison of Numerical Results with Asymptotic Theory

A first opportunity to scrutinise the quality of the results obtained from our numerical simulations using the `PCNAVWAKEBD` code is the examination of the flow structure produced for the region around the trailing edge. Not only is this structure both restricted to a very small area and associated with analytically singular values and steep gradients but it is taking place around the area where the numerical properties are fundamentally changed by an alteration of the boundary conditions – from the no-slip condition on the rigid wall to the symmetry condition of the wake. Therefore, this test of the basic state simulation can be considered a relatively rigorous one.

4.3.1. Scaling and Structures

The accordance of the results obtained from numerical simulation with the structures predicted by triple-deck theory can easily verified by scaling the flow field as the expansions for the different decks suppose and as detailed above. Values obtained from simulations accomplished under different Reynolds numbers should coincide in the respective X–Y–$U/V/P$ space as the dependency from the Reynolds number should be removed by the scaling.

Fig. 4.1, 4.2 and 4.3 (pp. 48, 49 and 50) give the distribution of the variables u, v and p over x after applying these scalings for upper, main and lower deck, respectively.

When we return to Reynolds numbers based on streamwise distance, using (1.84), we see that the range $Re = 10 \ldots 10000$ corresponds to the range $Re_{x^*} = 1.72 \times 10^2 \ldots 10^8$ and therefore this is a span of seven magnitudes. Given this fact, the results in Fig. 4.1, 4.2 and 4.3 (pp. 48, 49 and 50) may be considered as coinciding quite well, especially when not focusing on the "low Reynolds number" regime $Re_{x^*} = 172$.

At the near right corner of Fig. 4.1 (p. 48) ($X \overset{<}{\rightarrow} 15$, $Y \overset{>}{\rightarrow} 2$) a steep increase of U can be noticed for $Re = 10$, with U departing the otherwise good qualitative congruence with results from other Reynolds numbers. An explanation to this is probably the small-scale velocity perturbation as discussed in 5.1, that is most apparent for this low Reynolds number.

4.3.2. Centreline Velocity

Asymptotic theory gives the centreline velocity as

$$U(X,0) = 0.8991 \cdot 1.343^{\frac{2}{3}} X^{\frac{1}{3}} + \mathcal{O}\left(X^{\frac{2}{3}}\right) , \tag{4.16}$$

as determined by Jobe and Burggraf (1974).

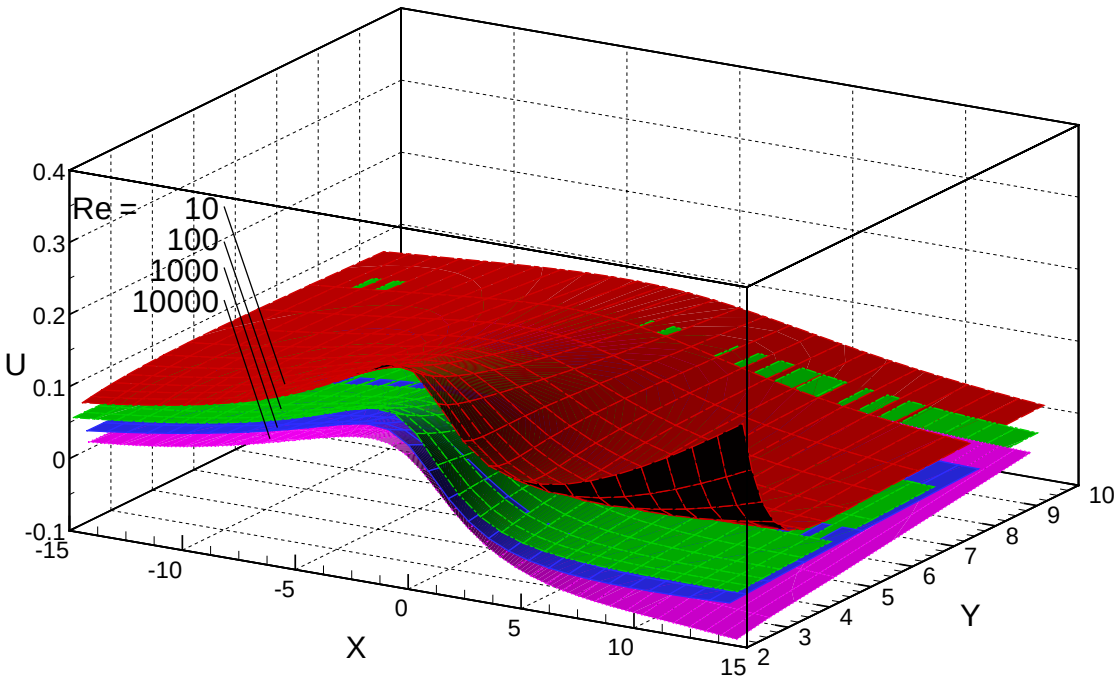

FIGURE 4.1. – *Scaled streamwise velocity in the upper deck for several* REYNOLDS *numbers in the scaled coordinates.*

In FIG. 4.4 (p. 51) this value is plotted together with the results obtained from the simulations with the `PCNAVWAKEBD` code. A qualitative accordance is clear, even in quantitative terms the difference is moderate for higher REYNOLDS numbers.

4.3.3. Pressure

The pressure in the triple deck region is related to the so-called slip velocity $A(X)$, that can be obtained from asymptotic theory, via the HILBERT integral

$$P(X) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{\frac{\partial A}{\partial \xi}\, \mathrm{d}\xi}{X - \xi}\,, \tag{4.17}$$

which was solved numericaly e. g. by Jobe and Burggraf (1974).

Comparison of the results obtained here with the results from asymptotic theory can be seen in FIG. 4.5 (p. 52).

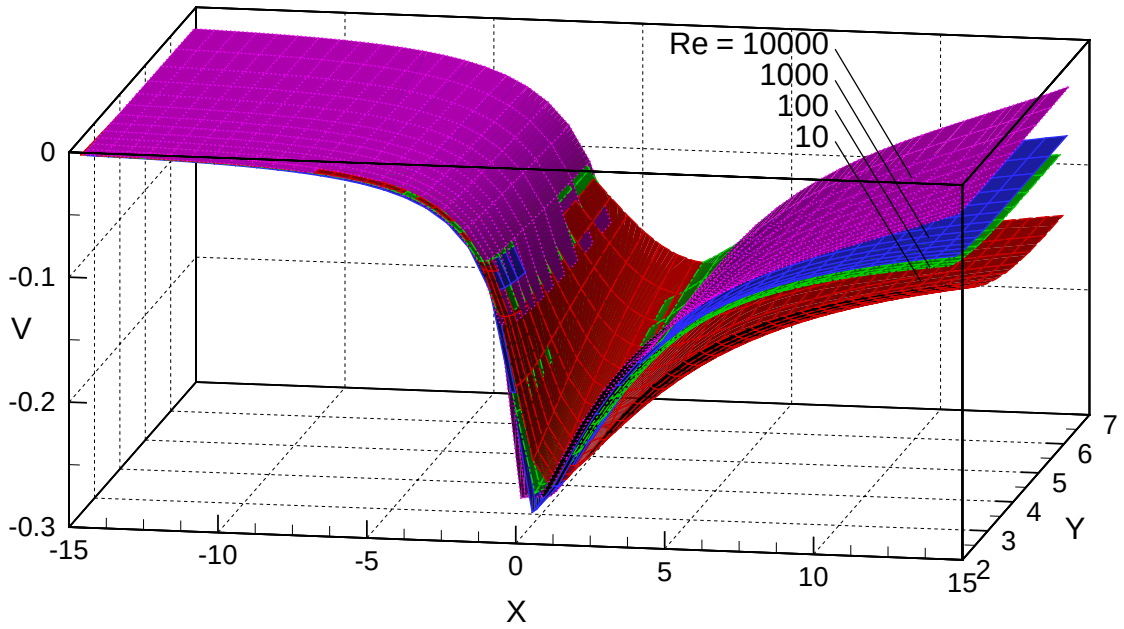

FIGURE 4.2. – *Scaled normal velocity in the main deck for several* REYNOLDS *numbers in the scaled coordinates.*

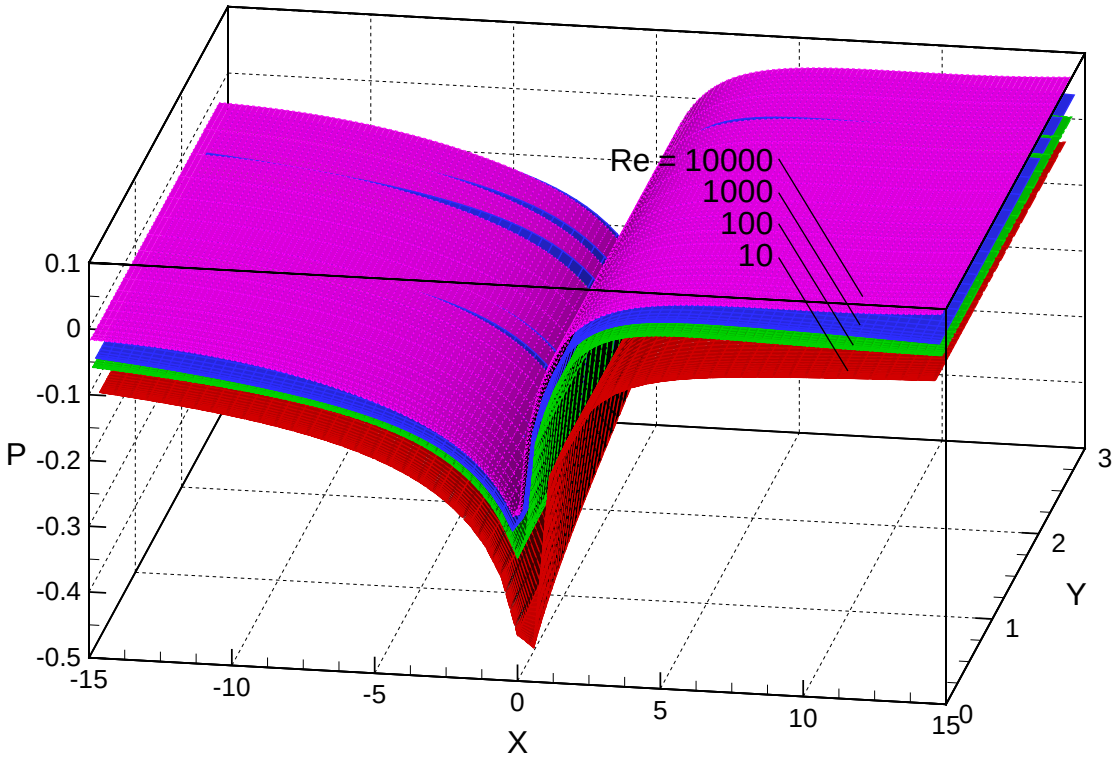

FIGURE 4.3. – *Scaled pressure in the lower deck for several* REYNOLDS *numbers in the scaled coordinates.*

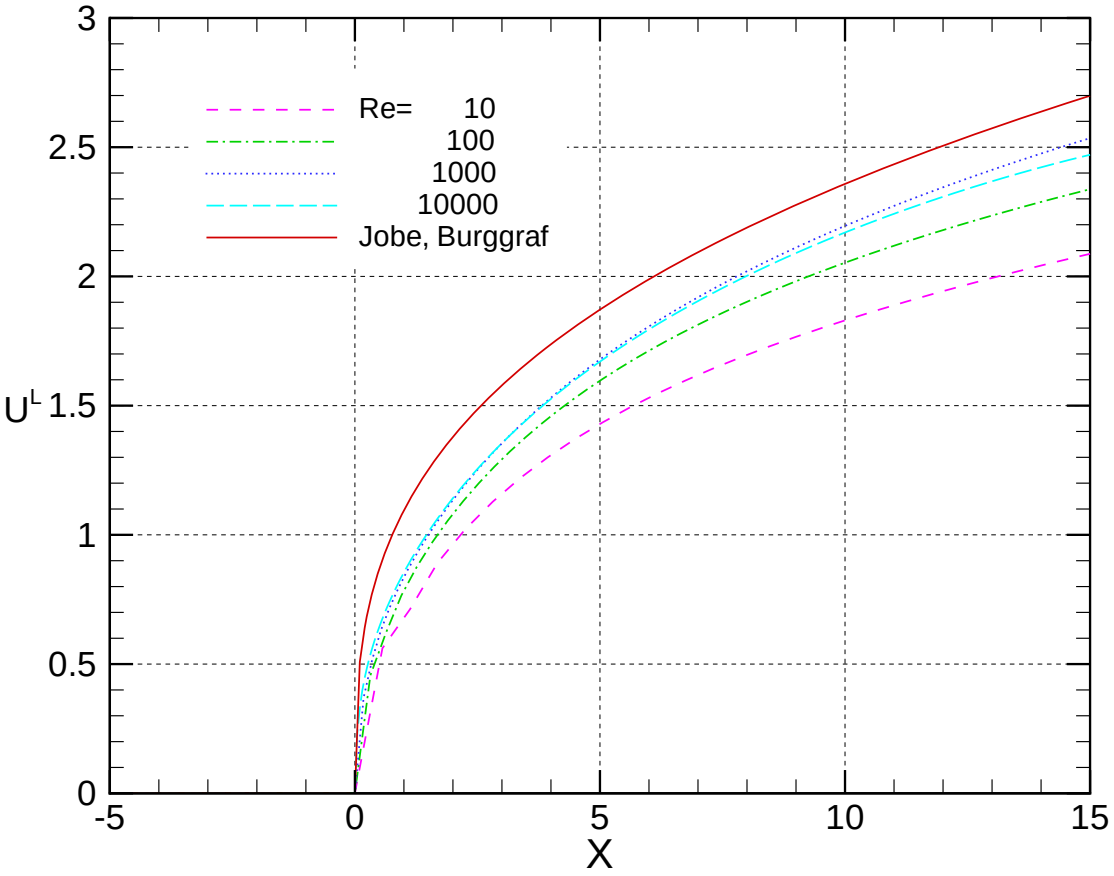

FIGURE 4.4. – *Centreline streamwise velocity for several* REYNOLDS *numbers and as calculated with asymptotic theory by Jobe and Burggraf (1974).*

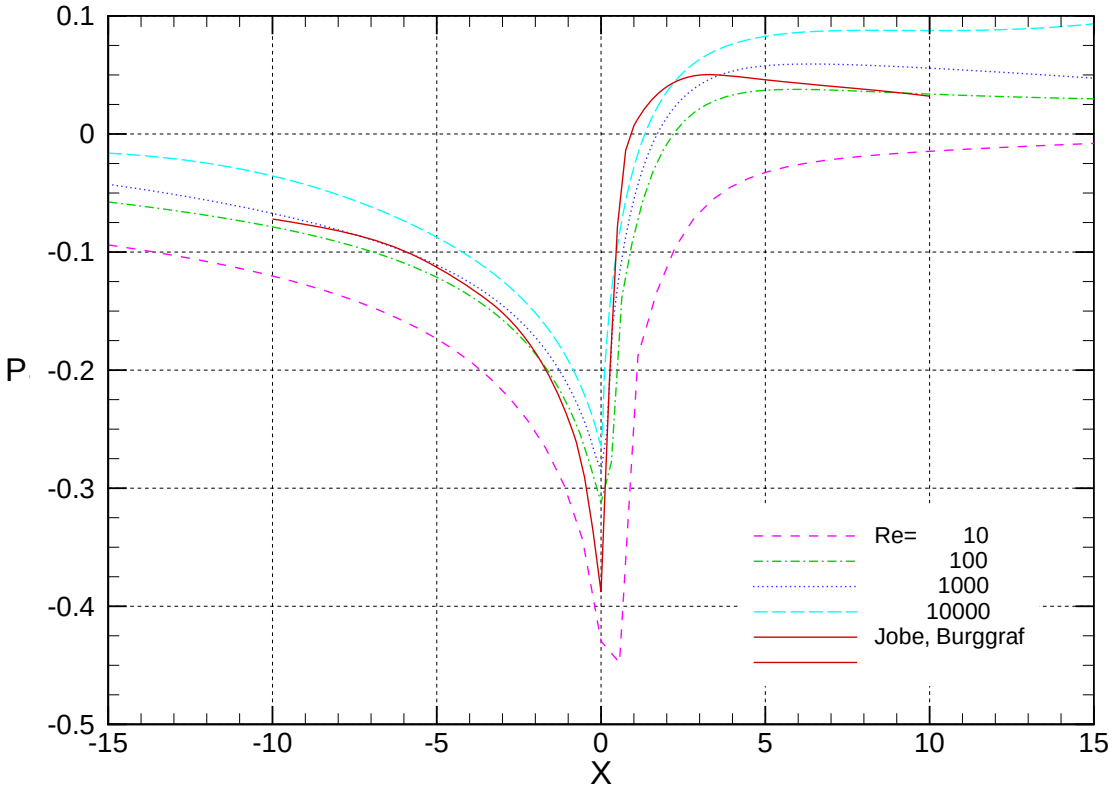

FIGURE 4.5. – *Pressure value in lower deck for several* REYNOLDS *numbers and as calculated with asymptotic theory by Jobe and Burggraf (1974).*

5. Stability Analysis

As an alternative to the stability analysis by direct numerical simulation that could not be carried out, the eigenvalue problem for the two basic flow types can be solved. This chapter outlines the approach to this, gives some findings about the simulated mean flow that were discovered here and presents the results of the eigenvalue solving. Dependence of the stability properties from several parameters is discussed.

5.1. Idea and Approach

To get information on the stability properties of the flow considered in this study without being dependent on direct numerical simulation, the approach is to solve the eigenvalue problem for the partial disturbances as discussed in 1.8, namely to solve the Orr-Sommerfeld equation (1.106).

The value of the stability analysis using the eigenvalue solver is to gain a basis for comparison for the stability behaviour of the flow obtained by direct numerical simulation (involving wake flow also coming from numerical simulation or from the prescribed mean flow concept). When both mean flow types will be used in the direct numerical simulation to carry out stability analysis by simulation, it will be very helpful and of high relevance for being able to make reasonable comparisons, to know their correlation from a more ideal-type setting. In this sense, the stability properties of both flow types are determined by eigenvalue analysis and their similarities and differences shall be worked out.

Concretely, the mean flow drawn from the direct numerical simulation had a centreline velocity at the downstream station chosen (cf. discussion below) of $U^t(x = 220, y = 0) = 0.3201$ at $Re = 500$ and $U^t(x = 220, y = 0) = 0.2500$ at $Re = 1000$, therefore a prescribed mean flow profile with $Q = 0.6799$ and $Q = 0.75$ respectively was taken as the reference for comparisons. The similarity of the simulated and the prescribed velocity profiles for $Re = 1000$ can be assessed in Fig. 1.2 (p. 21).

During the evaluation of the DNS mean flow for use for this stability analysis a problem with the small-scale velocity distribution became apparent: as it can be seen in Fig. 5.1 (p. 54) the streamwise velocity U (i) converges to a value of over 1 at x-stations closely behind the trailing edge and (ii) velocity profiles "peak" at a point at around $y = 3$ for greater x, while their respective U-distributions asymptotically approach a value of under 1 for $y \to \infty$. When extending the simulation towards higher x and / or higher times t, the U-values in the "hump" at around $y = 3$ grow boundless. Particular consequence of

that non-monotony is that the stability properties of the respective velocity distribution will be altered – probably significantly – due to the existence of additional inflexion points.

That behaviour hints to a developing inherent instability, which is probably due to instable growing modes that are introduced by numerical noise and error. It remains to be examined whether this is also linked to the global domain instability, but it seems improbable because instabilities excited by numerical truncation are not unusual. A change in the domain size in y-dimension, e. g. a multiplication from $y \in [-92.39, 92.39]$ to $y \in [-231.0, 231.0]$, did not affect this problem. The graphs did – apart from the differing locations of the grid points – perfectly coincide, therefrom an influence of the y-boundary can be excluded. For other Reynolds numbers the picture is also qualitatively the same with the extreme values growing faster in time and space for lower Re, cf. Fig. 5.2 (p. 55).

The x-station 220, chosen for the stability analysis carried out with the `LINSTAB` code, is safely placed between these two phenomena; its velocity distribution is monotonically increasing and converges to 1 for both $Re = 500$ and $Re = 1000$.

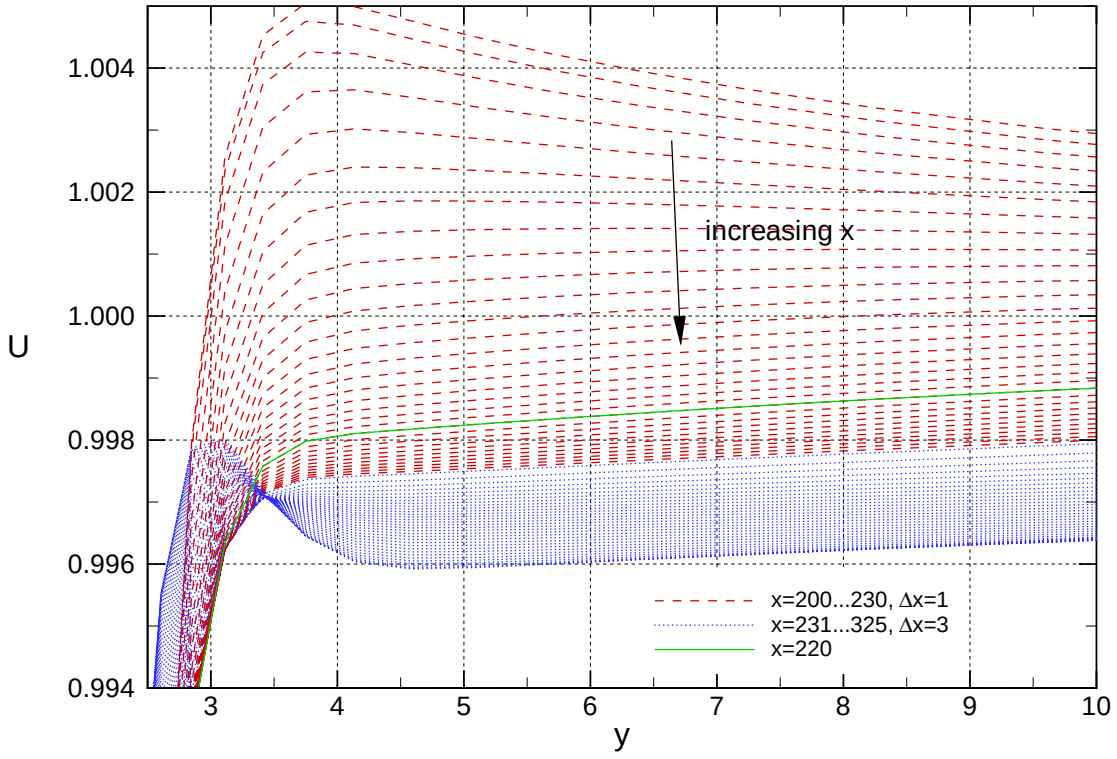

Figure 5.1. – *Streamwise velocity U^t for several x-positions from simulation at $Re = 1000$ and $t = 400$.*

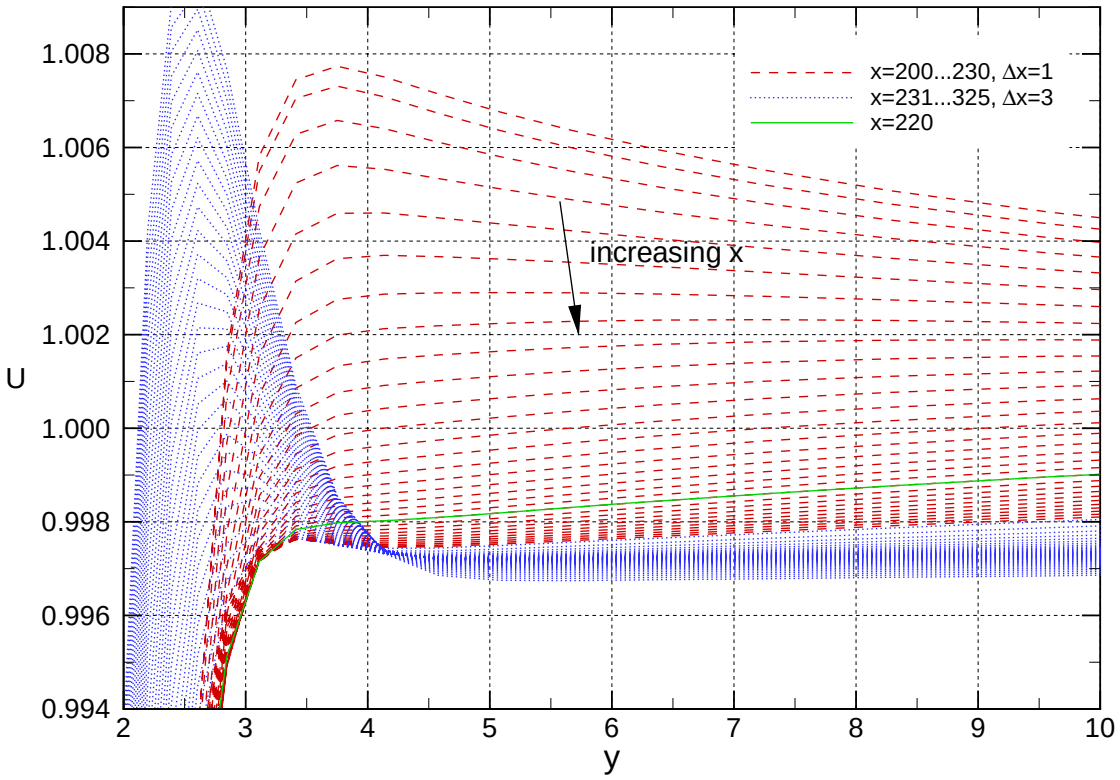

FIGURE 5.2. – *Streamwise velocity U^t for several x-positions from simulation at $Re = 500$ and $t = 400$.*

5.2. Numerical Method

The `LINSTAB` code available for this analysis is an eigenvalue solver for linear stability theory which solves the equation system of continuity equation, ideal gas law, momentum equation and energy equation. This means it is designed to cope with compressible flow but for the use here, the Mach number is set to 0.001, so it restricts to incompressibility, and effectively solves the ORR-SOMMERFELD equation. Refer to the work of Babucke (n.d.) for details on the theoretical and numerical aspects of `LINSTAB`.

For every specified x-station and for a given wavenumber α_r (with $\alpha_i = 0$) `LINSTAB` returns the eigenvalues $\omega_r + \mathrm{i}\omega_i$ and additionally to this spectrum, the associated eigenfunctions $\phi(y)$ for u, v and p are given with amplitude and phase.

The general approach here is a temporal stability analysis, which can be transformed onto the spatial stability problem by means of an eigenvalue following routine. Such a programme iteratively adjusts α_i to assure that ω_i vanishes, which then returns $\alpha_r + \mathrm{i}\alpha_i$ for fixed ω_r for the spatial stability analysis. This is another task to be persued in the course of a subsequent study that takes up the results gained here.

Both the mean flow obtained from running the PCNAVWAKE code with a trailing edge within the simulation domain, and the prescribed mean flow, which is itself foreseen as comparison basis for the results from the complete simulation, were examined for stability behaviour. As the prescribed mean flow is constant in x, the result for one station is valid for all stations but with the non-parallel wake-flow from the direct simulation a dependence on the downstream position can be investigated. This, of course, requires usable velocity profiles for all x-stations that are not available here, as discussed above.

5.3. Realisation of Numerical Analysis

In preparation for the execution of the eigenvalue solving, the mean flow produced by the PCNAVWAKE code had to be projected onto an equidistant grid as the LINSTAB code requires such an even spacing of the grid points. To achieve this, a linear interpolation, using the data processing and visualisation tool Tecplot, was applied. The range $y \in [-10, 10]$, represented by 115 original grid points, was interpolated by 301 new data points. In the inner part around the centreline with higher gradients, the original spacing was very narrow (e. g. 81 grid points in the range $y \in [-1.86, 1.86]$, making up an averaged spacing of 0.022), so with the constant spacing of 0.066 for the new grid, other types of interpolation would not have yielded any advantage. Likewise, in the outer regions where the original spacing are coarser than the new one, gradients are so small and nearly constant so that a linear interpolation was considered sufficient in accuracy also seen from that side. As the original grid is equidistant in x-direction, no change was applied in this dimension.

The prescribed mean flow was obtained in the correct format by creating a new independent programme derived from the PRESCMF module and adjusting it such that it produces the flow on an equidistant grid.

The EAS3 code – also provided by Institut für Aerodynamik und Gasdynamik – that performs the read-in of the mean flow for the LINSTAB code was adapted to the file format and formatting stemming from the PCNAVWAKE standard and some parameter settings of LINSTAB were tuned to arrive at producing meaningful results.

To get a significant picture of the stability, a certain number of solver runs, with varying parameters, were performed.

In order to rationalise launching of the solver runs and the processing of the data produced, again some Unix shell scripts were written; they can be consulted in A.6.

5.4. Results

The direct output of the solver is a spectrum for the given flow and for a pre-specified real wavenumber α_r, for which Fig. 5.3 (p. 57) gives two examples. It is visible that

– as expected for such an unbounded flow decaying to a constant in the freestream – there is a continuous spectrum of eigenvalues (the vertical accumulation of eigenvalues that remain discrete points here because of the numerical discretisation) and a range of discrete eigenvalues. Some eigenvalues are expected to be numerical eigenvalues not having any physical meaning. As the flow is symmetric in y, there is only one continuous spectrum.

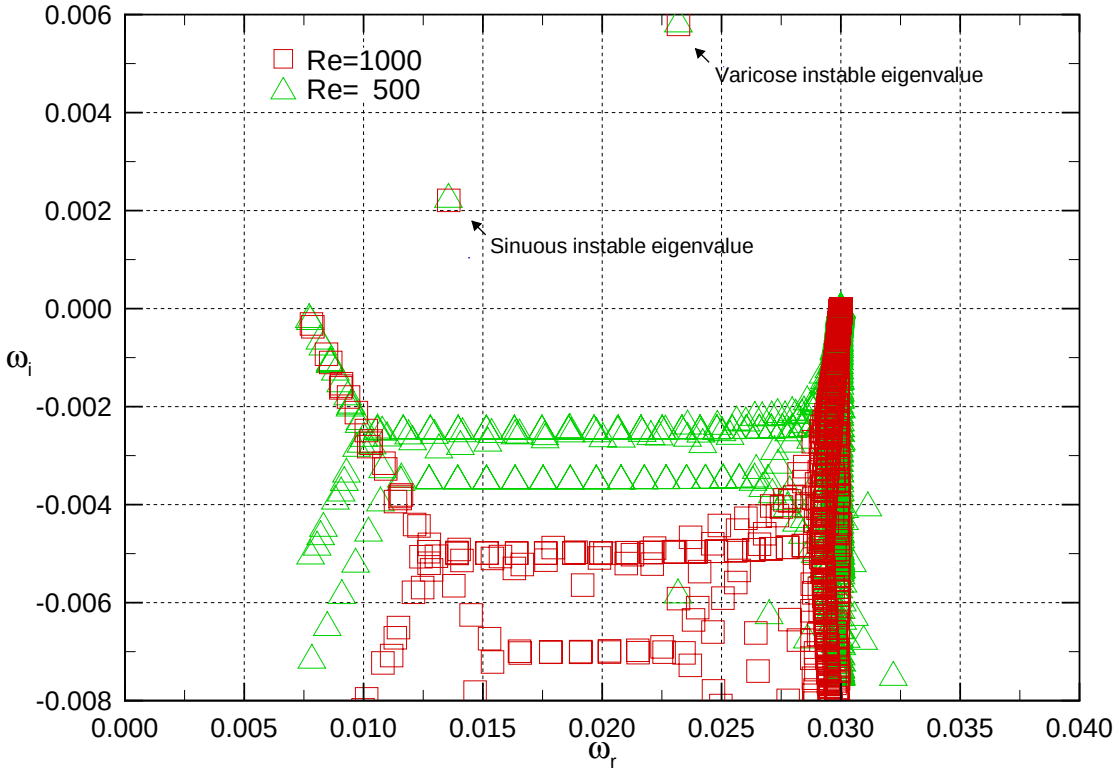

FIGURE 5.3. – *Spectrum of the prescribed flow for two* REYNOLDS *numbers at* $\alpha_r = 3{\cdot}10^{-2}$ *and* $Q = 0.75$.

On the other hand, FIG. 5.3 shows that the discrete eigenvalues move towards the continuous sprectrum for decreasing REYNOLDS numbers and one by one ultimatively move onto it, as it should be expected, and move towards $\omega_i = 0$ as the flow gets less viscous.

After identifying the instable eigenvalues – if there are any – and repeating the procedure for other wavenumbers, all instable eigenvalues are gathered in one graph.

FIG. 5.4 (p. 58) through FIG. 5.10 (p. 64) give an overview of these extracts of the results obtained from the linear stability solver execution.

Generally, it is apparent that two different modes of instability form for the flow under consideration – represented by the two curves in the eigenvalue diagrams. Mode I (the

more unstable, dominant one) is a varicose mode with the streamfunction and thereby the variables antisymmetric in y and mode II (the less unstable one in the lower right corner) is a sinuous mode with a symmetric streamfunction and symmetric distribution of physical quantities. In FIG. 5.11 (p. 65) this symmetry behaviour is reflected (cf. 5.4.5).

For all configurations examined, the flow is unstable for wavenumbers between $\alpha_r = 0$ and $\alpha_r \approx 11.8 \ldots 13.3 \cdot 10^{-2}$, at frequencies between $\omega_r = 0$ and $\omega_r \approx 6.2 \ldots 8.0 \cdot 10^{-2}$. Maximum instability occurs for wavenumbers of $\alpha_r \approx 5.0 \ldots 6.0 \cdot 10^{-2}$ at frequencies of about $\omega_r \approx 3.4 \ldots 4.4 \cdot 10^{-2}$ with temporal growth rates of $\omega_i \approx 6.3 \ldots 8.0 \cdot 10^{-3}$. In principle the varicose mode is the more instable one but for a small range of low wavenumbers / large wavelengths of $\alpha_r = 0$ to $\alpha_r \approx 1.0 \ldots 1.2 \cdot 10^{-2}$ the sinuous mode is less stable for some (not all) flow configurations.

5.4.1. Simulated and Prescribed Flow Stability Properties

The foremost interest is to compare the simulated and the prescribed flow in their respective stability properties and FIG. 5.4 and FIG. 5.5 (p. 59) allow to do so.

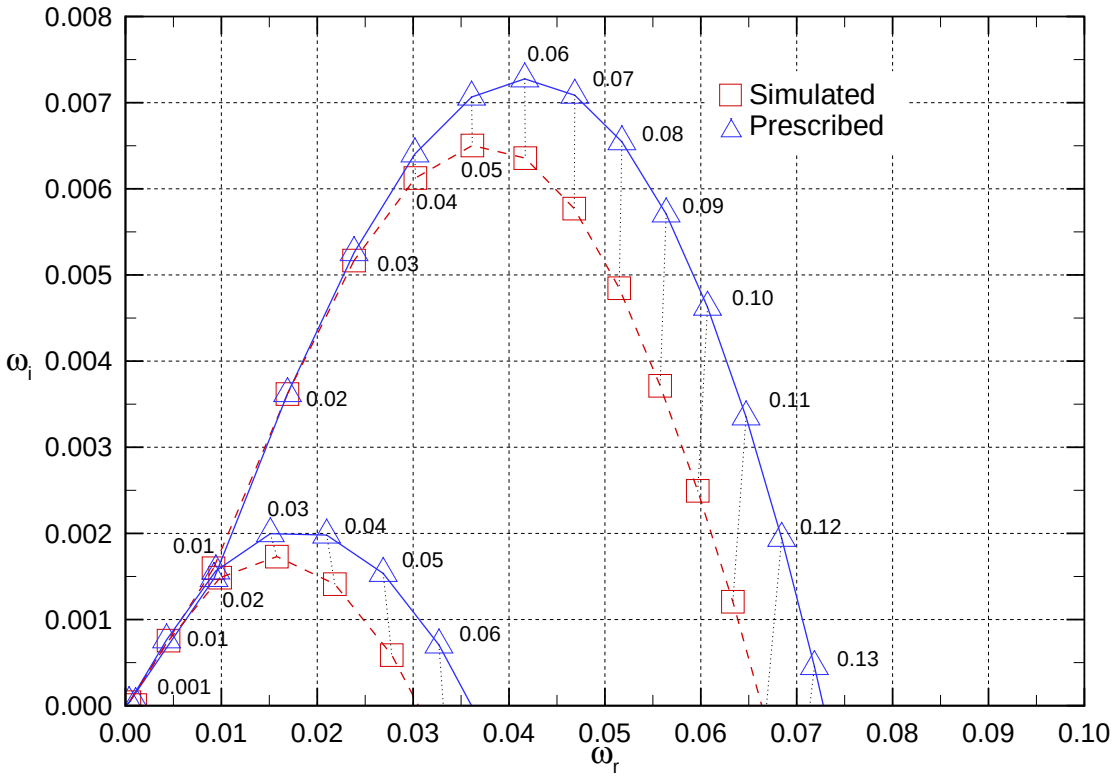

FIGURE 5.4. – *Instable eigenvalues of the simulated and the prescribed ($Q = 0.6799$) flow at $Re = 500$.*

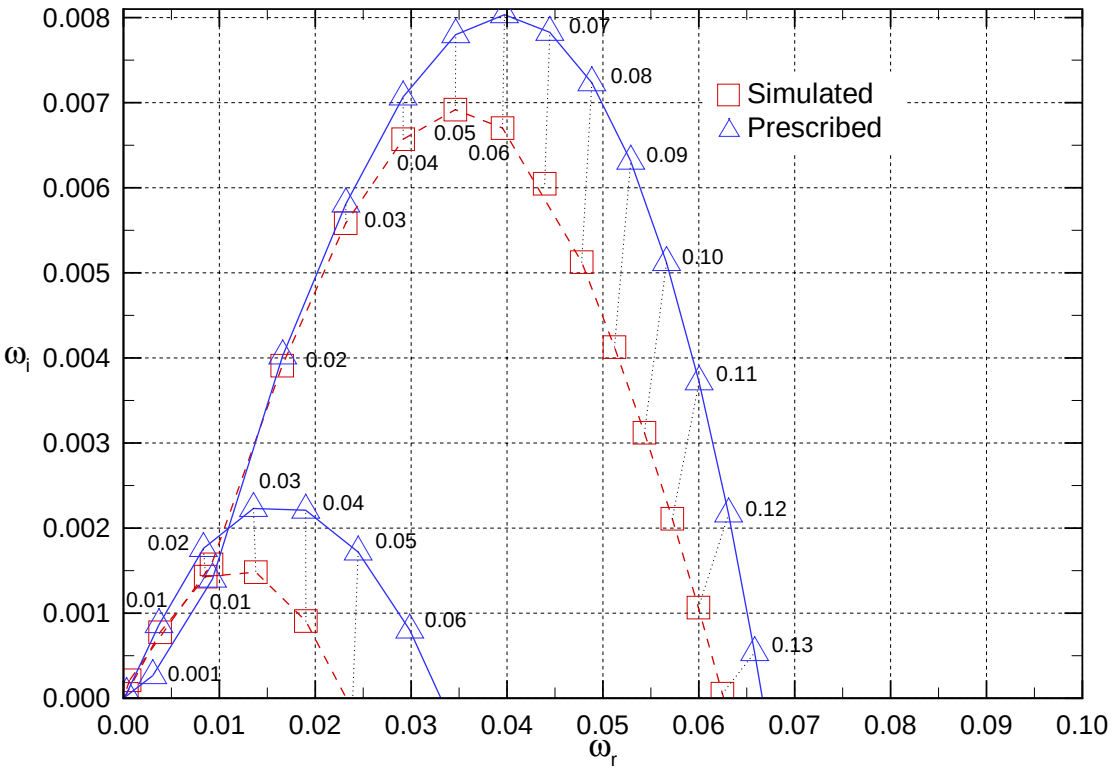

FIGURE 5.5. – *Instable eigenvalues of the simulated and the prescribed ($Q = 0.75$) flow at $Re = 1000$.*

The eigenvalue distributions do not perfectly coincide for all wavenumbers but for the lower range up to $\alpha_r \approx 3 \cdot 10^{-2}$ the congruence is very well. At the higher range, the α_r-isolines' vertical orientation (particularly ideally for $Re = 500$) and the relatively even spacing between the two curves suggest a linear scaling between them – the reasons for that hypothetic relationship remain to be clarified.

The exact values of $\omega_r = 0$ and ω_i for the case $Re = 500$ for both simulated and prescribed flow, together with the ratios and differences of them to show their relation, can be inspected in TAB. 5.1 (p. 66).

5.4.2. Variation with Profile Velocity Ratio

When focusing on the prescribed flow, a comparison of the flow with varying profile velocity ratio Q can be done. As FIG. 5.6 (p. 60) and FIG. 5.7 (p. 61) show, a lower value of Q comes with lower growth rates but within a higher range of instable frequencies, whereas the marginally instable wavenumber remains constant (the last α_r-isoline, though not drawn, would be exactly horizontal).

FIG. 5.6 (p. 60) also reveals another point: the spacing between the three eigenvalues

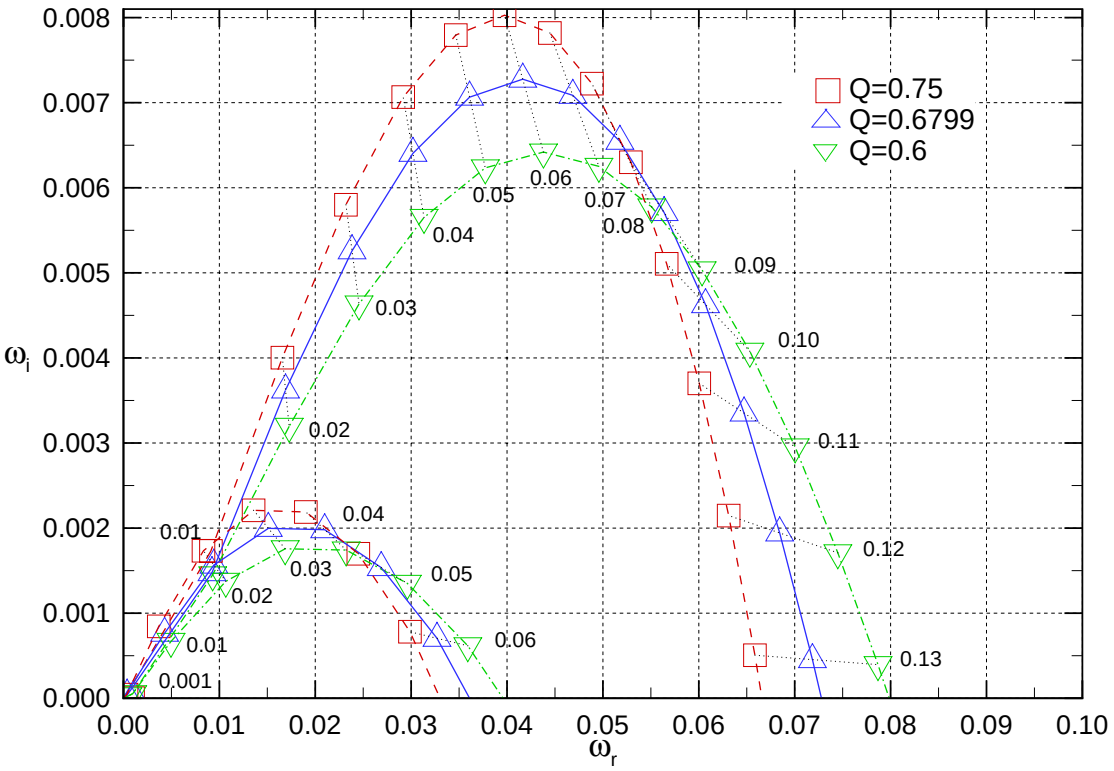

FIGURE 5.6. – *Instable eigenvalues of the prescribed flow at $Re = 500$ and different profile velocity ratios.*

for a given wavenumber apparently always has the same ratio (i. e. the ratio of the lengths of the α_r-isolines to both sides of the curve for $Q = 0.6799$ is constant). The details tabled in TAB. 5.2 (p. 67) confirm that and also show that this ratio is equal to the ratio of the profile velocity ratios (within less than 0.4 %). That can only mean that there must be a direct linear relationship between the profile velocity ratio and the location of an eigenvalue for a given wavenumber. A conclusion from that is that the wavenumber from which the most unstable eigenvalue is excited must be the same for all profile velocity ratios considered – confirmed by FIG. 5.6 (p. 60) and FIG. 5.7 (p. 61).

5.4.3. REYNOLDS Number Dependence

One can expect that the REYNOLDS number dependence on the instability is, for the simulated flow, primarily influenced by the different velocity profiles that come with different REYNOLDS numbers. Accordingly, FIG. 5.9 (p. 63) and FIG. 5.8 (p. 62) show two totally different situations: The REYNOLDS number is not a parameter for the prescribed

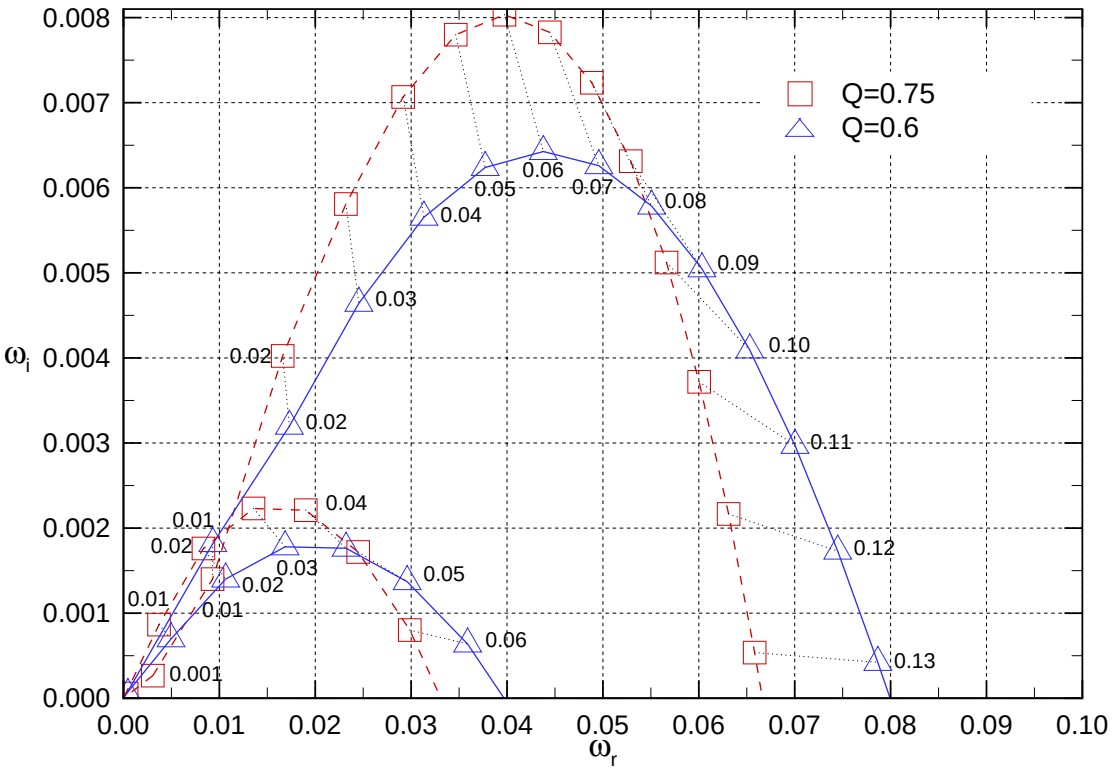

FIGURE 5.7. – *Instable eigenvalues of the prescribed flow at* $Re = 1000$ *and different profile velocity ratios.*

mean flow[3], which is manifested when looking into the eigenfunctions that also coincide for all wavenumbers[4]. Obviously, any influence of Re is negligible in the governing equations.

On the other hand, for the simulated flow, the location of the eigenvalues is different for the two curves, thus suggesting that this is entirely due to the differently shaped streamwise velocity distribution. In general, the picture agrees with the fact that the instabilities here are expected to be inviscid instabilities, i. e. appearing for both viscous and non-viscous flow.

[3] Note that the divergence of the eigenvalues for $\alpha_r = 0.001$ and 0.01 is most probably a numerical error: even in the curve showing the MACH number there is this deviation and here it is very improbable that just for these two points there should be a difference.

[4] With the exception that the amplitude of the eigenfunction for $\alpha_r = 0.01$ does not decay to 0 but slowly rises again while oscillating, together with a phase shift of 0.7π for the associated phases; see FIG. 5.12 (p. 68). This could be an explanation to the deviation discussed in the footnote above

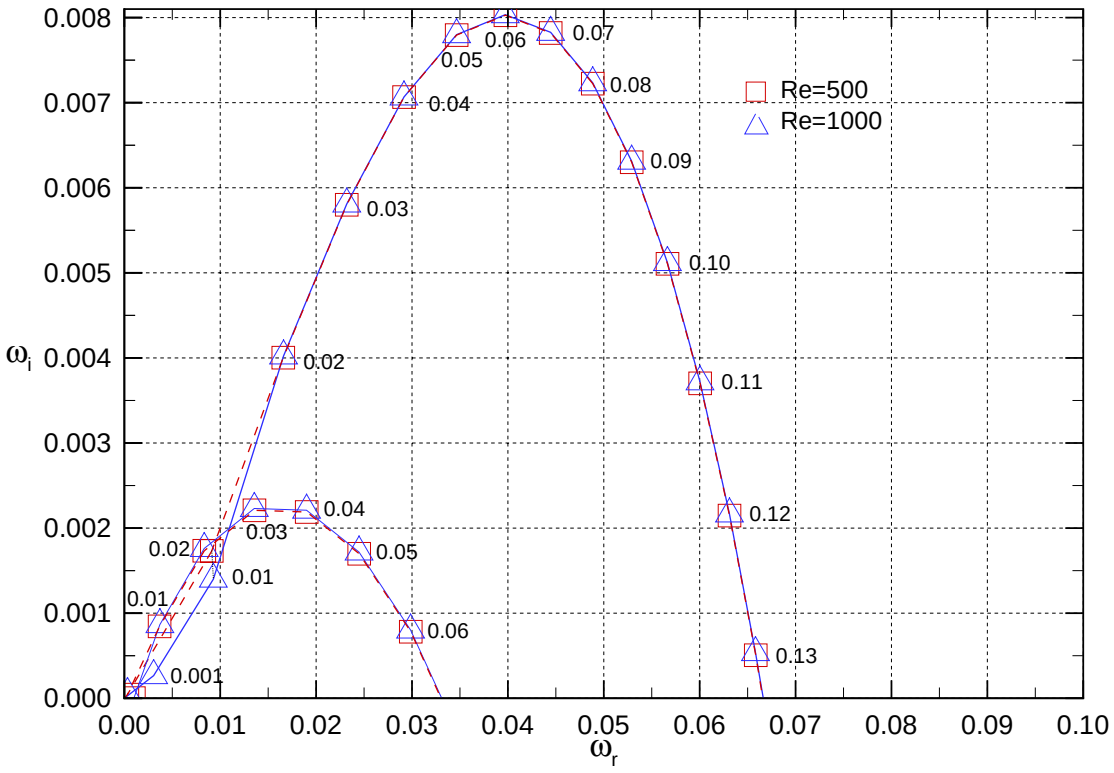

FIGURE 5.8. – *Instable eigenvalues of the prescribed flow at different* REYNOLDS *numbers and* $Q = 0.75$.

5.4.4. MACH Number Dependence

Finally, the change with the MACH number has to be considered as the solutions were obtained by running a solver that deals intrinsically with compressible flow. As FIG. 5.10 (p. 64) shows, there is no influence of the MACH number (apart from the erratic values for $\alpha_r = 0.001$ and 0.01, cf. footnote 3) for two magnitudes up from the fixed value chosen here – thus choosing $Ma = 0.001$ was a safe choice for approximating incompressible flow.

5.4.5. Eigenfunctions

The eigenfunction with amplitude and phase is also determined by the solver, FIG. 5.11 (p. 65) gives the picture of one example for this. It can be seen that the characteristic phase jumps are resolved and that the amplitudes follow the distributions expected for the two modes: symmetric for the sinuous mode and antisymmetric for the varicose mode.

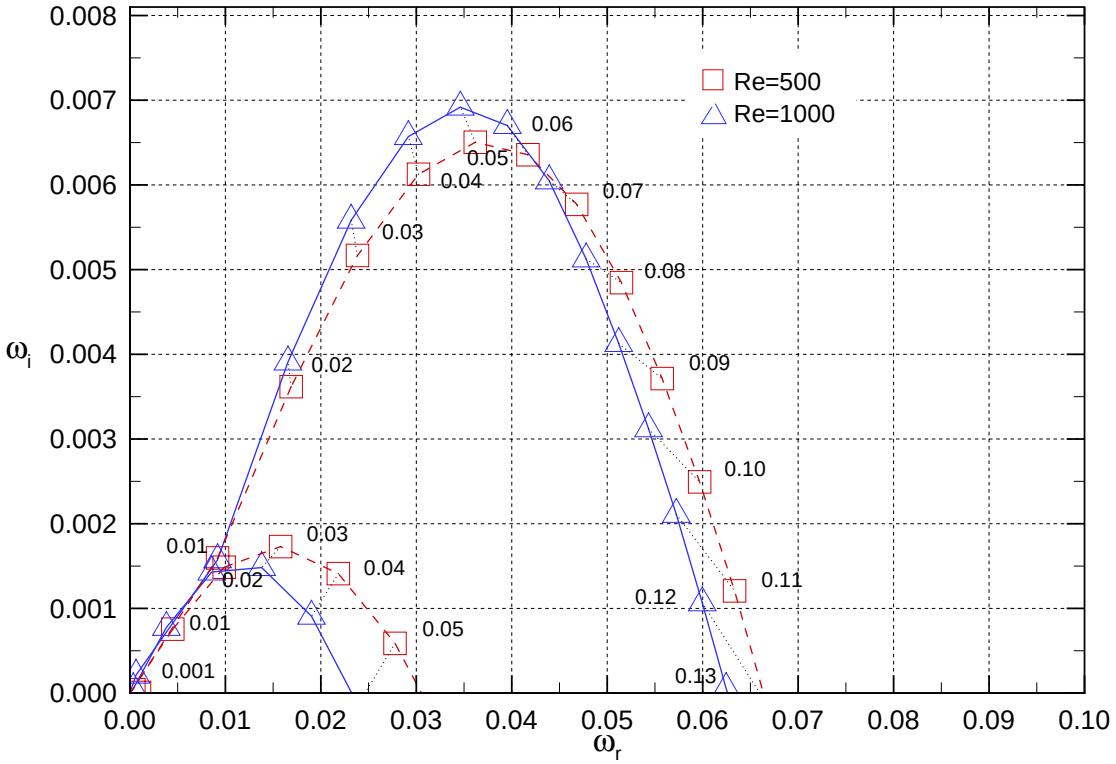

FIGURE 5.9. – *Instable eigenvalues of the simulated flow at different* REYNOLDS *numbers.*

5.5. Further Reading

The following literature was also reviewed during the study in preparation of the DNS stability analysis that could not be carried out. It contains information that will be relevant when this analysis can be conducted.

Chomaz (2003)
Crawford (2004)
Criminale et al. (2003)
Davies (2005)
Davies and Carpenter (2001)
Delbende and Chomaz (1998)
Dratler and Fasel (1996)
Drazin (2002)
Ferziger and Perić (2002)
Goldstein (1930)
Hannemann and Oertel (1989)
Huerre (2002)
Huerre and Rossi (1998)
Jobe and Burggraf (1974)
Maekawa, Masour, and Buell (1992)
Mattingly and Criminale (1972)
Messiter (1970)
Monkewitz (1988)
Oertel (1990)
Prandtl (1904)
Rist and Fasel (1995)
Rothmayer and Smith (1998)
Schlichting and Gersten (2006)
Schmid and Henningson (2001)

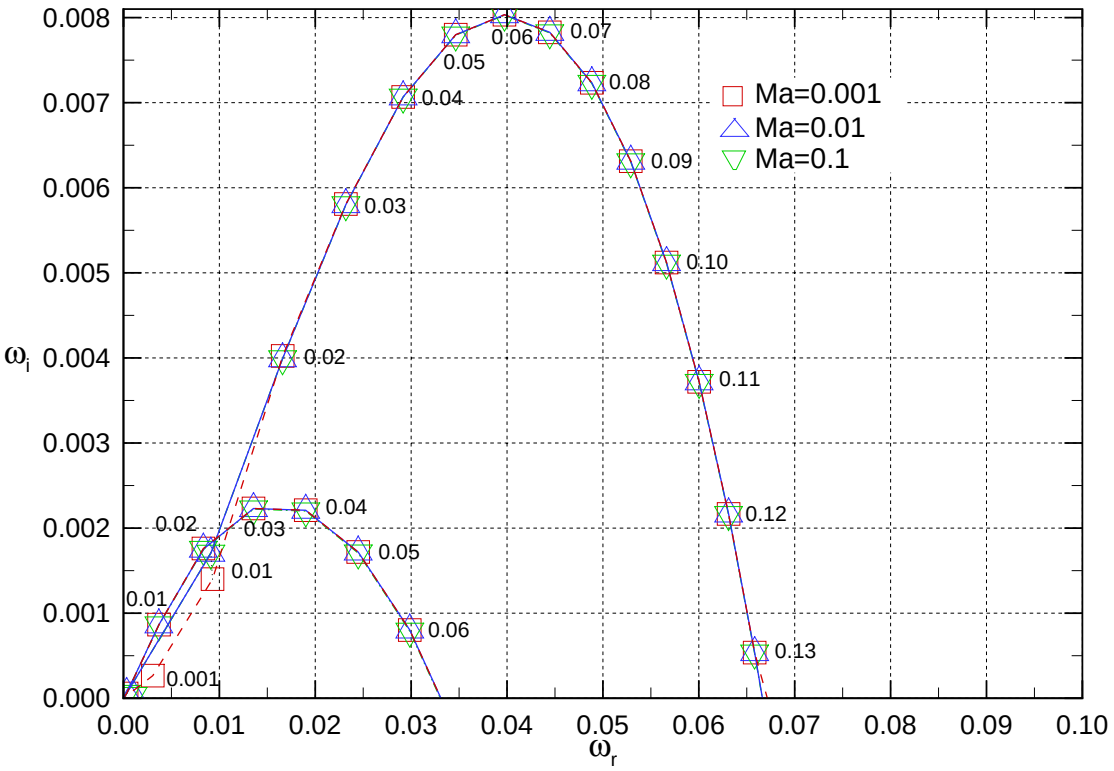

FIGURE 5.10. – *Instable eigenvalues of the prescribed flow at* $Re = 1000$, $Q = 0.75$ *and different* MACH *numbers.*

Smith (1982)
Smith, Bowles, and Li (2000)
Stewartson (1969)
Taylor and Peake (1999)
Turkyilmazaglu, Gajjar, and Ruban (1999)
Woodley and Peake (1997)
Zabusky and Deem (1971)

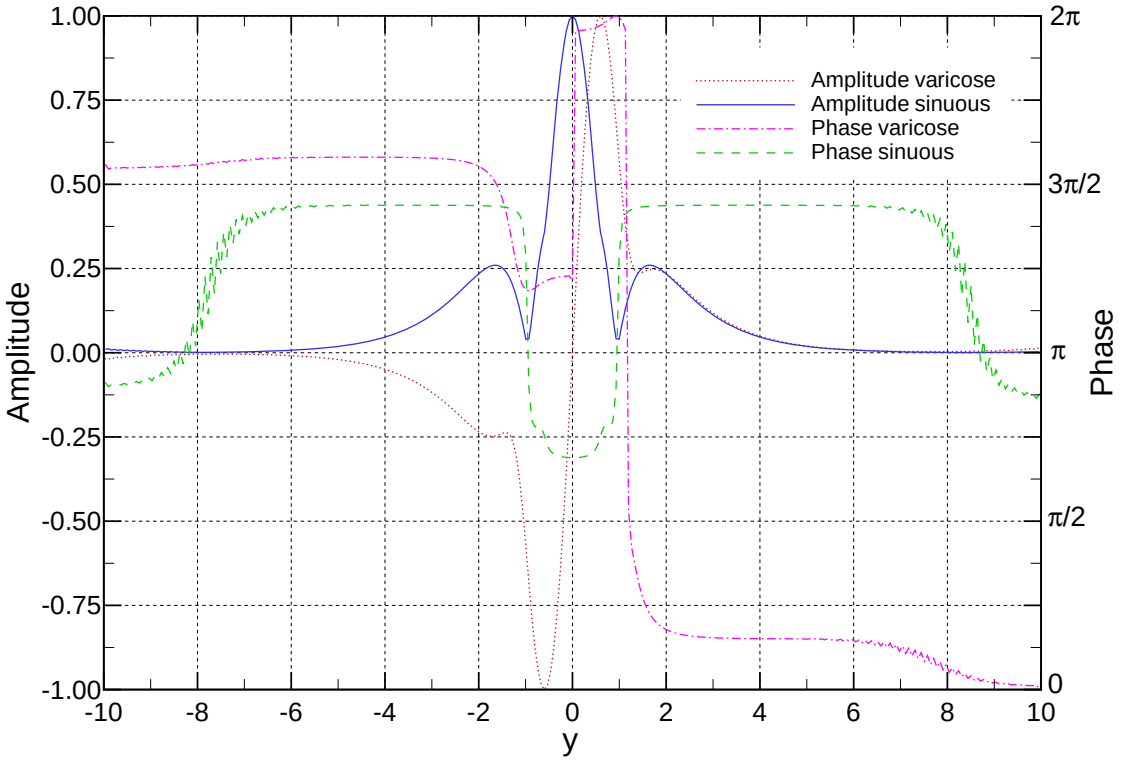

FIGURE 5.11. – *Amplitude and phase of the eigenfunction of the streamwise velocity u in the prescribed mean flow for the most instable varicose eigenvalue $\omega_r = 4.3788 \cdot 10^{-2}$, $\omega_i = 6.4259 \cdot 10^{-3}$ and the associated sinuous eigenvalue $\omega_r = 3.5888 \cdot 10^{-2}$, $\omega_i = 6.3778 \cdot 10^{-4}$ at $\alpha_r = 6 \cdot 10^{-2}$, $Re = 1000$, $Q = 0.6$.*

Simulated					Prescribed				
	Varicose		Sinuous			Varicose		Sinuous	
α_r	ω_r	ω_i	ω_r	ω_i	α_r	ω_r	ω_i	ω_r	ω_i
$\cdot 10^2$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^3$
0.1	0.100	0.041	0.000	0.000	0.1	0.038	0.036	0.000	0.000
1	0.918	1.596	0.449	0.755	1	0.923	1.476	0.429	0.764
2	1.689	3.618	0.987	1.486	2	1.690	3.623	0.942	1.564
3	2.384	5.167	1.579	1.730	3	2.384	5.259	1.511	1.997
4	3.023	6.125	2.183	1.411	4	3.019	6.405	2.098	1.981
5	3.617	6.503	2.777	0.587	5	3.609	7.065	2.688	1.536
6	4.169	6.356	3.349	-0.062	6	4.164	7.276	3.269	0.705
7	4.680	5.770	3.903	-0.211	7	4.686	7.086	3.836	-0.470
8	5.149	4.846			8	5.177	6.546		
9	5.576	3.713			9	5.638	5.708		
10	5.968	2.494			10	6.070	4.626		
11	6.334	1.206			11	6.472	3.352		
12	6.673	-0.148			12	6.843	1.943		
13	6.980	-1.516			13	7.185	0.454		
14	7.241	-2.593			14	7.497	-1.053		

Ratio prescribed/simulated					Delta prescribed-simulated				
	Varicose		Sinuous			Varicose		Sinuous	
α_r	ω_r	ω_i	ω_r	ω_i	α_r	ω_r	ω_i	ω_r	ω_i
$\cdot 10^2$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^2$	$\cdot 10^3$	$\cdot 10^2$	$\cdot 10^3$
0.1	0.38	0.88	1.00	1.00	0.1	-0.06	-0.01	0.00	0.00
1	1.01	0.92	0.96	1.01	1	0.01	-0.12	-0.02	0.01
2	1.00	1.00	0.95	1.05	2	0.00	0.01	-0.05	0.08
3	1.00	1.02	0.96	1.15	3	0.00	0.09	-0.07	0.27
4	1.00	1.05	0.96	1.40	4	0.00	0.28	-0.09	0.57
5	1.00	1.09	0.97	2.62	5	-0.01	0.56	-0.09	0.95
6	1.00	1.14	0.98	-11.37	6	0.00	0.92	-0.08	0.77
7	1.00	1.23	0.98	2.23	7	0.01	1.32	-0.07	-0.26
8	1.01	1.35			8	0.03	1.70		
9	1.01	1.54			9	0.06	2.00		
10	1.02	1.85			10	0.10	2.13		
11	1.02	2.78			11	0.14	2.15		
12	1.03	-13.13			12	0.17	2.09		
13	1.03	-0.30			13	0.20	1.97		
14	1.04	0.41			14	0.26	1.54		

TABLE 5.1. – *Eigenvalues of the simulated and the prescribed ($Q = 0.6799$) flow at $Re = 500$ and corresponding ratios and differences.*

α_r $\cdot 10^2$	n	Q	ω_r $\cdot 10^2$	ω_i $\cdot 10^3$	γ $\cdot 10^3$	$\frac{\gamma_{n-1}}{\gamma_n}$	$\frac{Q_1-Q_2}{Q_2-Q_3}$
3	1	0.6	2.456	4.634			
	2	0.6799	2.384	5.259	0.953		
	3	0.75	2.320	5.800	0.838	1.138	
6	1	0.6	4.380	6.420			
	2	0.6799	4.164	7.276	2.323		
	3	0.75	3.974	8.028	2.043	1.137	1.140
9	1	0.6	6.034	5.035			
	2	0.6799	5.638	5.708	4.017		
	3	0.75	5.292	6.300	3.510	1.144	
12	1	0.6	7.450	1.710			
	2	0.6799	6.843	1.943	6.074		
	3	0.75	6.311	2.147	5.324	1.141	

TABLE 5.2. – *Selected instable eigenvalues of the prescribed flow at* $Re = 500$ *and different profile velocitiy ratios –* γ *is the* EUCLIDEAN *distance of the eigenvalues:* $\gamma = \sqrt{(\omega_{r\,n-1} - \omega_{r\,n})^2 + (\omega_{i\,n-1} - \omega_{i\,n})^2}$.

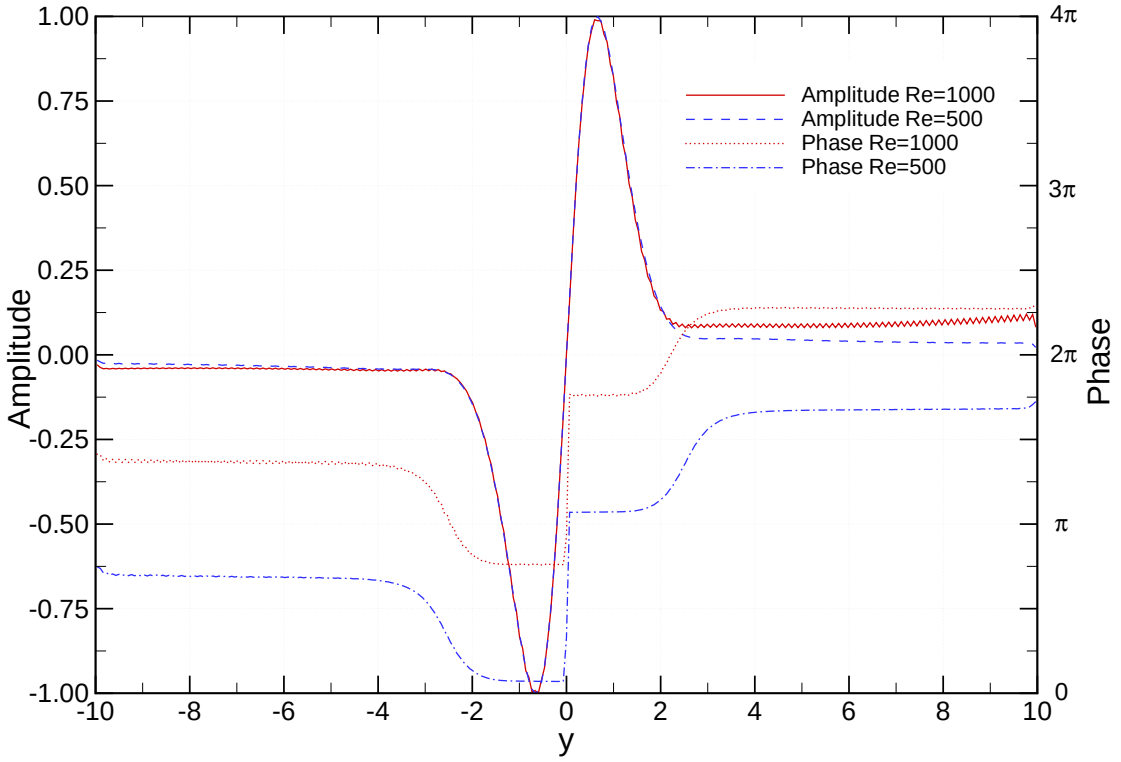

FIGURE 5.12. – *Amplitude and phase of the eigenfunction of the streamwise velocity u in the prescribed mean flow at $Q = 0.75$, $\alpha_r = 1 \cdot 10^{-2}$ and two* REYNOLDS *numbers for the respective varicose eigenvalues.*

6. Conclusion and Outlook

It could be shown that the `PCNAVWAKEBD` code in principle can be used to carry out the investigation of disturbance evolution that was the scope of this thesis. The numerical instability that occurred when introducing the perturbations was the major obstacle to accomplish this and could not be overcome because the code provided is way too complex to be completely analysed as a by-product. It should be pursued in a separate work as to where the exact roots of this instability lie – this thesis shed light on some possible reasons and worked out which parameters could have an influence and which not. In particular, the boundary conditions and the irregular velocity profile of U vs. y on the small scale should be focused on in the future.

Still, for a range of x-stations, the code produces a mean flow state that can serve as an input to an eigenvalue-oriented stability analysis. Using this and the analytic sech^2-type prescribed mean flow, stability properties of the parallel near-wake flow could be examined thoroughly. The main findings were that

- simulated and prescribed mean flow have nearly identical eigenvalues for lower wavenumbers
- eigenvalues of simulated and prescribed mean flow for higher wavenumbers seem to correlate systematically
- there is no Reynolds number dependence of the eigenvalues for a constant velocity profile
- there is no Mach number dependence of the eigenvalues in the range of $Ma = 0.001 \ldots 0.1$
- there is a linear relation between the locations of the eigenvalues and the velocity profile ratio.

When the direct numerical simulation will be available and after translating the temporal stability analysis done here into the spatial analysis, then the next steps to bring the investigation of disturbance evolution in the wake of a flat plate further will be to make comparisons between the results from simulation and from eigenvalue solving. For this, a comprehensive basis was established with this work.

Part III.

Résumé

Résumé

The investigation of the disturbance evolution in the near-wake behind a flat plate is approached via two different ways in this thesis. The direct numerical simulation of the flow, subject to specified perturbations, is one method. Yet, this method could not be brought to an end because the `PCNAVWAKEBD` code available for this simulation could not be made to run entirely stable. The second technique is to solve the Orr-Sommerfeld equation for both a wake flow obtained from numerical simulation and a wake flow defined by an analytical function of the sech^2-type.

The principal outline of this approach in this thesis is the following:

After presenting the basic flow properties, the governing equations for the flow and basics of fluid flow stability, the numerical approach and the solution scheme of the `PCNAVWAKEBD` code are briefly reviewed. Furthermore, changes and extensions to this code as well as separate programmes written for post-processing the data obtained from running `PCNAVWAKEBD` are presented.

Subsequently it is shown under which settings the simulations were actually carried out to support the search for the root of the artificial instability. Additionally, a method how the wavenumber of the perturbation could have been extracted from the flow is presented, together with re-dimensionalisations that give an impression of what the physical values of the results would be.

The basic state without introduction of disturbances could be simulated without numerical instabilities and thus can be used to examine the quality and credibility of the flow the `PCNAVWAKEBD` code produces. This is done by comparing the numerical results with what asymptotical theory (triple-deck theory, which is reviewed in short) predicts for the delicate region around the trailing edge of the flat plate. The accordance of results from these two sources shows to be generally satisfactory.

Finally, the approach to solving the eigenvalue problem for the two basic flow types using the programme `LINSTAB` is outlined and some additional findings about the simulated mean flow are given. The results of the eigenvalue solving are presented, the stability of the two wake flow types is compared and dependence of the stability properties from several parameters is discussed. Foremost it was found that there is a linear relationship between the profile velocity ratio and the location of an eigenvalue for a given wavenumber.

Bibliography

Babucke, A. (n.d.). *Lineare Stabilitätstheorie & Matrixlöser.* http://www.iag.uni-stuttgart.de/people/andreas.babucke/linstab.html. (accessed: 17/05/2007)

Chomaz, J.-M. (2003). Fully nonlinear dynamics of parallel wakes. *J. Fluid Mech.*, *495*, 57-75.

Crawford, D. (2004). *gnuplot – an interactive plotting program* (Software manual).

Criminale, W. O., Jackson, T. L., & Joslin, R. D. (2003). *Theory and computation in hydrodynamic stability.* Cambridge: University Press.

Davies, C. (2005). Numerical simulation of boundary-layer disturbance evolution. *Philosophical Transactions of The Royal Society A*, *363*, 1109-1118.

Davies, C., & Carpenter, P. W. (2001). A novel velocity-vorticity formulation of the navier-stokes equations with applications to boundary layer disturbance evolutions. *Journal of Computational Physics*, *172*, 119-165.

Delbende, I., & Chomaz, J.-M. (1998). Nonlinear convective/absolute instabilities in parallel two-dimensional wakes. *Physics of Fluids*, *10*(11), 2724-2736.

Dratler, D. I., & Fasel, H. F. (1996). Spatial evolution of a monochromatically forced flat-plate wake. *AIAA Journal*, *34*(11), 2299-2305.

Drazin, P. G. (2002). *Introduction to hydrodynamic stability.* Cambridge University Press.

Fasel, H., Rist, U., & Konzelmann, U. (1990). Numerical investigation of the three-dimensional development in boundary-layer transition. *AIAA Journal*, *28*(1), 29-37.

Ferziger, J. H., & Perić, M. (2002). *Computational methods for fluid dynamics* (3 ed.). Berlin Heidelberg: Springer.

Goldstein, S. (1930). *Proc. Camb. phil. Soc.*, *26*, 1.

Hannemann, K., & Oertel, H. (1989). Numerical simulation of the absolutely and convectively unstable wake. *J. Fluid Mech.*, *199*, 55-88.

Heaney, C. (2007). *Numerical simulation of wavepackets in a transitional boundary layer.* Unpublished doctoral dissertation, Cardiff University School of Mathematics.

Huerre, P. (2002). Perspectives in fluid dynamics. In G. K. Batchelor, H. K. Moffatt, & M. G. Worster (Eds.), (2nd ed., p. 159-229). Cambridge University Press.

Huerre, P., & Rossi, M. (1998). Hydrodynamics and nonlinear instabilities. In C. Godrèche & P. Manneville (Eds.), (p. 81-294). Cambridge University Press.

Jobe, C. E., & Burggraf, O. R. (1974). The numerical solution of the asymptotic equations of trailing edge flow. *Proc. R. Soc. Lond. A.*, *340*, 91-111.

Kloker, M., Konzelmann, U., & Fasel, H. (1993, April). Outflow boundary conditions for spatial navier-stokes simulations of transition boundary layers. *AIAA Journal*, *31*(4), 620-628.

Maekawa, H., Masour, N. N., & Buell, J. C. (1992). Instability mode interactions in a spatially developing plane wake. *J. Fluid Mech.*, *235*, 223-254.

Mattingly, G. E., & Criminale, W. O. (1972). The stability of an incompressible two-dimensional wake. *J. Fluid Mech.*, *51(2)*, 233-272.

Messiter, A. F. (1970). *SIAM J. appl. Math.*, *90*, 625.

Monkewitz, P. A. (1988). The absolute and convective nature of instability in two-dimensional wakes at low reynolds numbers. *Phys. Fluids*, *31*(5), 999-1006.

Oertel, H. (1990). Wakes behind blunt bodies. *Annu. Ref. Fluid Mech.*, *22*, 539-64.

Prandtl, L. (1904). Über Flüssigkeitsbewegung bei sehr kleiner Reibung. In *Verhandlg. III. Intern. Math. Kongr. Heidelberg* (p. 484-491).

Rist, U., & Fasel, H. (1995). Direct numerical simulation of controlled transition in a flat-plate boundary layer. *J. Fluid Mech.*, *298*, 211-248.

Rothmayer, A. P., & Smith, F. T. (1998). High reynolds number asymptotic theories. In R. W. Johnson (Ed.), (p. 23-1-25-25). CRC Press.

Schlichting, H., & Gersten, K. (2006). *Grenzschicht-Theorie* (10 ed.). Berlin, Heidelberg: Springer.

Schmid, P. J., & Henningson, D. S. (2001). *Stability and transition in shear flows*. New York: Springer.

Smith, F. T. (1982). On the high reynolds number theory of laminar flows. *Journal of Applied Mathematics*, *28*, 207-281.

Smith, F. T., Bowles, R. G. A., & Li, L. (2000). Nonlinear effects in absolute and convective insta-bilities of a near-wake. *Eur. J. Mech. B - Fluids*, *19*, 173-211.

Stewartson, K. (1969). *Mathematica*, *16*, 106.

Taylor, M. J., & Peake, N. (1999). A note on the absolute instability of wakes. *Eur. J. Mech. B/Fluids*, *18*(4), 573-579.

Turkyilmazaglu, M., Gajjar, J. S. B., & Ruban, A. I. (1999). The absolute instability of thin wakes in an incompressible/compressible fluid. *Theoret. Comput. Fluid Dynamics*, *13*, 91-114.

Woodley, B. M., & Peake, N. (1997). Global linear stability analysis of thin aerofoil wakes. *J. Fluid Mech.*, *339*, 239-260.

Zabusky, N. J., & Deem, G. S. (1971). Dynamical evolution of two-dimensional unstable shear flows. *J. Fluid Mech.*, *47*, 353-379.

Appendix

A.1. PRESCMF Module Code

```
subroutine PRESCMF(UB,D2UB,VBC,DUBW,LY,YPTS,LETA,NPROF,LAMBDA
    )

integer YM1,YPTS
parameter (YM1=64)
double precision UB(YM1),D2UB(YM1),VBC(YM1),DUBW
double precision LETA,LY,DISP
double precision PI,Y(YM1),ZETA(YM1)
integer I,J

double precision LAMBDA
integer NPROF
double precision T1,T3,T4,T5,T7,T8,T9,T13,T15,T16,T17,T19,T20
    ,T22,T23,T27,T28

DISP=1.d0
LY=LETA/DISP

PI=4.d0*datan(1.d0)

do J=1,YPTS
VBC(J)=0.d0
end do

DUBW=0.d0

C For the function U(Y)=1-LAMBDA*(sech(Y))**2
do I=1,YPTS
ZETA(I) = dcos(PI*(I-1.d0)/(2.d0*YPTS))
Y(I) = LETA*(1.d0/(ZETA(I))-1.d0)
UB(I) = 1.d0-LAMBDA*(2.d0/(dexp(Y(I))+dexp(-Y(I))))**2
```

```
C      = 1-LAMBDA*(sech(Y))**2
C DUB(I) = 2.d0*LAMBDA*(2.d0/(dexp(Y(I))+dexp(-Y(I))))**2*
    dtanh(Y(I))
C         = 2*LAMBDA*(sech(Y))**2*tanh(Y)
D2UB(I) = 2.d0*LAMBDA*(-2.d0*(2.d0/(dexp(Y(I))+dexp(-Y(I))))
    **2*(dtanh(Y(I)))**2+
& (2.d0/(dexp(Y(I))+dexp(-Y(I))))**2*(1.d0-(dtanh(Y(I)))**2))
C        = 2*LAMBDA*(-2*(sech(Y))**2*(tanh(Y))**2+(sech(Y))
    **2*(1-(tanh(Y))**2))
end do

CC For the function U(Y)=(1-LAMBDA+2*LAMBDA*(1+(sinh(Y*arsinh
    (1)))**(2*NPROF))**(-1.d0))
C do I=1,YPTS
C  ZETA(I) = dcos(PI*(I-1.d0)/(2.d0*YPTS))
C  Y(I) = LETA*(1.d0/(ZETA(I))-1.d0)
CC  UB(I) = (1-LAMBDA+2*LAMBDA*(1+(dsinh(Y(I)*DLOG(1.d0+dsqrt
    (2.d0))))**(2*NPROF))**(-1.d0))/2.d0
C  UB(I) = (1-LAMBDA+2*LAMBDA*(1+(dsinh(Y(I)*DLOG(1.d0+dsqrt
    (2.d0))))**(2*NPROF))**(-1.d0))
C end do
C
C do I=1,(YPTS-1)
C  if (I .eq. 1) then
C   if (NPROF .eq. 1) then
C    D2UB(I) = -0.155363880d0*LAMBDA
C   else
C    D2UB(I) = 0.d0
C   end if
C  else
C   T1 = dsqrt(2.d0)
C   T3 = dlog(1.d0+T1)
C   T4 = Y(I)*T3
C   T5 = dsinh(T4)
C   T7 = T5**(2*NPROF)
C   T8 = 1.d0+T7
C   T9 = T8**2
C   T13 = T7**2
C   T15 = NPROF**2
C   T16 = dcosh(T4)
C   T17 = T16**2
C   T19 = T3**2
C   T20 = T5**2
```

```
C    T22 = T19/T20
C    T23 = T15*T17*T22
C    T27 = LAMBDA/T9
C    T28 = T27*T7
CC   D2UB(I) = 0.8d0*LAMBDA/T9/T8*T13*T23-0.4d0*T28*T23-0.2d0*
     T27*T7*NPROF*T19+0.2d0*T28*NPROF*T17*T22
C   end if
C end do
C          D2UB(YPTS) = 0.d0

CC For the function U(Y)=1
C do I=1,YPTS
C   ZETA(I) = dcos(PI*(I-1.d0)/(2.d0*YPTS))
C   Y(I) = LETA*(1.d0/(ZETA(I))-1.d0)
C   UB(I) = 1.d0
C   D2UB(I) = 0.d0
C end do

CC For the function U(Y)=Y*A
C do I=1,YPTS
C   ZETA(I) = dcos(PI*(I-1.d0)/(2.d0*YPTS))
C   Y(I) = LETA*(1.d0/(ZETA(I))-1.d0)
C   if (Y(I).lt.1.d0) then
C    UB(I) = 1.d0*Y(I)
C   else
C    UB(I) = 1.d0
C   end if
C   D2UB(I) = 0.d0
C end do

end subroutine
```

A.2. Unix Shell Script for Launching Condor Grid Computing Jobs

A.2.1. batchcmd.sh

```
rm condorbatch
echo "universe = vanilla" >> condorbatch
echo "getenv = true" >> condorbatch
echo "log = log.log" >> condorbatch
echo "error = error.log" >> condorbatch
```

```
6 echo "executable␣=␣pcnavwakebdfs" >> condorbatch
7 echo "WhenToTransferOutput␣=␣ON_EXIT" >> condorbatch
8 for R in 100. 1000.
9 do
10  for AF in 1.d-10
11  do
12   for w in 0.001 0.005 0.01 0.05 0.1 0.5 1.0 5.0 10.
13   do
14    for LXF in 5.0
15    do
16     for Q in 0.6
17     do
18      for alph in 0.05
19      do
20       for NY in 64
21       do
22        for DX in 0.5
23        do
24         for DT in 0.125
25         do
26          for NITS in 5
27          do
28           for DISP in 1.0
29           do
30 mkdir ${R}_${w}_${Q}_${LXF}_${AF}_${DX}_${DT}_${NITS}
31 cp inputbatch.dat inputtemp.dat
32 sed -e "s:@R@:${R}:" inputtemp.dat > inputtemp1.dat
33 sed -e "s:@NY@:${NY}:" inputtemp1.dat > inputtemp2.dat
34 sed -e "s:@alph@:${alph}:" inputtemp2.dat > inputtemp3.dat
35 sed -e "s:@DX@:${DX}:" inputtemp3.dat > inputtemp4.dat
36 sed -e "s:@DT@:${DT}:" inputtemp4.dat > inputtemp5.dat
37 sed -e "s:@NITS@:${NITS}:" inputtemp5.dat > inputtemp6.dat
38 sed -e "s:@AF@:${AF}:" inputtemp6.dat > inputtemp7.dat
39 sed -e "s:@w@:${w}:" inputtemp7.dat > inputtemp8.dat
40 sed -e "s:@LXF@:${LXF}:" inputtemp8.dat > inputtemp9.dat
41 sed -e "s:@Q@:${Q}:" inputtemp9.dat > inputtemp10.dat
42 sed -e "s:@DISP@:${DISP}:" inputtemp10.dat > inputtemp11.dat
43 cp inputtemp11.dat ./${R}_${w}_${Q}_${LXF}_${AF}_${DX}_${DT}
     _${NITS}/input.dat
44 echo "initialdir␣=␣${R}_${w}_${Q}_${LXF}_${AF}_${DX}_${DT}_${
     NITS}" >> condorbatch
45 echo "input␣=␣input.dat" >> condorbatch
46 echo "queue" >> condorbatch
```

```
47            done
48           done
49          done
50         done
51        done
52       done
53      done
54     done
55    done
56  done
57 done
58 rm inputtemp*.dat
```

A.2.2. inputbatch.dat

```
1 @R@
2 0.
3 (@alph@,0)
4 @DX@
5 @DT@
6 800.
7 1000.
8 1000.
9 @NITS@
10 @NY@
11 4.
12 300.
13 @AF@
14 @w@
15 @LXF@
16 1
17 1
18 0
19 1
20 1
21 @Q@
22 @DISP@
```

A.3. Miscellaneous Changes to the PCNAVWAKE Programme Code

A.3.1. outputall Module Code

```
1   subroutine outputall(TT,VEL,TVEL,TTVEL,VOR,DX,DT,R,XEND,
       LMAP,NY,SU,TP)
2 C        output physical data fields
3   integer N,XG,NY,XEND,NY2
4   parameter (N=64,XG=2048)
5
6   double complex VEL(N,XG),TVEL(N,XG),TTVEL(XG),VOR(N,XG)
7   double complex UPTS(N,XG),VPTS(N,XG),VORPTS(N,XG),PPTS(N,XG
       )
8   double precision US(N,XG),VS(N,XG),VORS(N,XG)
9   double complex UTCOLL(N,XG),D2UTCOLL(N,XG)
10  double precision UCR,VCR,VORCR
11  double precision UCI,VCI,VORCI
12  double precision UCT,VCT,VORCT
13  double precision UCRP,VCRP,VORCRP
14  double precision UCIP,VCIP,VORCIP
15  double precision UCTP,VCTP,VORCTP
16  double precision PCR,PCI,PCT
17  double precision TT,DX,DT,R,LMAP
18  double complex IVEL(XG),DIVEL(XG),IVOR(XG),IVELS(XG),IVORS(
       XG)
19  double precision YPOS(N),PI,YREAL(N)
20  double precision VI,VR,WI,WR
21
22  integer XJ,YJ
23  logical SU,TP
24
25  double precision XPOS,DU
26  double complex UCENT(XG)
27  double precision A,B
28
29        open (1,file='collinit.dat')
30        do XJ=1,XEND
31         read(1,*)
32         do YJ=1,NY
33          read(1,*) A,B,US(YJ,XJ),VS(YJ,XJ),VORS(YJ,XJ)
34         end do
35        end do
36        close (1)
37
38 C      calculate collocation values
39        call ucollptsfft(VEL,VOR,UPTS,DX,XEND,LMAP,NY)
40        call vcollptsfft(VEL,VPTS,XEND,NY)
```

```
41          call vcollptsfft(VOR,VORPTS,XEND,NY)
42
43          call utotcoll (VEL,VOR,UTCOLL,D2UTCOLL,DX,XEND,LMAP,
               NY)
44
45          call nlpcollptsfft (VEL,TVEL,TTVEL,VOR,PPTS,DX,DT,R,
               XEND,LMAP,NY)
46
47
48          PI=4.d0*datan(1.D0)
49
50  101     format(A)
51  100     format(F6.1,'  ',F18.15,'  ',F19.15,'  ',E22.15,'  ',
    E22.15,'  ',E22.15,'  ',E22.15,'  ',E22.15,'  ',E22.15,'
    ',E22.15,'  '
52        & , E22.15,'  ',E22.15,'  ',E22.15,'  ',E22.15,'  ',E22
             .15,'  ',E22.15,'  ',E22.15)
53
54          do  YJ=1,NY
55           YPOS(YJ)=cos((YJ-1)*PI/(2*NY))
56           YREAL(YJ)=LMAP*(1.d0/YPOS(YJ)-1.d0)
57          end do
58
59  C       and print them out
60          open (1,file='sym-asym.dat')
61          if(TP) then
62           write(1,101) 'VARIABLES = "x","1-zeta","y","u_sym","
                v_sym","vor_sym","u_asym","v_asym","vor_asym","
                u_p_sym","v_p_sym",
63        & "vor_p_sym","u_p_asym","v_p_asym","vor_p_asym","p_sym"
             ,"p_asym"'
64           write(1,*) 'ZONE I=',NY,' J=',XEND
65          else
66           write(1,101) '#VARIABLES = "x","1-zeta","y","u_sym",
                "v_sym","vor_sym","u_asym","v_asym","vor_asym","
                u_p_sym","v_p_sym",
67        & "vor_p_sym","u_p_asym","v_p_asym","vor_p_asym","p_sym"
             ,"p_asym"'
68           write(1,*) '#ZONE I=',NY,' J=',XEND
69           write(1,*)
70          end if
71
72          do XJ=1,XEND
```

```
73           XPOS=(XJ-1)*DX
74           do YJ=1,NY
75            UCR=dreal(UPTS(YJ,XJ))
76            VCR=dreal(VPTS(YJ,XJ))
77            VORCR=dreal(VORPTS(YJ,XJ))
78            UCI=dimag(UPTS(YJ,XJ))
79            VCI=dimag(VPTS(YJ,XJ))
80            VORCI=dimag(VORPTS(YJ,XJ))
81            UCRP=dreal(UPTS(YJ,XJ))-US(YJ,XJ)
82            VCRP=dreal(VPTS(YJ,XJ))-VS(YJ,XJ)
83            VORCRP=dreal(VORPTS(YJ,XJ))-VORS(YJ,XJ)
84            UCIP=dimag(UPTS(YJ,XJ))-US(YJ,XJ)
85            VCIP=dimag(VPTS(YJ,XJ))-VS(YJ,XJ)
86            VORCIP=dimag(VORPTS(YJ,XJ))-VORS(YJ,XJ)
87            PCR=dreal(PPTS(YJ,XJ))
88            PCI=dimag(PPTS(YJ,XJ))
89
90     write(1,100) real(XPOS),1.d0-real(YPOS(YJ)),YREAL(YJ),real(
          UCR),real(VCR),real(VORCR),real(UCI),real(VCI),real(
          VORCI),real(UCRP),
91        & real(VCRP),real(VORCRP),real(UCIP),real(VCIP),real(
             VORCIP),
92        & real(PCR),real(PCI)
93            end do
94           end do
95           close (1)
96
97 C       print out total field
98           open (1,file='total.dat')
99            NY2=2*NY
100          if(TP) then
101           write(1,101) 'VARIABLES = "XPOS","1-zeta","y","u_t",
                 "v_t","vor_t","u_p_t","v_p_t","vor_p_t","u_tot","
                 d2u_tot","p_t"'
102           write(1,*) 'ZONE I=',NY2,' J=',XEND
103          else
104           write(1,101) '#VARIABLES = "XPOS","1-zeta","y","u_t"
                 ,"v_t","vor_t","u_p_t","v_p_t","vor_p_t","u_tot",
                 "d2u_tot","p_t"'
105           write(1,*) '#ZONE I=',NY2,' J=',XEND
106           write(1,*)
107          end if
108
```

```
109         do XJ=1,XEND
110          XPOS=(XJ-1)*DX
111          do YJ=NY,1,-1
112           UCT=dreal(UPTS(YJ,XJ))+dimag(UPTS(YJ,XJ))
113           VCT=dreal(VPTS(YJ,XJ))+dimag(VPTS(YJ,XJ))
114           VORCT=dreal(VORPTS(YJ,XJ))+dimag(VORPTS(YJ,XJ))
115
116           UCTP=dreal(UPTS(YJ,XJ))+dimag(UPTS(YJ,XJ))-US(YJ,XJ
                 )
117           VCTP=dreal(VPTS(YJ,XJ))+dimag(VPTS(YJ,XJ))-VS(YJ,XJ
                 )
118           VORCTP=dreal(VORPTS(YJ,XJ))+dimag(VORPTS(YJ,XJ))-
                 VORS(YJ,XJ)
119
120           PCT=dreal(PPTS(YJ,XJ))+dimag(PPTS(YJ,XJ))
121           write(1,100) real(XPOS),1.d0-real(YPOS(YJ)),YREAL(
                 YJ),real(UCT),real(VCT),real(VORCT),real(UCTP),
                 real(VCTP),real(VORCTP),
122      & real(dreal(UTCOLL(YJ,XJ))),real(dreal(D2UTCOLL(YJ,XJ))
         ),real(PCT)
123          end do
124          do YJ=1,NY
125           UCT=dreal(UPTS(YJ,XJ))-dimag(UPTS(YJ,XJ))
126           VCT=-dreal(VPTS(YJ,XJ))+dimag(VPTS(YJ,XJ))
127           VORCT=-dreal(VORPTS(YJ,XJ))+dimag(VORPTS(YJ,XJ))
128
129           UCTP=dreal(UPTS(YJ,XJ))-dimag(UPTS(YJ,XJ))-US(YJ,XJ
                 )
130           VCTP=-dreal(VPTS(YJ,XJ))+dimag(VPTS(YJ,XJ))+VS(YJ,
                 XJ)
131           VORCTP=-dreal(VORPTS(YJ,XJ))+dimag(VORPTS(YJ,XJ))+
                 VORS(YJ,XJ)
132
133           PCT=dreal(PPTS(YJ,XJ))-dimag(PPTS(YJ,XJ))
134           write(1,100) real(XPOS),real(YPOS(YJ))-1.d0,-YREAL(
                 YJ),real(UCT),real(VCT),real(VORCT),real(UCTP),
                 real(VCTP),real(VORCTP),
135      & real(dreal(UTCOLL(YJ,XJ))),real(dreal(D2UTCOLL(YJ,XJ))
         ),real(PCT)
136          end do
137         end do
138         close (1)
139
```

```
140 111      format(A)
141 110      format(F6.1,'  ',I3,'  ','(',E22.15,',',E22.15,')','
        ','(',E22.15,',',E22.15,')')
142
143 C        print out chebvectors for primary variables
144          open (1,file='cheb.dat')
145          write(1,*) '#TIME',TT
146          write(1,*) '#CHEB COEFTS'
147          write(1,*) '#xpos, yj, v, vor'
148          do XJ=1,XEND
149           write(1,*)
150           XPOS=(XJ-1)*DX
151           do YJ=1,NY
152            write(1,110) real(XPOS),YJ,VEL(YJ,XJ),VOR(YJ,XJ)
153           end do
154          end do
155          close (1)
156
157 121      format(A)
158 120      format(F6.1,'  ',E22.15,'  ',E22.15,'  ',E22.15,'  ',
        E22.15)
159
160 C        wall values
161          open (1,file='wallvals.dat')
162          if(TP) then
163           write(1,121) 'VARIABLES = "XPOS","u_centcd␣","u","v"
                ,"vor"'
164           write(1,*) 'ZONE I=',XEND
165          else
166           write(1,*) '#TIME',TT
167           write(1,121) '#VARIABLES = "XPOS","u_centcd␣","u","v
                ","vor"'
168           write(1,*) '#ZONE I=',XEND
169          end if
170
171 C checks the vorticity boundary condition
172          call profinteg(VOR,IVOR,XEND,NY)
173          call profinteg(VEL,IVEL,XEND,NY)
174          call sderiv(IVEL,DIVEL,DX,XEND)
175
176          do XJ=1,XEND
177            UCENT(XJ)=-(IVOR(XJ)+DIVEL(XJ))
178          end do
```

```
179
180         do XJ=1,XEND
181          XPOS=(XJ-1)*DX
182          write(1,120) real(XPOS),real(dreal(UCENT(XJ))),real(
                dreal(UPTS(1,XJ))),real(dreal(VPTS(1,XJ))),real(
                dreal(VORPTS(1,XJ)))
183         end do
184
185         write(1,*)
186         if(TP) then
187          write(1,121) 'VARIABLES = "XPOS","u_centcd␣","u","v"
                ,"vor"'
188          write(1,*) 'ZONE I=',XEND
189         else
190          write(1,*) '#TIME',TT
191          write(1,121) '#VARIABLES = "XPOS","u_centcd␣","u","v
                ","vor"'
192          write(1,*) '#ZONE I=',XEND
193         end if
194         do XJ=1,XEND
195          XPOS=(XJ-1)*DX
196          write(1,120) real(XPOS),real(dimag(UCENT(XJ))),real(
                dimag(UPTS(1,XJ))),real(dimag(VPTS(1,XJ))),real(
                dimag(VORPTS(1,XJ)))
197         end do
198         close(1)
199
200   end subroutine
201
202   subroutine outputcent(TT,VEL,TVEL,TTVEL,VOR,DX,DT,R,XEND,
         LMAP,NY,SU,TP,XAMP,ABORT)
203 C        output physical data fields
204
205   integer N,XG,NY,XEND,NY2
206   parameter (N=64,XG=2048)
207
208   double complex VEL(N,XG),TVEL(N,XG),TTVEL(XG),VOR(N,XG)
209   double complex UPTS(N,XG),VPTS(N,XG),VORPTS(N,XG),PPTS(N,XG
         )
210   double precision US(N,XG),VS(N,XG),VORS(N,XG)
211   double complex UTCOLL(N,XG),D2UTCOLL(N,XG)
212   double precision UCR,VCR,VORCR
213   double precision UCI,VCI,VORCI
```

```
214   double precision UCT,VCT,VORCT
215   double precision UCRP,VCRP,VORCRP
216   double precision UCIP,VCIP,VORCIP
217   double precision UCTP,VCTP,VORCTP
218   double precision PCR,PCI,PCT
219   double precision TT,DX,DT,R,LMAP
220   double complex IVEL(XG),DIVEL(XG),IVOR(XG),IVELS(XG),IVORS(
         XG)
221   double precision YPOS(N),PI
222   double precision VI,VR,WI,WR
223
224   integer XJ,YJ
225   logical SU,TP,ABORT
226
227   double precision XPOS,DU
228   double complex UCENT(XG)
229   double precision A,B
230   double precision XAMP(XG)
231   double precision XAMPC
232
233   integer TTI
234   character(LEN=14) fname
235
236         open (1,file='collinit.dat')
237         do XJ=1,XEND
238          read(1,*)
239          do YJ=1,NY
240           read(1,*) A,B,US(YJ,XJ),VS(YJ,XJ),VORS(YJ,XJ)
241          end do
242         end do
243         close (1)
244
245 C       calculate collocation values
246         call ucollptsfft(VEL,VOR,UPTS,DX,XEND,LMAP,NY)
247         call vcollptsfft(VEL,VPTS,XEND,NY)
248         call vcollptsfft(VOR,VORPTS,XEND,NY)
249
250         PI=4.d0*datan(1.D0)
251
252 131     format(A)
253 130     format(F6.1,'  ',E22.15,'  ',E22.15,'  ',E22.15,'  ',
      E22.15,'  ',E22.15,'  ',E22.15)
254
```

```
255          do  YJ=1,1
256           ypos(YJ)=cos((YJ-1)*PI/(2*NY))
257          end do
258
259 C        centre line values
260          TTI=int(TT)
261           fname(1:10)='centvt0000.dat'
262           if ((tt.ge.1).and.(tt.lt.10)) then
263             write(fname(10:10),'(I1)') TTI
264           else if ((tt.ge.10).and.(tt.lt.100)) then
265             write(fname(9:10),'(I2)') TTI
266           else if ((tt.ge.100).and.(tt.lt.1000)) then
267             write(fname(8:10),'(I3)') TTI
268           else if (tt.ge.1000) then
269             write(fname(7:10),'(I4)') TTI
270           end if
271             write(fname(11:14),'(A4)') '.dat'
272           open (1,file=fname)
273
274          if(TP) then
275           write(1,131) 'VARIABLES = "XPOS","|u_t|","|v_t|","|
                vor_t|","u_t","v_t","vor_t"'
276           write(1,*) 'ZONE I=',XEND
277          else
278           write(1,*) '#TIME',TT
279           write(1,131) '#VARIABLES = "XPOS","|u_t|","|v_t|","|
                vor_t|","u_t","v_t","vor_t"'
280           write(1,*) '#ZONE I=',XEND
281          end if
282
283          do XJ=1,XEND
284           XPOS=(XJ-1)*DX
285           do YJ=1,1
286            if (TTI.eq.0) then
287            UCT=0.d0
288            VCT=0.d0
289            VORCT=0.d0
290
291            UCTP=0.d0
292            VCTP=0.d0
293            VORCTP=0.d0
294
295            PCT=0.d0
```

```
296            else
297
298            UCT=dreal(UPTS(YJ,XJ))+dimag(UPTS(YJ,XJ))
299            VCT=dreal(VPTS(YJ,XJ))+dimag(VPTS(YJ,XJ))
300            VORCT=dreal(VORPTS(YJ,XJ))+dimag(VORPTS(YJ,XJ))
301
302            end if
303
304            write(1,130) real(XPOS),abs(dreal(UCT)),abs(dreal(
                  VCT)),abs(dreal(VORCT)),real(UCT),real(VCT),real
                  (VORCT)
305 C Set flag for programme abortion when values get too high
306             if (VORCT.gt.1.d0) ABORT=.true.
307           end do
308          end do
309           write(1,*)
310          close (1)
311    end subroutine
```

A.3.2. `startupparams` Module Code

```
1           subroutine startupparams(R,MFS,ALPH,DX,DT,XPTS,XP,
               TPTS,NITS,NY,LETA,XF,AMP,W,LXF,IMP,SU,PF,TP,NPROF
               ,LAMBDA)
2
3  C routines for reading in flow and discretization parameters
4
5       double precision R,MFS
6       double complex ALPH
7     integer XPTS,XP,TPTS,NITS,NY,N,XG
8       parameter (N=64,XG=2048)
9       double precision DX,DT,LETA
10      double precision XLEN,XPLATE,TTIME
11      integer XF,TECPL
12      double precision XFORCE,AMP,W,LXF
13 C    double precision DX,XFORCE,AMP,W,LXF
14      logical IMP,SU,PF,TP
15      integer FORCETYPE,STARTTYPE,FLOWTYPE
16      integer NPROF
17      double precision LAMBDA
18 C    double precision DISP
19
20        open (1,file='input.dat')
```

```
21      read(1,*) R
22  C     read in falk skan parameter
23        read(1,*) MFS
24      read(1,*) ALPH
25      read(1,*) DX
26      read(1,*) DT
27      read(1,*) XLEN
28      XPTS=int(XLEN/DX)+1
29  C     read in plate end location
30        read(1,*) XPLATE
31        XP=int(XPLATE/DX)+1
32      read(1,*) TTIME
33      TPTS=int(TTIME/DT)
34      read(1,*) NITS
35      read(1,*) NY
36      read(1,*) LETA
37  C         read in forcing location
38        read(1,*) XFORCE
39          XF=int(XFORCE/DX)+1
40  C         read in wall forcing amplitude scale
41          read(1,*) AMP
42  C         read in forcing frequency
43      read(1,*) W
44  C         read in forcing lengthscale parameter
45          read(1,*) LXF
46  C         read in forcing type
47         read(1,*) FORCETYPE
48         if (FORCETYPE.eq.1) then
49          IMP=.false.
50         else
51          IMP=.true.
52         end if
53  C         read in start-up type
54         read(1,*) STARTTYPE
55         if (STARTTYPE.eq.1) then
56          SU=.false.
57         else
58          SU=.true.
59         end if
60  C         read in whether or not parallel flow
61         read(1,*) FLOWTYPE
62         if (FLOWTYPE.eq.1) then
63          PF=.false.
```

```
64          else
65           PF=.true.
66          end if
67 C         read in whether Tecplot or Gnuplot compliant output
68          read(1,*) TECPL
69          if (TECPL.eq.1) then
70           TP=.true.
71          else
72           TP=.false.
73          end if
74           read(1,*) NPROF
75           read(1,*) LAMBDA
76 C         read(1,*) DISP
77           close (1)
78
79           open (2,file='outputparam.dat')
80           write(2,*) 'Reynolds number R'
81           write(2,*) R
82           write(2,*) 'fs param'
83           write(2,*) MFS
84           write(2,*) 'Outflow wavenumber'
85           write(2,*) ALPH
86           write(2,*) 'Streamwise stepsize'
87           write(2,*) DX
88           write(2,*) 'Time step'
89           write(2,*) DT
90           write(2,*) 'Streamwise total length'
91           write(2,*) XLEN
92 C         there is an extra point beyond the downstream
     boundary
93           if (XPTS.gt.XG) write(2,*) 'Too many streamwise
                points'
94           write(2,*) 'Plate location'
95           write(2,*) XPLATE
96           write(2,*) 'Total time interval'
97           write(2,*) TTIME
98           write(2,*) 'Number of iterations (streamwise march
                and coupling)'
99           write(2,*) NITS
100          write(2,*) 'Chebyshev order'
101          write(2,*) NY
102          if (NY.gt.N) write(2,*) 'Too many chebs'
103          write(2,*) 'Mapping lengthscale'
```

```
104           write(2,*) LETA
105           write(2,*) 'Forcing location'
106           write(2,*) XFORCE
107           write(2,*) 'Amplitude scale'
108           write(2,*) AMP
109           write(2,*) 'Forcing frequency W'
110           write(2,*) W
111           write(2,*) 'Streamwise forcing lengthscale'
112           write(2,*) LXF
113           write(2,*) 'Impulse or periodic forcing'
114           if (IMP) then
115            write(2,*) 'Impulsive excitation'
116           else
117            write(2,*) 'Periodic excitation'
118           end if
119           write(2,*) 'Null or given initial field?'
120           if (SU) then
121            write(2,*) 'Starting from given field'
122           else
123            write(2,*) 'Null start-up'
124           end if
125           write(2,*) 'Parallel flow?'
126           if (PF) then
127            write(2,*) 'Parallel flow'
128           else
129            write(2,*) 'Non-parallel flow'
130           end if
131           write(2,*) 'Tecplot?'
132           if (TP) then
133            write(2,*) 'Tecplot'
134           else
135            write(2,*) 'Gnuplot'
136           end if
137           write(2,*) 'Profile shape parameter N'
138           write(2,*) NPROF
139           write(2,*) 'Velocity ratio parameter'
140           write(2,*) LAMBDA
141           write(2,*) 'Characteristic lenght'
142 C         write(2,*) DISP
143           close (2)
144
145           end subroutine
```

A.4. POSTPROCESSING Programme Code

```
1  module SHAREDDATA
2
3  implicit none
4  save
5  character (len=*), parameter :: VERS='2.4'
6  integer :: NITS, NY, NX, XG, XPTS, TPTS, XF, FORCETYPE,
       STARTTYPE, XP, FLOWTYPE, NPROF, TECPL
7  double precision :: LMAP, R, W, DX, DT, XLEN, TTIME, XFORCE,
       AMP, XPLATE, LAMBDA, MFS, LETA, LXF, DISP
8  parameter (XG=2048)
9  double complex :: ALPH
10 logical :: IMPULSE, SU, PF
11
12 integer :: ADDMF, ORIGIN, RESCX, ETAZETASC, PLOT, RESCY,
       RESCUV
13 logical :: AMF, PS, RS, YY, YE, YZ, PL, RY, RU, XS, TP
14 double precision :: RDELTAASIN, XIN, XSCE, YSCE, USCE, VSCE,
       PSCE
15 double precision,parameter :: DELTAETA=1.21678
16
17 contains
18
19 !
20 !-------------------------------------
21 subroutine CALCSHAREDDATA
22
23
24
25   open (2,file='input.dat')
26   read(2,*) R
27   read(2,*) MFS
28   read(2,*) ALPH
29   read(2,*) DX
30   read(2,*) DT
31   read(2,*) XLEN
32   read(2,*) XPLATE
33   read(2,*) TTIME
34   read(2,*) NITS
35   read(2,*) NY
36   read(2,*) LETA
```

```
37   read(2,*) XFORCE
38   read(2,*) AMP
39   read(2,*) W
40   read(2,*) LXF
41   read(2,*) FORCETYPE
42   read(2,*) STARTTYPE
43   read(2,*) FLOWTYPE
44   read(2,*) TECPL
45   read(2,*) NPROF
46   read(2,*) LAMBDA
47   close(2)
48
49   XPTS=int(XLEN/DX)+1
50   NX=XPTS
51   TPTS=int(TTIME/DT)
52   XF=int(XFORCE/DX)+1
53   XP=int(XPLATE/DX)+1
54   DISP=1.d0
55   LMAP=LETA/DISP
56
57   RDELTAASIN = R
58   XIN = RDELTAASIN/(2*DELTAETA**2)
59
60
61   if (FORCETYPE.eq.1) then
62     IMPULSE=.false.
63   else
64     IMPULSE=.true.
65   end if
66
67   if (STARTTYPE.eq.1) then
68     SU=.false.
69   else
70     SU=.true.
71   end if
72
73   if (FLOWTYPE.eq.1) then
74     PF=.false.
75   else
76     PF=.true.
77   end if
78   TP=.true.
79
```

```
80 end subroutine CALCSHAREDDATA
81 !-----------------------------------
82 !
83
84 !
85 !-----------------------------------
86 subroutine READPPINPUT
87
88   open(1,file='ppinput.dat')
89   read(1,*) ADDMF
90   read(1,*) ORIGIN
91   read(1,*) RESCX
92   read(1,*) ETAZETASC
93   read(1,*) PLOT
94   read(1,*) RESCY
95   read(1,*) RESCUV
96   read(1,*) XSCE
97   read(1,*) YSCE
98   read(1,*) USCE
99   read(1,*) VSCE
100  read(1,*) PSCE
101  close(1)
102
103  if (ADDMF.eq.1) then
104    AMF=.true.
105  else
106    AMF=.false.
107  end if
108
109  if (ORIGIN.eq.1) then
110    PS=.true.
111    XS=.false.
112  else
113    PS=.false.
114    if (ORIGIN.eq.2) then
115      XS=.true.
116    else
117      XS=.false.
118    end if
119  end if
120
121  if (RESCX.eq.1) then
122    RS=.true.
```

```
123   else
124     RS=.false.
125   end if
126
127   if (ETAZETASC.eq.1) then
128     YY=.true.
129     YE=.false.
130     YZ=.false.
131   else
132     YY=.false.
133     if (ETAZETASC.eq.2) then
134       YE=.true.
135       YZ=.false.
136     else
137       if (ETAZETASC.eq.3) then
138         YE=.false.
139         YZ=.true.
140       else
141         YE=.false.
142         YZ=.false.
143       end if
144     end if
145   end if
146
147   if (PLOT.eq.1) then
148     PL=.true.
149   else
150     PL=.false.
151   end if
152
153   if (RESCY.eq.1) then
154     RY=.true.
155   else
156     RY=.false.
157   end if
158
159   if (RESCUV.eq.1) then
160     RU=.true.
161   else
162     RU=.false.
163   end if
164
165 end subroutine READPPINPUT
```

```
166 !-----------------------------------
167 !
168
169 end module SHAREDDATA
170 !-------------------------------------------------------
171 !
172
173 !
174 !-------------------------------------------------------
175 program POSTPROCESSING
176
177 use SHAREDDATA
178
179 implicit NONE
180
181 integer :: IY,IX,NSTOP
182 double precision :: XPOS,YPOS,U,V,P,XNEW,ZETA,Y,ETA,YNEW,UNEW
      ,VNEW,PNEW,RDELTAAS,BUFFER(11)
183 double precision,dimension(:),allocatable :: UCOLL,D2UCOLL
184
185 call CALCSHAREDDATA
186 call READPPINPUT
187
188 allocate(UCOLL(NY),D2UCOLL(NY))
189
190   open (1,file='sym-asym.dat')
191   1 format(F19.14,'    ',F18.14,'    ',F18.14,'    ',F16.14,'
            ',F16.14)
192   open (3,file='resc.dat',status='replace')
193
194 NSTOP = 50
195   if(TP) then
196     write(3,*)'VARIABLES = "X", "Y", "U", "V", "P"'
197     write(3,*) 'ZONE I= 50 J= 801'
198   else
199     if(AMF) then
200       write(3,*) '#Meanflow overlaid'
201     else
202       write(3,*) '#'
203     end if
204     if(PS) then
205       if(XS) then
206         write(3,*) '#x-origin shifted to LE before and to TE
```

```
          after scaling'
207       else
208         write(3,*) '#x-origin shifted to TE after scaling'
209       end if
210       write(3,*) '#x-origin shifted to plate end'
211     else
212       if(XS) then
213         write(3,*) '#x-origin shifted to LE before scaling'
214       else
215         write(3,*) '#'
216       end if
217       write(3,*) '#'
218     end if
219     if(RS) then
220       if(RY) then
221         if(RU) then
222           write(3,*) '#x and y and u (...) Scaled to Re'
223         else
224             write(3,*) '#x and y and - Scaled to Re'
225           end if
226       else
227         if(RU) then
228           write(3,*) '#x and - and u (...) Scaled to Re'
229         else
230           write(3,*) '#x and - and - Scaled to Re'
231         end if
232       end if
233     else
234       if(RY) then
235         if(RU) then
236             write(3,*) '#- and y and u (...) Scaled to Re'
237           else
238             write(3,*) '#- and y and - Scaled to Re'
239           end if
240         else
241           if(RU) then
242             write(3,*) '#- and - and u (...) Scaled to Re'
243           else
244             write(3,*) '#- and - and - Scaled to Re'
245           end if
246         end if
247     end if
248
```

```
249     write(3,*) '#Re', R
250     write(3,*) '#L', LMAP
251     write(3,*) '#XNEW, YNEW, UNEW, VNEW, PNEW'
252   end if
253
254   read(1,*)
255   read(1,*)
256
257   if(AMF) then
258     call readmf(UCOLL,D2UCOLL)
259   end if
260
261   if(RS.or.RY.or.RU) then
262     if(PF) then
263       RDELTAAS = RDELTAASIN
264     else
265 !     RDELTAAS = RDELTAASIN*DELTAAS/DELTAASIN
266     end if
267   end if
268
269   do IX=1,NX
270 !!!!!!!!
271 !   do IY=1,NY
272     do IY=1,NSTOP
273       read(1,*) XPOS,BUFFER(1),YPOS,U,V,BUFFER(2),BUFFER(3),
              BUFFER(4),BUFFER(5),BUFFER(6),BUFFER(7),BUFFER(8),
              BUFFER(9),BUFFER(10),BUFFER(11),P
274
275       if(AMF) then
276         U = U+UCOLL(IY)
277       end if
278
279       if(YY.or.YE.or.YZ) then
280         ZETA = 1.d0-YPOS
281       else
282         YNEW = YPOS
283       end if
284       if(YY) then
285         Y = LMAP/ZETA-LMAP
286         YNEW = Y
287       end if
288       if(YE) then
289         Y = LMAP/ZETA-LMAP
```

```
290         ETA = DELTAETA*Y
291         YNEW = ETA
292       end if
293       if(YZ) then
294         YNEW = ZETA
295       end if
296
297       XNEW = XPOS
298       if(XS) then
299         XNEW = XNEW+XIN
300       end if
301       if(PS) then
302         XNEW = XNEW-XPLATE
303       end if
304
305       if(RS) then
306         XNEW = XNEW*RDELTAAS**XSCE
307       end if
308       if(RY) then
309         YNEW = YNEW*RDELTAAS**YSCE
310       end if
311       if(RU) then
312         UNEW = U*RDELTAAS**USCE
313         VNEW = V*RDELTAAS**VSCE
314         PNEW = P*RDELTAAS**PSCE
315       else
316         UNEW = U
317         VNEW = V
318         PNEW = P
319       end if
320
321       write(3,1) XNEW,YNEW,UNEW,VNEW,PNEW
322
323     end do
324
325     do IY=(NSTOP+1),NY
326       read(1,*) XPOS,BUFFER(1),YPOS,U,V,BUFFER(2),BUFFER(3),
              BUFFER(4),BUFFER(5),BUFFER(6),BUFFER(7),BUFFER(8),
              BUFFER(9),BUFFER(10),BUFFER(11),P
327     end do
328
329     write(3,1)
330   end do
```

```
331
332   close(1)
333   close(3)
334
335   if(PL) then
336     call plotsub
337   end if
338
339   write(*,*) 'OK'
340
341 end program POSTPROCESSING
342 !-------------------------------------------------------
343 !
344
345 !
346 !-------------------------------------------------------
347 subroutine SUMUP
348
349 use SHAREDDATA
350
351 implicit none
352
353   open (4,file='sumup.dat',status='replace')
354
355   write(4,*) VERS, ' VERS POSTPROCESSING.EXE'
356
357   write(4,*) NITS, ' NITS'
358   write(4,*) NY, ' NY'
359   write(4,*) LMAP, ' LMAP'
360   write(4,*) R, ' R'
361   write(4,*) W, ' W'
362   write(4,*) DX, ' DX'
363   write(4,*) DT, ' DT'
364   write(4,*) XLEN, ' XLEN'
365   write(4,*) TTIME, ' TTIME'
366   write(4,*) ALPH, ' ALPH'
367   write(4,*) XFORCE, ' XFORCE'
368   write(4,*) FORCETYPE, ' FORCETYPE'
369   write(4,*) STARTTYPE, ' STARTTYPE'
370   write(4,*) AMP, ' AMP'
371   write(4,*) FLOWTYPE, ' FLOWTYPE'
372   write(4,*) XPLATE, ' XPLATE'
373
```

```
374   write(4,*) ADDMF, ' ADDMF'
375   write(4,*) ORIGIN, ' ORIGIN'
376   write(4,*) RESCX, ' RESCX'
377   write(4,*) ETAZETASC, ' ETAZETASC'
378   write(4,*) PLOT, ' PLOT'
379   write(4,*) RESCY, ' RESCY'
380   write(4,*) RESCUV, ' RESCUV'
381   write(4,*) XSCE, ' XSCE'
382   write(4,*) YSCE, ' YSCE'
383   write(4,*) USCE, ' USCE'
384   write(4,*) VSCE, ' VSCE'
385   write(4,*) PSCE, ' PSCE'
386
387   close(4)
388
389 end subroutine SUMUP
390 !-------------------------------------------------------
391 !
392
393 !
394 !-------------------------------------------------------
395 subroutine readmf(UCOLLs,D2UCOLLs)
396
397 use shareddata
398
399 implicit NONE
400
401 integer :: IYs
402 DOUBLE PRECISION :: IDATAs,UCOLLs(NY),D2UCOLLs(NY)
403
404   open (2,file='meanflow.dat')
405   do IYs=1,NY
406     read(2,*) IDATAs,UCOLLs(IYs),D2UCOLLs(IYs)
407   end do
408   close(2)
409
410 end subroutine readmf
411 !-------------------------------------------------------
412 !
413
414 !
415 !-------------------------------------------------------
416 subroutine plotsub
```

```
417
418 use shareddata
419
420 implicit NONE
421 !DOUBLE PRECISION :: RMANT
422 integer :: RI, XPLATEI, XFORCEI, AMPI, XLENI, REXP
423 CHARACTER (LEN=*), PARAMETER :: text0 = 'set title "',␣texta=
      ␣'Re=',␣textb=',␣x_plate=',␣textc=',␣x_force=',␣textd=',␣
      amplitude=',␣textz='"', texte=', x_length='
424
425   XPLATEI = INT(XPLATE)
426 ! XFORCEI = INT(XFORCE)
427 ! AMPI = INT(AMP)
428   XLENI = INT(XLEN)
429   RI = INT(R)
430 ! IF (R < 10000) THEN
431     1 format(A,A,I4,A,I3,A,I3,A)
432 ! ELSE
433 !   RMANT = R/1000
434 !   REXP = INT(R
435 !   1 FORMAT(A,I6,A,I3,A,I3,A)
436 ! END IF
437
438   open (1,file='title.gns',status='replace')
439   write(1,1) text0, texta, RI, textb, XPLATEI, texte, XLENI,
        textz
440   close(1)
441
442 end subroutine plotsub
443 !-------------------------------------------------------
444 !
```

A.5. GATHERING Programme Code

```
1 C
2 C-------------------------------------------------
3   module SHAREDDATA
4
5   implicit none
6   save
7   integer NITS,NY,NX,XG,XPTS,TPTS,XF,FORCETYPE,STARTTYPE,XP
8   integer TPER,NPROF,FLOWTYPE,TECPL
```

```
9   double precision LMAP,R,W,DX,DT,XLEN,TTIME,XFORCE,AMP,
       XPLATE,MFS,LETA
10  double precision LAMBDA,LXF
11  parameter (XG=2048)
12  double complex ALPH
13  logical IMPULSE, SU, PF
14
15  double precision PI
16
17
18
19  INTEGER ADDMF, ORIGIN, RESCX, ETAZETASC, PLOT, RESCY,
       RESCUV
20  LOGICAL AMF, PS, RS, YY, YE, YZ, PL, RY, RU, XS
21  DOUBLE PRECISION RDELTAASIN, XIN, XSCE, YSCE, USCE, VSCE,
       VORSCE
22  DOUBLE PRECISION DELTAETA
23  PARAMETER (DELTAETA=1.21678)
24
25  contains
26 C
27 C-----------------------------------
28  subroutine CALCSHAREDDATA
29
30  OPEN (2,file='input.dat')
31  READ(2,*) R
32  READ(2,*) MFS
33  READ(2,*) ALPH
34  READ(2,*) DX
35  READ(2,*) DT
36  READ(2,*) XLEN
37  READ(2,*) XPLATE
38  READ(2,*) TTIME
39  READ(2,*) NITS
40  READ(2,*) NY
41  READ(2,*) LETA
42  READ(2,*) XFORCE
43  READ(2,*) AMP
44  READ(2,*) W
45  READ(2,*) LXF
46  READ(2,*) FORCETYPE
47  READ(2,*) STARTTYPE
48  READ(2,*) FLOWTYPE
```

```
49    READ(2,*) TECPL
50    READ(2,*) NPROF
51    READ(2,*) LAMBDA
52    CLOSE(2)
53
54    PI=4.d0*datan(1.D0)
55    XPTS=INT(XLEN/DX)+1
56    NX=XPTS
57    TPTS=INT(TTIME/DT)
58    TPER=int((8.d0*datan(1.d0)/W)/DT)
59    XF=int(XFORCE/DX)+1
60    XP=int(XPLATE/DX)+1
61
62    RDELTAASIN = R
63    XIN = RDELTAASIN/(2*DELTAETA**2)
64
65
66    IF (FORCETYPE.eq.1) THEN
67      IMPULSE=.false.
68    ELSE
69      IMPULSE=.true.
70    END IF
71
72    IF (STARTTYPE.eq.1) THEN
73      SU=.false.
74    ELSE
75      SU=.true.
76    END IF
77
78    IF (FLOWTYPE.eq.1) THEN
79      PF=.false.
80    ELSE
81      PF=.true.
82    END IF
83
84    end subroutine CALCSHAREDDATA
85  C-----------------------------------
86  C
87
88    end module SHAREDDATA
89  C-------------------------------------------------
90  C
91
```

```
92 C
93 C-------------------------------------------------
94
95   program GATHERING
96
97   use SHAREDDATA
98
99   implicit none
100
101  real BUFFER(3)
102  double precision XPOS(XG), DUMP, TT,TEND
103  integer XJ,TS,TTI,TTIOLD,ERR,STN,TMAX
104  character(LEN=14) FNAME
105  logical INTELLIGENT,TRY
106
107  intelligent=.true.
108
109  call CALCSHAREDDATA
110
111  write(*,*) 'Time of final output (real):'
112  read(*,*) TEND
113  write(*,*) 'Please wait...'
114
115  if (TEND.gt.0) then
116    TMAX = int(TEND/DT)
117  else
118    TMAX = TPTS
119  end if
120
121 131 format(A)
122 130 format(F6.1,'  ',F6.1,'  ',E22.15,'  ',E22.15,'  ',E22
     .15)
123  open (2,file='txfield.dat')
124  write(2,131) '# VARIABLES = "TT","XPOS","|u_t|","|v_t|","|
       vor_t|"'
125  if (TPER.lt.8) then
126    STN = int(TEND)
127  else
128    STN = int(TEND/(TPER/8))
129  end if
130  write(2,*) '# ZONE J=',STN,' I=',XPTS
131
132  do TS=1,TMAX
```

```
133      TT=TS*DT
134      TTI=int(TT)
135 C    if (intelligent) then
136 C      if (TPER.lt.8) then
137 C        if (TTI.gt.TTIOLD) then
138 C          TRY=.true.
139 C        else
140 C          TRY=.false.
141 C        end if
142 C      else
143 C        if ((mod(TS,(TPER/8)).eq.0) .and. (TTI.gt.TTIOLD))
     then
144 C          TRY=.true.
145 C        else
146 C          TRY=.false.
147 C        end if
148 C      end if
149 C    else
150 C      TRY=.true.
151 C    end if
152      if (TTI.gt.TTIOLD) TRY=.true.
153      TTIOLD=TTI
154      if (TS.eq.TMAX) TRY=.true.
155      if (TRY) then
156        FNAME(1:10)='centvt0000.dat'
157        if ((TT.ge.1).and.(TT.lt.10)) then
158          write(FNAME(10:10),'(I1)') TTI
159        elseif ((TT.ge.10).and.(TT.lt.100)) then
160          write(FNAME(9:10),'(I2)') TTI
161        elseif ((TT.ge.100).and.(TT.lt.1000)) then
162          write(FNAME(8:10),'(I3)') TTI
163        elseif (TT.ge.1000) then
164          write(FNAME(7:10),'(I4)') TTI
165        end if
166        write(FNAME(11:14),'(A4)') '.dat'
167        open (1,file=FNAME,iostat=ERR)
168        if (ERR.eq.0) then
169 C        SUCC=SUCC+1
170          read(1,*)
171          read(1,*)
172          write(2,*)
173          do XJ=1,XPTS
174            read(1,*) XPOS(XJ),BUFFER(1),BUFFER(2),BUFFER(3)
```

```
175           write(2,130) TT,XPOS(XJ),BUFFER(1),BUFFER(2),BUFFER
                 (3)
176         end do
177       end if
178       close (1)
179     end if
180   end do
181
182   close (2)
183 C open (2,file='txfield.dat',position='rewind')
184 C write(2,131) 'VARIABLES = "TT","XPOS","u_t","v_t","vor_t"'
185 C write(2,*) 'ZONE I=',SUCC,' J=',XPTS
186 C close (2)
187   write(*,*) 'Done.'
188
189   end program GATHERING
190 C-------------------------------------------------
191 C
```

A.6. Unix Shell Script for LINSTAB Runs and Processing Associated Data

A.6.1. batchlin.sh Script to be Performed on a1.hww.de

```
1 for alpha in 0.001 0.010 0.020 0.030 0.040 0.050 0.060 0.070
     0.080 0.090 0.100 0.110 0.120 0.130 0.140
2 do
3 rm baseflow_in_000.setup
4 echo "# SETUPFILE FOR LINSTAB" >> baseflow_in_000.setup
5 echo "# ----------------------" >> baseflow_in_000.setup
6 echo "" >> baseflow_in_000.setup
7 echo "# GENERAL SPECIFICATIONS" >> baseflow_in_000.setup
8 echo "<version>           1.0" >> baseflow_in_000.setup
9 echo "<program type>       linstab" >> baseflow_in_000.setup
10 echo "" >> baseflow_in_000.setup
11 echo "# PATH SPECIFICATIONS" >> baseflow_in_000.setup
12 echo "<code directory>     /nfs/home6/HLRS/iag/iagjanis/code/
     linstab/aktuell" >> baseflow_in_000.setup
13 echo "<baseflow directory> /nfs/home6/HLRS/iag/iagjanis/daten
     /0.75presflow/1000" >> baseflow_in_000.setup
14 echo "<output directory>   /nfs/home6/HLRS/iag/iagjanis/daten
     /0.75presflow/1000/${alpha}" >> baseflow_in_000.setup
```

```
15 echo "" >> baseflow_in_000.setup
16 echo "# JOB SPECIFICATIONS" >> baseflow_in_000.setup
17 echo "<jobname>           baseflow_in_000" >>
     baseflow_in_000.setup
18 echo "<queue>                dq" >> baseflow_in_000.setup
19 echo "<cputime>           00:01:00" >> baseflow_in_000.setup
20 echo "<memory>            1gb" >> baseflow_in_000.setup
21 echo "<account>           iag02307" >> baseflow_in_000.setup
22 echo "<email>             iagjanis@iag.uni-stuttgart.de" >>
     baseflow_in_000.setup
23 echo "" >> baseflow_in_000.setup
24 echo "# INPUT PARAMETRS" >> baseflow_in_000.setup
25 echo "<y resolution>      301" >> baseflow_in_000.setup
26 echo "<dydeta>            1.0" >> baseflow_in_000.setup
27 echo "<flow pattern>      0" >> baseflow_in_000.setup
28 echo "<x index>           221" >> baseflow_in_000.setup
29 echo "<alpha_r>           ${alpha}" >> baseflow_in_000.setup
30 echo "<alpha_i>           0.0" >> baseflow_in_000.setup
31 echo "<gamma_r>           0.0" >> baseflow_in_000.setup
32 echo "<gamma_i>           0.0" >> baseflow_in_000.setup
33 mkdir ~/daten/0.75presflow/1000/${alpha}
34 linstab baseflow_in_000.setup
35 done
```

A.6.2. kc.sh Script to be Performed on a1.hww.de

```
1 for alpha in 0.001 0.010 0.020 0.030 0.040 0.050 0.060 0.070
    0.080 0.090 0.100 0.110 0.120 0.130 0.140
2 do
3 mv ~/daten/0.75presflow/1000/${alpha}/baseflow_in_000/*.eas
    ~/daten/0.75presflow/1000/${alpha}/
4 rm -R ~/daten/0.75presflow/1000/${alpha}/baseflow_in_000/*
5 rmdir ~/daten/0.75presflow/1000/${alpha}/baseflow_in_000
6 eas2tec ~/daten/0.75presflow/1000/${alpha}/*.eas
7 rm ~/daten/0.75presflow/1000/${alpha}/spektrum.eas
8 done
```

A.6.3. iagcp.sh Script to be Performed on a1.hww.de

```
1 for alpha in 0.001 0.010 0.020 0.030 0.040 0.050 0.060 0.070
    0.080 0.090 0.100 0.110 0.120 0.130 0.140
2 do
3 scp ~/daten/0.75presflow/1000/${alpha}/spektrum.plt cipserv.
```

```
    iag.uni-stuttgart.de:~/daten/0.75presflow/1000/${alpha}/
4 done
```

A.6.4. catchiag.sh Script to be Performed on cipserv.iag.uni-stuttgart

```
1 for alpha in 0.001 0.010 0.020 0.030 0.040 0.050 0.060 0.070
    0.080 0.090 0.100 0.110 0.120 0.130 0.140
2 do
3 mkdir ~/daten/0.75presflow/1000/${alpha}
4 done
```