TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Linear stability of the flow in a rectangular cavity driven by oblique lid motion

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Physikalische Energie- und Messtechnik

eingereicht von

Dipl.-Ing. Johannes Gugler
Matrikelnummer 01026727

ausgeführt am Institut für Strömungsmechanik und Wärmeübertragung
der Fakultät für Maschinenwesen und Betriebswissenschaften der Technischen Universität Wien

Betreuung
Betreuer/in: Univ.Prof. Dipl.-Phys. Dr.rer.nat. Hendrik Christoph Kuhlmann
Mitwirkung: Univ.Ass. MSc Pierre-Emmanuel Des Boscs

Wien, 12.12.2018

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Declaration of Authorship

I, Johannes Gugler, declare that this thesis titled, "Linear stability of the flow in a rectangular cavity under oblique lid motion" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

"*When I meet God, I am going to ask him two questions: Why relativity ? And why turbulence ? I really believe he will have an answer for the first.*"

Werner Heisenberg

TU WIEN

Abstract

Vienna University of Technology
TU Wien

Master of Science

Linear stability of the flow in a rectangular cavity under oblique lid motion

by Johannes GUGLER

This thesis is about a linear stability analysis for the lid driven cavity problem. The system of choice consists of a two dimensional rectangular box in $x-$ and $y-$direction, extended to infinity in the 3rd dimension ($z-$axis). The top lid of the box is moving tangentially to itself with a constant velocity and an inclination angle α with respect to the $x-$axis in the $z-$direction. In this thesis a Python program using the FEniCS library is written to simulate the flow and perform a stability analysis. The lid driven cavity is a benchmark system due to the simple rectangular geometry and therefore much theoretical work has already been done, which allows to test the written code and assure the correctness of the results. A linear stability analysis is carried out and the critical Reynolds numbers are determined as functoins of the cross-sectional aspect ration and the direction of lid motion. The energy budget of critical modes is analyzed using the Reynolds-Orr equation. For some parameter combinations, new modes are found at lower Reynolds numbers than already published results. The correctness of the present results is verified by full 3-dimensional flow simulations.

Acknowledgements

I acknowledge the understanding of my girlfriend Stephanie, who accepted the many hours I had to spend on this thesis while often neglecting the work that had to be done in the household. I also want to thank my family for paving my way to higher education with endless support.
From the specialists point of view, I want to appreciate the help and patience of Prof. Hendrik Kuhlmann and Pierre-Emmanuel des Boscs, who introduced me to the FEniCS code and helped me by simulating 3-dimensional flows.

Contents

Chapter 1

Introduction to the problem

This work is concerned with a stability analysis of a famous and simple system in fluid dynamics, the lid-driven cavity. The first chapter is meant as a small general introduction to the topic, while the theoretical framework is postponed to chapter 2. To stress the importance and applications of stability and turbulence in fluids, an introduction to the topic is given in section 1.1. The lid-driven cavity as the test system of this study is presented in section 1.2.

1.1 The importance of hydrodynamic stability

If a fluid is flowing at a low speed, quantified by the dimensionless Reynolds number, the behaviour of its flow is determined, once the boundary conditions are given. Increasing the velocity of the fluid may result in multiple solutions of the problem, which tend to preserve the symmetries of the underlying equations and boundary conditions. However, the existence of mulitple solutions may lead to a spontaneous symmetry breaking, which results in a bifurcation to other solutions. In order to be observed in experiment, the mathematical solution of the problem has to be stable, as it would decay otherwise. If the conditions of a basic flow are altered by the application of a disturbance, this disturbance may either decrease or increase in time. If the latter is the case, the underlying basic flow is said to be unstable. Finding the critical point, where the symmetry changes, is the goal of this thesis. An example for such a breaking of symmetry could be the transition from a two-dimensional to a three-dimensional flow. We will find these critical points, by performing a linear stability analysis, which will be introduced in chapter 2.
Instabilities also pave the way towards turbulences, which are a nuisance from a mathematical point of view because of the chaotic nature of the flow, which forbids to accurately predict the future state thereof, if the initial conditions are not given to infinite precision. The transition towards turbulence is dependent on the particular flow and may be accompanied by a sequence of bifurcations

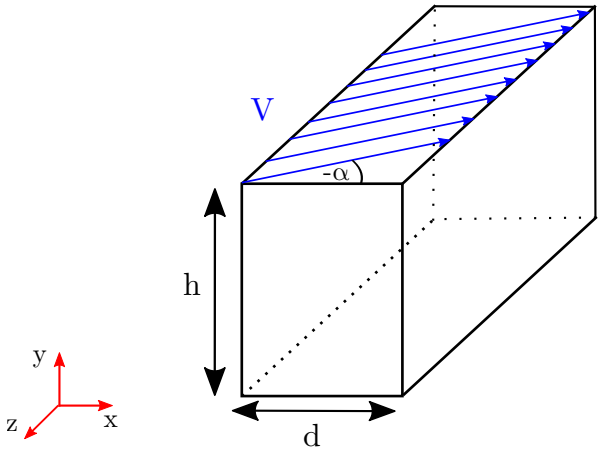


FIGURE 1.1: The lid driven cavity. The origin of the coordinate system will be chosen to be the center of the rectangular cross section in the x-y plane.

(for example in a Taylor-Couette or Benard system) or by a complete breaking of all symmetries (for example in a pipe flow). From the point of view of applications turbulences are unwanted when fluids are transported because they are accompanied by the evolution of interacting eddies, which causes drag and thus needs more energy for transport. However, there are cases, where turbulence is helpful in applications, e.g. when fluids are to be mixed. The transition towards turbulence is of utmost importance in understanding the physics of the atmosphere, in aircraft and in industrial applications. The importance of turbulence is reflected by Richard Feynman saying that "*Turbulence is the most important unsolved problem of classical physics*"[1].

1.2 The lid-driven cavity problem

In this work, we will be concerned with a very simple geometry and study the onset of instability in an infinitely extended (in z-direction) rectangular container with width d and height h and an aspect ratio $\Gamma = h/d$, where the top lid is moving at a given velocity V and a drive angle α, that measures the inclination in the x-z-plane. The cavity with the moving lid is depicted in Figure 1.1. This kind of flow was heavily studied in literature for the case of $\alpha = 0$, but for $\alpha \neq 0$ only the standard cavity with $\Gamma = 1$ was considered up to now. This thesis is about the expansion of the parameter space, as we will

[1] Feynman R. 1964.

vary the angles and the aspect ratios to determine the stability boundary of the basic flow. We will write a program to calculate a basic flow and perform a linear stability analysis by applying a small perturbation to the basic flow. To verify the results, we will be concerned with a comparison to the already published work in chapter 3, where the basic flows are calculated and their properties discussed. Before we deal with the technical implementation of the program, we will derive the mathematical foundation, necessary to tackle the stability problem of the lid driven cavity in chapter 2, where we will see that the determining parameters of the flow are the aspect ratio Γ, the drive angle α and the Reynolds number Re, which is a dimensionless parameter, given by $Re = Vd/\nu$, where ν is the kinematic viscosity of the fluid inside the cavity.

Chapter 2

Mathematical formulation of the problem

This chapter is concerned with the theoretical foundations of the descriptions of fluids, which will be used to model the turbulence in the lid-driven cavity problem. First, a short motivation of the Navier-Stokes equations will be given in section 2.1. The following section 2.2 introduces, how the equations are rewritten in dimensionless notation to extract the physically most relevant figures that describe the fluid. The apparatus of linear stability analysis is presented in section 2.3. Section 2.4 provides a structural quantification of critical modes, based on an energy-analysis, where the Reynolds-Orr equation is derived. The last section 2.5 deals with the numerical implementation of the dimensionless Navier-Stokes equations as implemented in the simulation programs used in this work.

2.1 Derivation of the Navier–Stokes equation [1]

To derive the Navier–Stokes equations, which are essentially a reformulation of Newtons second law of motion, a preliminary necessity is to clarify the frame of reference, i.e. the way we observe the flow. There exist two different specifications for describing the behaviour of a flow:

- In the *Eulerian* specification the flow field is in the center of the description. Each quantity b is represented on a fixed point in space $\vec{x}$.
- In the *Lagrangian* specification the individual particles are considered. They are labelled according to their position $\vec{X}$ at some fixed time t_0, which is usually chosen to be $t_0 = 0$.

[1] This derivation follows the introduction to fluid dynamics of the scriptum "Strömungslehre für TPh" by Prof. Braun (Braun (2001))

These specifications come with two types of time derivatives

$$\frac{\partial b}{\partial t} = \frac{\partial b(\vec{x}, t)}{\partial t} \tag{2.1a}$$

$$\frac{Db}{Dt} = \frac{\partial b(\vec{X}, t)}{\partial t}, \tag{2.1b}$$

where the first one is in the Eulerian specification and therefore accounts for the change of the quantity b at a certain position $\vec{x}$ in space, whereas the second derivative describes the change of the quantity while following the particle which started at t_0 at $\vec{X}$ and is therefore called the material derivative. The vector field $\vec{v}(\vec{x}, t)$ allows to obtain the position of a fluid particle by solving the equation

$$\frac{D\vec{x}}{Dt} = \vec{v}\left(\vec{x}, t\right) = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad . \tag{2.2}$$

Following the particle, gives a general relation between the time derivatives in the Lagrangian and Eulerian specifications:

$$\begin{aligned} \frac{Db}{Dt} &= \lim_{\Delta t \to 0} \frac{b\left(\vec{x} + \Delta\vec{x}, t + \Delta t\right) - b\left(\vec{x}, t\right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{b\left(\vec{x} + \vec{v}\Delta t, t + \Delta t\right) - b\left(\vec{x}, t\right)}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{b\left(\vec{x}, t\right) + \vec{\nabla} b\left(\vec{x}, t\right) \vec{v}\Delta t + \frac{\partial b(\vec{x}, t)}{\partial t}\Delta t - b\left(\vec{x}, t\right)}{\Delta t} \\ &= \left(\vec{v} \cdot \vec{\nabla}\right) b\left(\vec{x}, t\right) + \frac{\partial b\left(\vec{x}, t\right)}{\partial t} \quad . \end{aligned} \tag{2.3}$$

If a calculation of the change of an integral quantity, such as mass, momentum or energy, is required, *Reynold's* transport theorem, whose derivation is given in Appendix A is used:

$$\frac{D}{Dt} \int\limits_V b \, dV = \int\limits_V \left(\frac{\partial b}{\partial t} + \vec{\nabla} \cdot (b\vec{v}) \right) dV = \int\limits_V \frac{\partial b}{\partial t} dV + \oint b(\vec{v} \cdot \vec{n}) dO \quad . \tag{2.4}$$

The conservation of mass leads to the continuity equation

$$0 = \frac{D}{Dt} \int\limits_V \rho \, dV = \int\limits_V \left(\frac{\partial \rho}{\partial t} + \vec{\nabla} \underbrace{(\rho\vec{v})}_{\vec{j}} \right) dV \tag{2.5}$$

and since this holds for any arbitrary volume, the differential form also holds

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \vec{j} = 0\,, \tag{2.6}$$

where ρ is the density and $\vec{j}$ the mass flux density. Newton's second law reads

$$\frac{D}{Dt} \int\limits_V \rho \vec{v}\, dV = \oint\limits_{\partial V} \overleftrightarrow{\sigma} \vec{n}\, dO + \int\limits_V \rho \vec{g}\, dV, \tag{2.7}$$

with $\overleftrightarrow{\sigma}$ being the stress tensor, accounting for forces on the surface and $\rho\vec{g}$ accounting for volume forces (e.g. gravity). Using Gauss' divergence theorem and equation (2.5) this may be rewritten as

$$\int\limits_V \rho \frac{D\vec{v}}{Dt}\, dV = \int\limits_V \left(\vec{\nabla} \overleftrightarrow{\sigma} + \rho \vec{g} \right)\, dV \tag{2.8}$$

and since the volume is arbitrary again Newton's second law for infinitesimal quantities, expressed in the index notation (and assuming the Einstein notation for sums), gives

$$\partial_t v_i + v_j \partial_j v_i = \frac{1}{\rho} \partial_j \sigma_{ij} + g_i \quad . \tag{2.9}$$

The last ingredient to obtain a closed set of equations for the velocity $\vec{v}$ is a closed expression for $\overleftrightarrow{\sigma}$:
The easiest approach is to neglect the *viscous* terms ($\hat{=}$ off-diagonal terms) in $\overleftrightarrow{\sigma}$, which gives the stress-tensor of a *perfect fluid*, with the pressure p in the diagonal :

$$\sigma_{ij} = -p\ \delta_{ij} \quad . \tag{2.10}$$

If a linear dependence of the stress-tensor on the strain-rate tensor $D_{ij} \equiv 1/2(\partial_i v_j + \partial_j v_i)$ is assumed, one obtains the stress-tensor of a *Newtonian fluid*

$$\sigma_{ij} = (-p + \bar{\mu} \frac{\partial}{\partial x_k} v_k)\ \delta_{ij} + 2\mu D_{ij}, \tag{2.11}$$

where $\bar{\mu}$ and μ are two material parameters, known as *volume viscosity* and *shear viscosity*, respectively. This expression is used in equation (2.9) to obtain the *Navier-Stokes* equations:

$$\rho \frac{Dv_i}{Dt} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_i}(\bar{\mu} \frac{\partial}{\partial x_k} v_k) + \frac{\partial}{\partial x_j}(2\mu D_{ij}) + \rho g_i. \tag{2.12}$$

If an incompressible fluid ($\partial_i v_i = 0$) with constant material parameters ρ, μ is assumed[2] , the *Navier-Stokes* equations simplify drastically, giving

$$\frac{Dv_i}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 v_i}{\partial x_j \partial x_j} + g_i \,, \tag{2.13}$$

where the *kinematic viscosity* $\nu = \mu/\rho$ is introduced. This form of the *Navier-Stokes* equations will be used in this work to model the instability in the lid-driven cavity problem and in addition the gravitational force g_i will be neglected.

2.2 Dimensionless formulation of the Navier-Stokes equations

The incompressible *Navier-Stokes* equations in (2.13) contain many quantities, which come with dimensions. Getting rid of the dimensions by a suitable choice of units allows to extract the physical relevant figures.
The path towards a dimensionless equation is paved by introducing the dimensionless quantities

$$x = \tilde{x}/\tilde{x}_r \qquad t = \tilde{t}/\tilde{t}_r \qquad v = \tilde{v}/\tilde{v}_r \tag{2.14}$$

$$\rho = \tilde{\rho}/\tilde{\rho}_r \qquad p = \tilde{p}/\tilde{p}_r \qquad \nu = \tilde{\nu}/\tilde{\nu}_r \quad . \tag{2.15}$$

Here all the quantities with a tilde have dimensions and the quantities in the enumerator are the ones that appear in equation 2.13, whereas the denominators are reference values that define a certain scale. The purpose of this procedure is to replace the dimensionful quantities by their dimensionless relatives and to identify the remaining prefactors as the physical relevant parameters, allowing to reduce the parameter space drastically:

$$\frac{\partial \tilde{v}_i}{\partial \tilde{t}} + \tilde{v}_j \frac{\partial \tilde{v}_i}{\partial \tilde{x}_j} = -\frac{1}{\tilde{\rho}}\frac{\partial \tilde{p}}{\partial \tilde{x}_i} + \tilde{\nu}\frac{\partial^2 \tilde{v}_i}{\partial \tilde{x}_j \partial \tilde{x}_j} \tag{2.16}$$

$$\Rightarrow \quad \frac{\tilde{v}_r}{\tilde{t}_r}\frac{\partial v_i}{\partial t} + \frac{\tilde{v}_r^2}{\tilde{x}_r} v_j \frac{\partial v_i}{\partial x_j} = -\frac{\tilde{p}_r}{\tilde{\rho}_r \tilde{x}_r}\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\tilde{\nu}_r \tilde{v}_r^2}{\tilde{x}_r^2}\nu \frac{\partial^2 v_i}{\partial x_j \partial x_j} \quad . \tag{2.17}$$

For the lid-driven cavity problem, two sets of units are common in literature:

[2] An incompressible fluid is defined as having no materials derivative of the volume, which is proportional to the divergence of $\vec{v}$ as derived in equation (A.6) in Appendix A

	$\tilde{x}_r$	$\tilde{v}_r$	$\tilde{t}_r$	$\tilde{\rho}_r$	$\tilde{p}_r$	$\tilde{\nu}_r$
a)	d	V	d/V	ρ	ρV^2	ν
b)	d	ν/d	h^2/ν	ρ	$\rho\nu^2/d^2$	ν

The two formulations use the material parameters of the fluid ρ and ν, which are constant in our formulation, resulting in $\rho = 1$ and $\nu = 1$. With these formulations the dimensionless Navier-Stokes equations and boundary conditions read:

$$\frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 v_i}{\partial x_j \partial x_j} \qquad V = 1 \qquad (2.18a)$$

$$\frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial^2 v_i}{\partial x_j \partial x_j} \qquad V = Re \qquad (2.18b)$$

where we introduced the crucial dimensionless paramater $Re = Vd/\nu$, the *Reynold's* number. Taking a look at the equations (2.18a) and (2.18b), one can conlude that flows with similar Re will behave the same, so a huge dimensional parameter space has been reduced to one single variable, which we will have to scan in order to obtain insight in the stability of the lid driven cavity problem. In the formulation of the lid driven cavity problem V denotes the magnitude of the moving lid, which may have nonzero components in x- and z-direction. The inclination angle is the second parameter of the given problem. The no-slip no-penetration boundary conditions are applied for the non-moving walls. The third parameter of the given problem is of geometrical origin and is the aspect ratio of the cavity, $\Gamma \equiv h/d$. Note that for unsteady problems, the initial state of the velocity field would have to be given for a solution of the problem. The three parameters will be of big concern in this work and all the various types of cavities may be simulated by looping over these parameters. Here, the formulation in equation (2.18b) will be used, in accordance with Albensoeder, Kuhlmann, and Rath (2001).

2.3 Linear Stability Analysis

When we are interested in the stability of a flow, we consider what happens to a solution of the Navier-Stokes equations $\boldsymbol{V}_0(\vec{x}, t) = (\vec{v}_0, p_0)$, if we apply a small perturbation $\tilde{\boldsymbol{V}}(\vec{x}, t) = (\vec{\tilde{v}}, \tilde{p})$. We will then analyze, how the resulting flow fields $\boldsymbol{V}_p(\vec{x}, t) = (\vec{v}_p, p_p) = \boldsymbol{V} + \tilde{\boldsymbol{V}}$ evolve in time. According to Lyapunov, there exist three possibilities that could arise in such a situation[3]:

[3]These definitions are taken from www.wikipedia.org

- The system is called Lyapunov stable, if, for every $\epsilon > 0$ there exists a $\delta > 0$ s.t. if $\| \tilde{\boldsymbol{V}}(0) - \boldsymbol{x}\| < \delta$, then for every $t \geq 0$ we have $\|\boldsymbol{V}_p(t) - \boldsymbol{x}\| < \epsilon$.
- The system is called asymptotically stable if it is Lyapunov stable and there exists $\delta > 0$ s.t. if $\|\tilde{\boldsymbol{V}}(0) - \boldsymbol{x}\| < \delta$, then $\lim_{t\to\infty} \|\boldsymbol{V}_p(t) - \boldsymbol{x}\| = 0$.
- The system is called exponentially stable if it is asymptotically stable and there exist $\alpha > 0, \beta > 0, \delta > 0$ s.t. if $\|\boldsymbol{V}_p(0) - \boldsymbol{V}\| < \delta$ then $\|\boldsymbol{V}_p(t) - \boldsymbol{V}\| \leq \alpha \|\boldsymbol{V}_p(t) - \boldsymbol{V}\| e^{-\beta t}$.

In our case, we will start from the stationary quasi two-dimensional state with $\partial v_{0i}/\partial t = \partial v_{0i}/\partial z = \partial p_0/\partial z = \partial p_0/\partial t = 0$ for a given Reynolds number, drive angle and aspect ratio. Thus, we solve the stationary equations

$$u_0\partial_x u_0 + v_0\partial_y u_0 = -\partial_x p_0 + (\partial_x\partial_x + \partial_y\partial_y)u_0 \tag{2.19}$$
$$u_0\partial_x v_0 + v_0\partial_y v_0 = -\partial_y p_0 + (\partial_x\partial_x + \partial_y\partial_y)v_0 \tag{2.20}$$
$$u_0\partial_x w_0 + v_0\partial_y w_0 = (\partial_x\partial_x + \partial_y\partial_y)w_0 \tag{2.21}$$
$$\partial_x u_0 + \partial_y v_0 = 0 \tag{2.22}$$

subject to the boundary conditions

$$u_0 = Re \cdot \cos(\alpha), \quad w_0 = Re \cdot \sin(\alpha) \quad \text{at } y = \Gamma/2 \tag{2.23}$$
$$u_0 = v_0 = w_0 = 0 \quad \text{at } x = \pm 1/2 \text{ or } y = -\Gamma/2 \quad , \tag{2.24}$$

where u_0, v_0, w_0 denote the x-,y- and z-component of the velocity vector $\vec{v}_0$. Once, equations (2.19) - (2.22) subject to the boundary conditions (2.23) and (2.24) have been solved, we may use the solution as a starting guess for the unsteady flow and insert $\boldsymbol{V}_p$ in the full Navier-Stokes equations, resulting in

$$\partial_t \tilde{v}_i + v_{0j}\partial_j \tilde{v}_i + \tilde{v}_j\partial_j v_{0i} + \tilde{v}_j\partial_j \tilde{v}_i = -\partial_i \tilde{p} + \partial_j\partial_j \tilde{v} \quad . \tag{2.25}$$

If we assume the perturbation to be small, we may neglect the term $\tilde{v}_j\partial_j\tilde{v}_i$, which is nonlinear in the perturbation and employ the normal mode Ansatz

$$\tilde{\boldsymbol{V}} = \sum_{k,\omega} \begin{pmatrix} \vec{\tilde{v}}(x,y) \\ \tilde{p}(x,y) \end{pmatrix} e^{i(kz-\omega t)} e^{-\sigma t} \quad k, \omega, \sigma \in \mathbb{R} \quad , \tag{2.26}$$

which is promising due to the homogeneity of the basic flow in z-direction. Depending on the sign of σ, we can deduce, whether a given mode is exponentially

stable or not. If we insert the resulting $\boldsymbol{V}_p$ in the Navier-Stokes equations, we arrive at the generalized eigenvalue problem:

$$\begin{aligned}
&i) \quad \tilde{u}\partial_x u_0 + \tilde{v}\partial_y u_0 + u_0\partial_x\tilde{u} + v_0\partial_y\tilde{u} + w_0 ik\tilde{u} + \partial_x\tilde{p} - (\partial_x\partial_x + \partial_y\partial_y)\tilde{u} \\
&\qquad = (\sigma + i\omega)\tilde{u} \\
&ii) \quad \tilde{u}\partial_x v_0 + \tilde{v}\partial_y v_0 + u_0\partial_x\tilde{v} + v_0\partial_y\tilde{v} + w_0 ik\tilde{v} + \partial_y\tilde{p} - (\partial_x\partial_x + \partial_y\partial_y)\tilde{v} \\
&\qquad = (\sigma + i\omega)\tilde{v} \\
&iii) \quad \tilde{u}\partial_x w_0 + \tilde{v}\partial_y w_0 + u_0\partial_x\tilde{w} + v_0\partial_y\tilde{w} + w_0 ik\tilde{w} + ik\tilde{p} - (\partial_x\partial_x + \partial_y\partial_y)\tilde{w} + k^2\tilde{w} \\
&\qquad = (\sigma + i\omega)\tilde{w} \\
&iv) \quad \partial_x\tilde{u} + \partial_y\tilde{v} + ik\tilde{w} = 0
\end{aligned} \tag{2.27}$$

subject to the boundary conditions:

$$\vec{\tilde{v}} = 0 \text{ at all boundaries} \quad . \tag{2.28}$$

Solving this generalized eigenvalue problem for different values of Γ, α, Re, k allows to investigate the stability for this parameter space. In particular important are the points, where a flow starts to evolve instabilities of the basic flow, which corresponds to negative decay rates σ. The Reynolds numbers for which $\sigma = 0$ are called *neutral* Reynolds numbers Re_n. The lowest neutral Reynolds number for a given Γ, α is called *critical* Reynolds number Re_c. This thesis will be mainly concerned with finding Re_n and Re_c for a given aspect ratio and drive angle.

2.4 Energy analysis – the Reynolds-Orr equation

When the critical modes are calculated, we can also take a look at where the energy that enhances the perturbation comes from. For this reason we consider the change of the mean kinetic energy per unit mass of the perturbation with time[4]

$$\frac{dE_{kin}}{dt} = \frac{d}{dt}\int\limits_V \frac{\vec{\tilde{v}}^2}{2} dV = \int\limits_V \vec{\tilde{v}} \cdot \frac{d\vec{\tilde{v}}}{dt} dV = \int\limits_V \vec{\tilde{v}} \cdot \frac{\partial\vec{\tilde{v}}}{\partial t} dV \quad , \tag{2.29}$$

where we assumed that the volume V is time independent and we made use of the fact that the nonlinear term with $\vec{\tilde{v}} \cdot \vec{\nabla}\vec{\tilde{v}}$ is energy conserving. This term may be rewritten by inserting the linearized Navier-Stokes equations (2.27) to

[4]This derivation of the Reynolds-Orr equation follows the treatment in Kuhlmann (2012).

give

$$\frac{dE_{kin}}{dt} = \int\limits_V \left(\underbrace{-\tilde{v}_i\tilde{v}_j\partial_j v_{0j}}_{I} \underbrace{-\tilde{v}_i v_{0j}\partial_j\tilde{v}_i}_{II} \underbrace{-\tilde{v}_i\tilde{v}_j\partial_j\tilde{v}_i}_{III} \underbrace{-\tilde{v}_i\partial_i p}_{IV} + \underbrace{\tilde{v}_i\partial_j\partial_j\tilde{v}_i}_{V} \right) dV \quad . \tag{2.30}$$

The terms II and III in this equation vanish because we get the same expressions with a different sign by an integration by parts, using the boundary conditions and the fact that we deal with an incompressible fluid:

$$\int\limits_V dV\tilde{v}_i v_{0j}\partial_j\tilde{v}_i = \int\limits_V dV\partial_j\left(\tilde{v}_i v_{0j}\tilde{v}_i\right) - \int\limits_V dV\tilde{v}_i(\partial_j\tilde{v}_i v_{0j}) \tag{2.31}$$

$$= \underbrace{\int\limits_{\partial V} dA_j\left(\tilde{v}_i v_{0j}\tilde{v}_i\right)}_{=0} - \int\limits_V dV\tilde{v}_i v_{0j}\partial_j\tilde{v}_i \tag{2.32}$$

and

$$\int\limits_V dV\tilde{v}_i\tilde{v}_j\partial_j\tilde{v}_i = \int\limits_V dV\partial_j\left(\tilde{v}_i\tilde{v}_j\tilde{v}_i\right) - \int\limits_V dV\tilde{v}_i(\partial_j\tilde{v}_i\tilde{v}_j) \tag{2.33}$$

$$= \underbrace{\int\limits_{\partial V} dA_j\left(\tilde{v}_i\tilde{v}_j\tilde{v}_i\right)}_{=0} - \int\limits_V dV\tilde{v}_i\tilde{v}_j\partial_j\tilde{v}_i \quad . \tag{2.34}$$

The term IV also vanishes due to incompressibility and the boundary conditions for $\tilde{v}$ and term V may be integrated by parts, such that we arrive at the *Reynolds-Orr* equation:

$$\frac{dE_{kin}}{dt} = \int\limits_V \left(-\tilde{v}_i\tilde{v}_j\partial_j v_{0i} - (\partial_j\tilde{v}_i)^2\right) dV \quad . \tag{2.35}$$

The second term in this equation is always negative and called the dissipation rate. The only term which may be responsible for a production of kinetic energy in the perturbation mode is the first term which is called the production rate even though it may also cause energy loss. To understand the mechanism of the local production of kinetic energy it is convenient to split the perturbation in contributions parallel and orthogonal to the basic flow

$$\vec{\tilde{v}}_{\parallel} = \frac{\left(\vec{\tilde{v}}\cdot\vec{v}_0\right)\vec{v}_0}{\vec{v}_0^2} \quad \text{and} \quad \vec{\tilde{v}}_{\perp} = \vec{\tilde{v}} - \vec{\tilde{v}}_{\parallel} \quad . \tag{2.36}$$

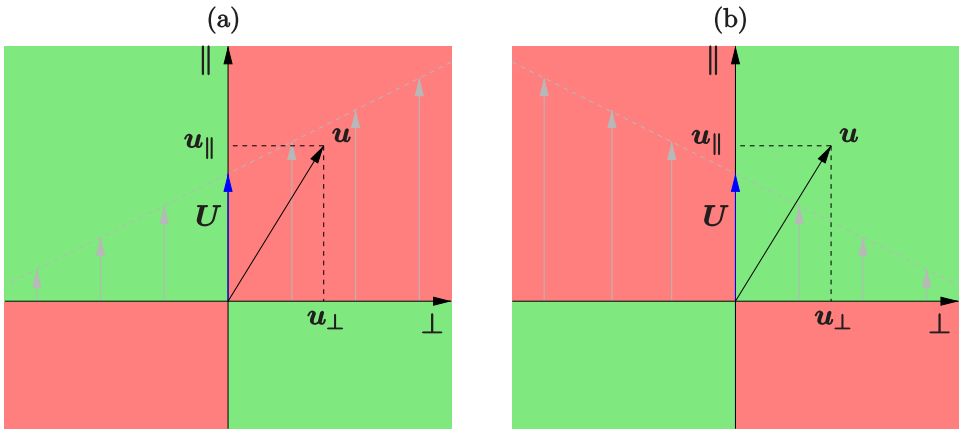


FIGURE 2.1: Energy production rate. a) the gradient of the basic flow is positive and thus the direction of the first and third quadrants are dissipating energy (red regions) and the second and fourth are producing.

This allows to interpret the sign of the production rate, as shown in Figure 2.1[5]. For a comparison with literature, we will decompose the change of the kinetic energy in so called normalized energy transfer terms

$$D = \frac{1}{D^*}\left[\vec{\nabla} \times (\vec{\tilde{v}}_\perp + \vec{\tilde{v}}_\parallel)\right]^2 \tag{2.37}$$

$$I_1 = -\frac{1}{D^*}\vec{\tilde{v}}_\perp \cdot (\vec{\tilde{v}}_\perp \cdot \vec{\nabla}\vec{v}_0) \tag{2.38}$$

$$I_2 = -\frac{1}{D^*}\vec{\tilde{v}}_\parallel \cdot (\vec{\tilde{v}}_\perp \cdot \vec{\nabla}\vec{v}_0) \tag{2.39}$$

$$I_3 = -\frac{1}{D^*}\vec{\tilde{v}}_\perp \cdot (\vec{\tilde{v}}_\parallel \cdot \vec{\nabla}\vec{v}_0) \tag{2.40}$$

$$I_4 = -\frac{1}{D^*}\vec{\tilde{v}}_\parallel \cdot (\vec{\tilde{v}}_\parallel \cdot \vec{\nabla}\vec{v}_0) \quad , \tag{2.41}$$

where all quantities are normalized by the volume integral of the dissipation rate $D^* = \int_V D dV$. This decomposition allows to understand the mechanism, which is responsible for the energy production by considering the gradients of the basic flow and the projections on the parallel and orthogonal components of the perturbation. Normalizing the change in kinetic energy, we obtain

$$\frac{1}{D^*}\frac{d\,E_{kin}}{dt} = -1 + \sum_{i=1}^{4}\int_V I_i dV \quad , \tag{2.42}$$

[5]This Figure is taken from Kuhlmann (2012)

which allows for the characterization of a neutral mode, for which the production and dissipation rate are of the same magnitude, s.t. $\sum_i \int_V I_i dV = 1$. In this thesis will use the notation I_i both for the local as well as the integrated value of each energy contribution. No confusion should arise because the actual meaning is usually deducible from the context. In chapter 4, we will analyze the perturbation modes from the point of view of the Reynolds–Orr equation.

2.5 Numerical implementation [6]

In the previous sections, the Navier–Stokes equations were derived and a dimensional analysis for the special case of an incompressible fluid was performed in order to obtain the physical relevant parameters of the problem. This section is concerned with the numerical solution of the problem, where general Galerkin methods are introduced and the specialization of two approaches taken in this work, the finite element and spectral element methods will briefly be explained.

2.5.1 Galerkin methods

In this section, we will derive an approach to obtain the solutions of the differential equation

$$\mathcal{L}\, f(\vec{x}, t) = 0\,, \tag{2.43}$$

where $\mathcal{L}$ denotes a differential operator. Such a differential equation may be solved numerically by expanding the unknown solution in a set of Ansatz functions [7]

$$\bar{f}(\vec{x}, t) = \sum_{n=1}^{N} a_n(t)\phi_n(\vec{x}) \quad . \tag{2.44}$$

In this equation the bar above the function f emphazises that the true solution will only be approximated by the finite set of Ansatz functions. The a_n are the expansion coefficients and the ϕ_n denote the Ansatz functions[8]. The insertion of the numerical Ansatz (2.44) in the differential equation (2.43) will not solve

[6] The treatment of this section follows the script "Grundlagen der numerischen Methoden der Strömungs- und Wärmetechnik" (Kuhlmann 2010)

[7] For completeness, the function space would in general have to be infinite $N \to \infty$, so completeness will only be fulfilled approximately and the convergence with respect to the number of test functions will always have to be tested.

[8] The choice of Ansatz function is crucial and should correspond to the given problem since a proper set of Ansatz functions reduces the number of necessary coefficients drastically, e.g. in quantum mechanical problems, the hydrogen orbitals are proper basis functions whereas in periodic systems plane waves are common Ansatz functions.

it exactly but leave some rest R, called the residue

$$\mathcal{L}\,\bar{f}(\vec{x},t) = R(\vec{x},t) \quad . \tag{2.45}$$

Minimization of the residue will give the best agreement between the approximate function $\bar{f}$ and the exact solution f. To determine the a_n, which give the minimal residue, a set of N equations is needed. This set of equations is obtained by weighing the residue with N different weighing functions W_m and demanding the weighted residues to be zero

$$\int_V W_m(\vec{x})R(\boldsymbol{x},t)\,dV = 0\,, \quad m = 1,\ldots,N \quad . \tag{2.46}$$

Depending on the choice of the weighing functions, we get different methods for the solution of the differential equation. If the Ansatz functions form an orthogonal basis, a practical choice for the weighing functions W_m are the Ansatz functions themselves

$$W_m(\vec{x}) = \phi_m(\vec{x}). \tag{2.47}$$

Methods with this choice of weighing functions are called *Galerkin methods* and will be used in this work.

The finite element method

In the finite element method the Ansatz functions are chosen to have only finite values at certain regions in space, therefore partitioning the problem in many small ones. In this method, space is partitioned in small divisions, called *elements*, which supply grid points, called *nodes*, where the value of the Ansatz function is to be evaluated and supposed to have the same value as the solution function $f(\vec{x},t)$, s.t. the problem resembles an interpolation of this function in the interstitial region between nodes. The Ansatz functions are chosen to have finite values only in the element, where the node is located and the neighbouring elements. This choice of Ansatz functions will yield block-tridiagonal matrices for the coefficients a_n in equation (2.44), when inserted in equation (2.46). The numerical advantage is that block-tridiagonal matrices may be solved very fast by exploiting the sparsity of the matrices. A one-dimensional example would consist of elements, which are represented by lines, and the number of necessary nodes is dependent on the degree of the interpolating Ansatz function, as may be seen in Figure 2.2. For linear interpolation, two nodes per element are

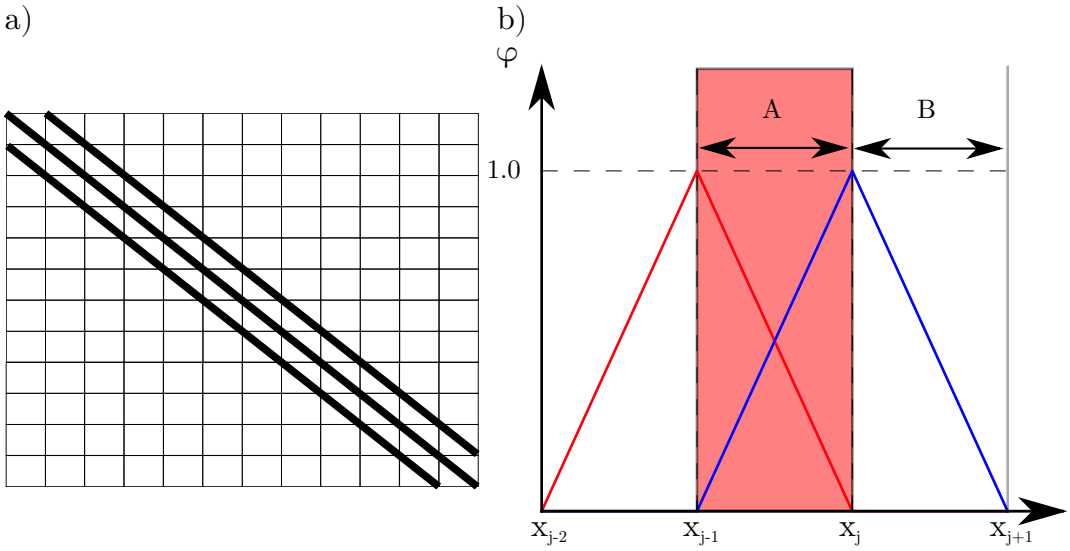


FIGURE 2.2: a) The structure of a block-tridiagonal matrix, which is easily diagonalized due to its structured sparsity. b)A sketch of two elements in 1d and two linear Ansatz functions ϕ_{j-1}(red line) and ϕ_j(blue line). The solution will be exactly represented at the nodes and the functional representation of the solution in element A is given by: $f(x) = a_{j-1}\phi_{j-1}(x) + a_j\phi_j(x)$.

sufficient to obtain an accuracy of $\mathcal{O}(\Delta x^2)$[9], quadratic interpolation needs three points with an improved accuracy of $\mathcal{O}(\Delta x^3)$. A physicists induction yields that the number of nodes for an interpolation polynomial of order N is $N+1$. The accuracy increases with the order of the interpolation polynomial, however if too many calculations per element have to be performed, it might be advantageous to use a finer mesh and a lower order of interpolation. If convergence is obtained with increasing order of interpolation polynomials the method is called p-FEM, if we converge by increasing the mesh size, we are talking about h-FEM. In this thesis, we will use low order polynomials and check convergence with respect to mesh densities.

For the finite element method we use the code FEniCS (written by Alnæs et al. (2015)), which is a Python/C hybrid, combining the syntax strength of Python with the speed of C, where meshing is already implemented. In the case of the lid-driven cavity, we fill the two-dimensional space with triangular elements. The meshing is already implemented in the program and since the boundaries of the cavity are important to be properly resolved, we use a mesh with a higher

[9] Δx denotes the grid spacing

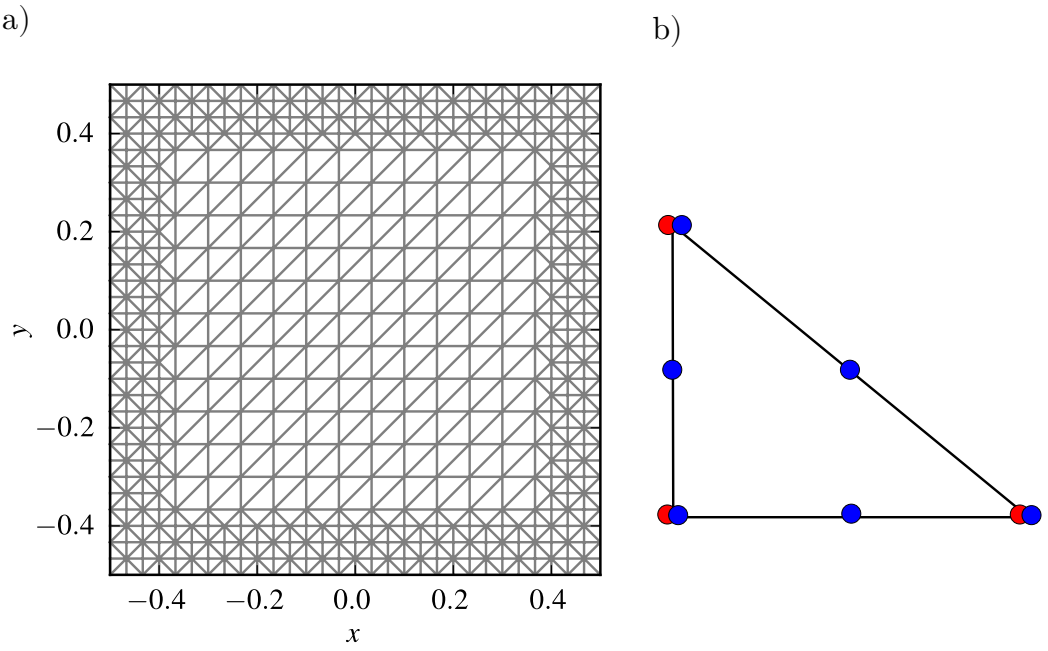


FIGURE 2.3: a) An example for the used meshes for the solution of the Navier-Stokes equations. For illustrational purposes the grid is chosen to be coarse, starting with 20x20 vertex points and after a refinement on the boundary we end up with 576 vertices. b) The Taylor-Hood element: The blue dots denote the nodes for the velocity field and the red dots the ones for pressure.

resolution at the boundaries, as may be seen in Figure 2.3a. Due to convergence reasons we define our functions on a mixed-element space, which is called the Taylor-Hood element. This element implements a quadratic interpolation for the velocity fields, whereas the pressure field is linearly interpolated (see Figure 2.3b). The finite element method will be used in this work for the linear stability study because of the flexibility of the FEniCS program but for comparison, we will also use a different method to crosscheck our results. The workhorse for crosschecking will be the spectral element method.

The spectral element method

The spectral element method, which we will use to check for convergence with respect to the basis functions, is a hybrid of a spectral method and the finite element method. In the original spectral element methods, the basis functions used in equation (2.44) are usually nonzero over the whole domain and more complex than the simple piecewise polynomials of the FEM, e.g. trigoniometric functions (that is, where the name stems from), Chebyshev polynomials or

high order polynomials. If we merge the local approach from the FEM and use these basis functions inside a given element, we end up with the spectral element method. In general, a more complicated basis function makes the method numerically more time consuming, as the number of evaluation points per element increases. Depending on the basis functions, integrals may be computed faster and more accurately by using special points and weights for their evaluation, similar to Gaussian quadrature. In our case, we can consider the use of the spectral elements as a convergence study with respect to basis functions. Again, we will make use of an already developed open-source code, in this case called NEK5000 (Paul F. Fischer and Kerkemeier (2008)). With these theoretical tools in our hand, we can start to analyze the lid-driven cavities.

Chapter 3

Determining the basic flow – convergence studies

Before we start to analyze the stability of a given lid-driven cavity, we have to carefully determine the two-dimensional basic flow of the problem. First, we will consider the drive angle $\alpha = 0$ in order to compare the flows, obtained with finite elements using FEniCS, with spectral elements using NEK5000 and with lid-driven cavity flows reported in literature. This step is very crucial, because a wrong basic flow will inevitably yield a wrong stability analysis. In this chapter, we will converge the basic flows with respect to mesh sizes, mesh shapes and basis functions. The convergence study will be performed for different Reynolds numbers and aspect ratios Γ. The plots for $\Gamma \neq 1$ are found in Appendix B.

3.1 $\Gamma = 1$ Basic flows

3.1.1 Basic flow computed with finite elements

Low Reynolds numbers

In order to converge the results more easily, we start our convergence study with low Reynolds numbers. For this purpose, we consider the case $Re = 10$. The first simulations are performed on a regular mesh and the Taylor-Hood element [1]. A calculation for a very fine grid with 200 grid points in x- and y-direction results in the basic flow shown in Figure 3.1. Since we will want to compare the resulting critical Reynolds numbers with published results, we consider the grid spacing used in the literature to obtain an estimate for the necessary grid density: Albensoeder, Kuhlmann, and Rath (2001) used a finite-volume formulation with a grid of 141x141 grid points, but to obtain a finer resolution at the boundaries, they compressed the last 35 cells in each direction

[1] this will be the case for all finite element calculations, unless mentioned otherwise

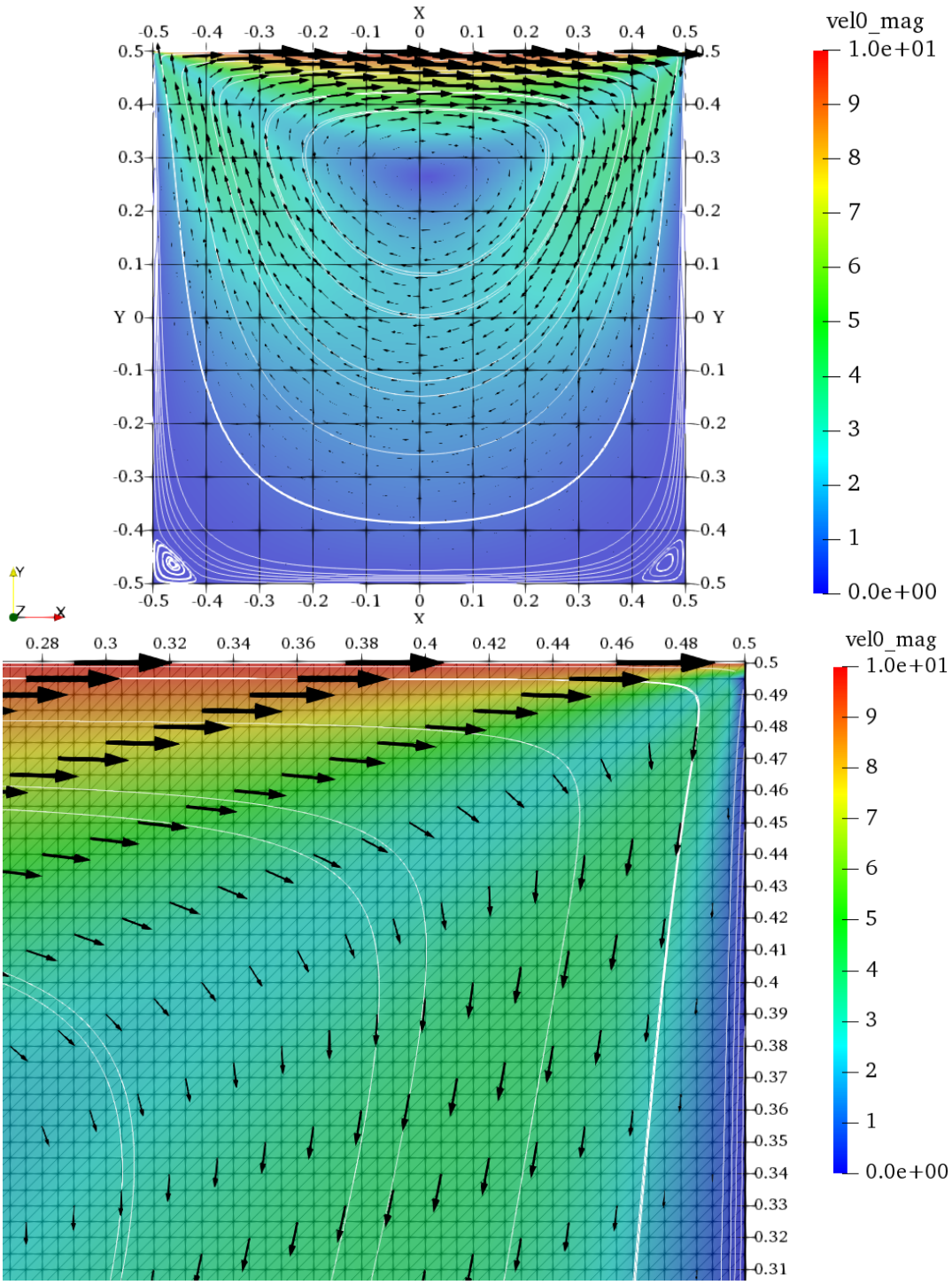


FIGURE 3.1: Basics flow for $Re = 10$: *top*, The stationary flow for a regular grid with $n_x = n_y = 200$ grid points in each direction. The arrows denote the velocity field and the colour the magnitude of the arrows. The white curves show the streamlines of the flow. *bottom*, A zoom in the right top edge of the cavity shows the structure of the underlying mesh.

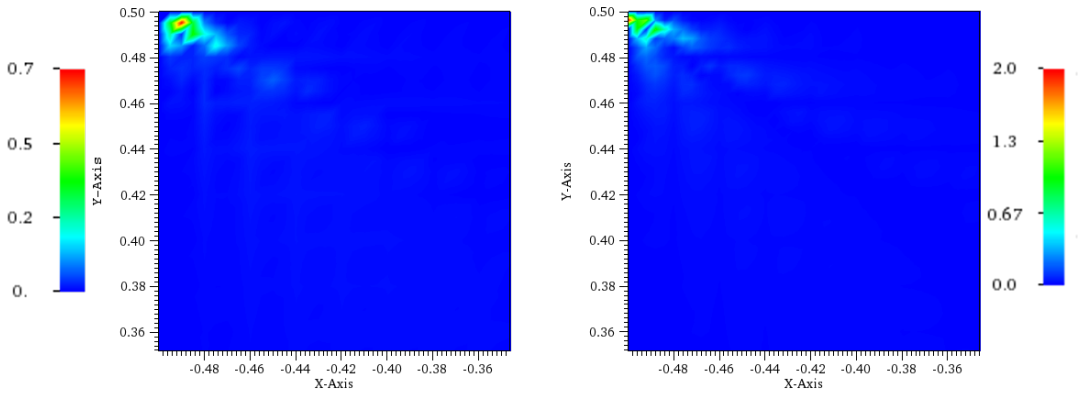


FIGURE 3.2: Grid comparison 200×200 versus 250×250 , $Re = 10$, the left plot shows the difference in the magnitude of the velocity fields, if the higher resolution grid is interpolated on the coarser grid and the right plot the vice versa interpolation. The lid is located at the top and moves to the right.

by a factor of 0.95 from cell to cell. The second reference (Theofilis, Duck, and Owen 2004) used a spectral method with 128 collocation points.

Our first approach is a regular grid without mesh refinement towards the boundaries. We compare the difference in the resulting magnitude of the velocities and pressures for the case of the highest tractable mesh resolution with 250x250 grid points[2]. For a graphical comparison, we first use the plotting program VisIt[3] to interpolate between the mesh points. For a comparison between the data on the two meshes, we can either interpolate the resulting magnitudes of the velocity field from the coarse grid to the fine one or vice versa. Comparing the two variants, we see that the only non-negligible difference of velocity magnitudes is present at the top edges of the cavity (see Figure 3.2). The argument here, however, is to be taken on a qualitative level, because the interpolation from one mesh to the other will give an additional source of error, due to the fact that VisIt does not know about the coefficients of the function in equation (2.44). In addition to this comparison, we calculate the velocities directly in FEniCS along three lines (the two diagonals and the vertical line in the center) using the calculated coefficients for the evaluation of the function values. The results for one diagonal are shown in Figure 3.3, whereas the other two lines are presented in the Appendix (Figure B.2 and Figure B.3). The convergence studies

[2]This thesis is mainly computed on a laptop, so the available RAM is low.
[3]Childs et al. 2012.

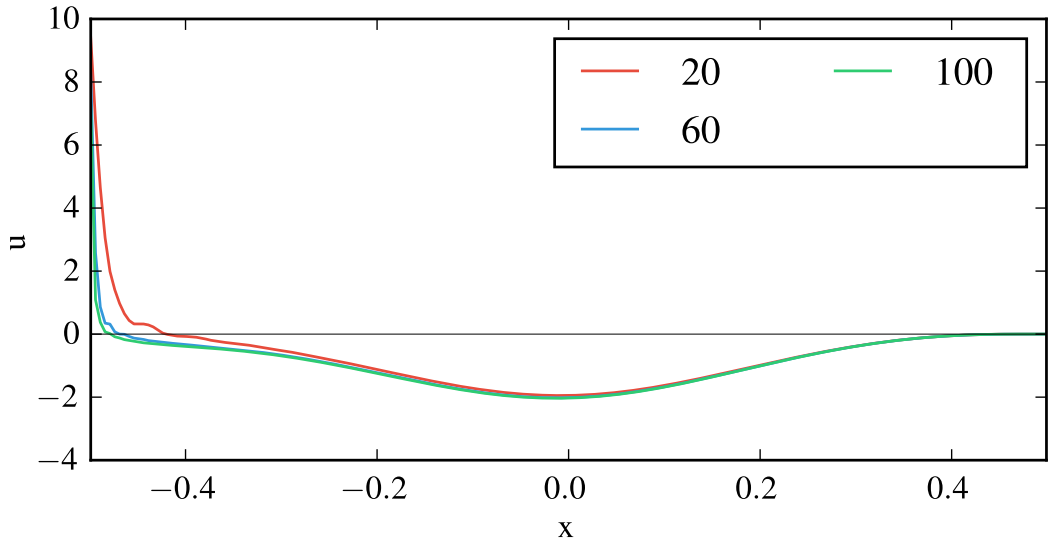


FIGURE 3.3: Grid convergence along the diagonal $y = -x + 0.5$ from the top left to the right bottom of the cavity. The lines show the interpolated data along the calculated points.

reveal that the edges are the most crucial regions. This is owed to the circumstance that we have a singularity in the boundary conditions at the positions $P_1 = (-0.5, \Gamma/2)$ and $P_2 = (0.5, \Gamma/2)$. This situation was analyzed by Hancock, Lewis, and Moffatt (1981) and Gupta, Manohar, and Noble (1981), where the streamlines were calculated. There is an interesting feature present in the bottom corners: As Moffatt (1964) has shown , the singular geometry results in progressively weaker counter rotating eddies, which means that the resolution towards the boundary is never able to resolve the correct behavior of the flow in this region. For the case of the stability analysis, we always have to consider this fact and check for the influence of the boundary region on the stability. The difference of the velocities decays very fast towards the center of the cavity, as may be seen in Figure B.1. The conclusion of this is that the mesh resolution is fine in the center of the cavity but needs to be refined towards the boundary. We do this by a refinement procedure, where we first double the mesh resolution for all elements that are less than 30% of the cavity extensions apart from the borders. The second refinement is done for each element in a vicinity of 15% and a last refinement for each point within the closest 5% of the border points. These refinements are done subsequently, such that the mesh density close to the boundary has increased by a factor of 8. The results of these calculations are visualized in Fig. 3.4 and we see that we can indeed improve convergence in the critical region. The simulations show that a mesh with 60×60 grid points might be sufficient to obtain proper results. The comparison with VisIt and

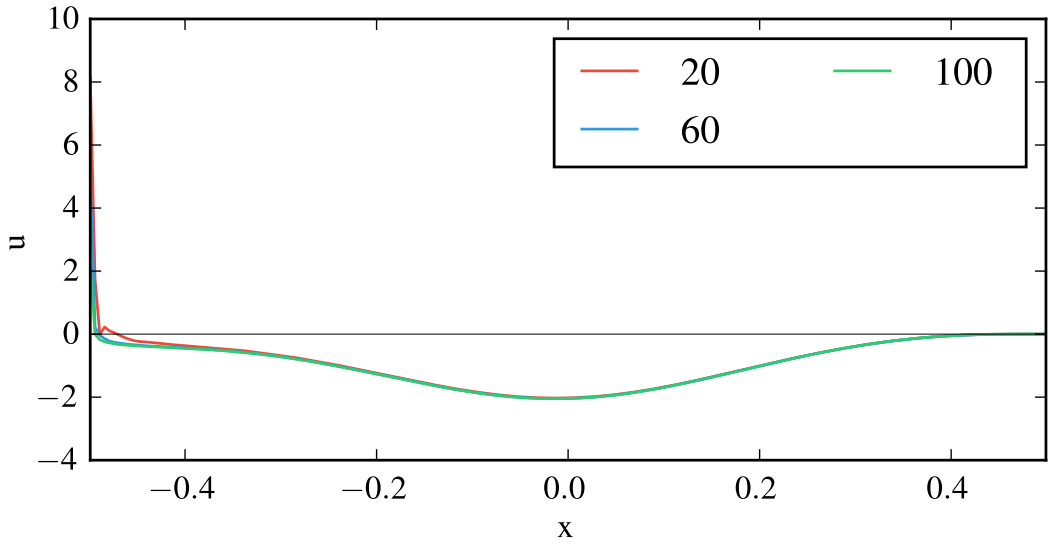


FIGURE 3.4: Grid convergence of the x-component of the velocity along the diagonal $y = -x + 0.5$ from the top left to the right bottom of the cavity.

the improvement with refinement over the equally spaced grids are shown in the Appendix (Fig. B.4,). Whether the mesh is sufficiently dense will be double checked in the actual calculations of critical Re - k combinations by a subsequent mesh refinement in the vicinity of critical values. Also we will repeat our convergence study for the more interesting cases of higher Re, where critical modes are likely to appear.

High Reynolds numbers

Since the critical modes arise in a regime of $Re \approx 800$ for the standard cavity with $\Gamma = 1$, we will take a look at how the basic flows behave, if the Reynolds numbers are increased. The behaviour of the basic flow with increasing Reynolds numbers is depicted in Appendix B, where a gallery of the basic flows for the Reynolds numbers between 100 and 1000 is given. Since we are only interested in the convergence behaviour with respect to grid size, we will only deal with $Re = 800$ here. First, comparing Fig. 3.1 with Fig. 3.5 we see that the relative magnitude of the velocity compared to the lid velocity intrudes deeper into the cavity. Taking a look at the resulting lines with mesh refinement in Fig. 3.6, we can deduce that a 70×70 mesh looks quite well converged, compared to a 90×90 mesh, except for the very close vicinity to the boundary. The refinement procedure described in the previous section seems to give a sufficiently converged result also in the case of higher Reynolds numbers. For the stability study, we will therefore start with meshes of 70×70 to get an estimate for critical

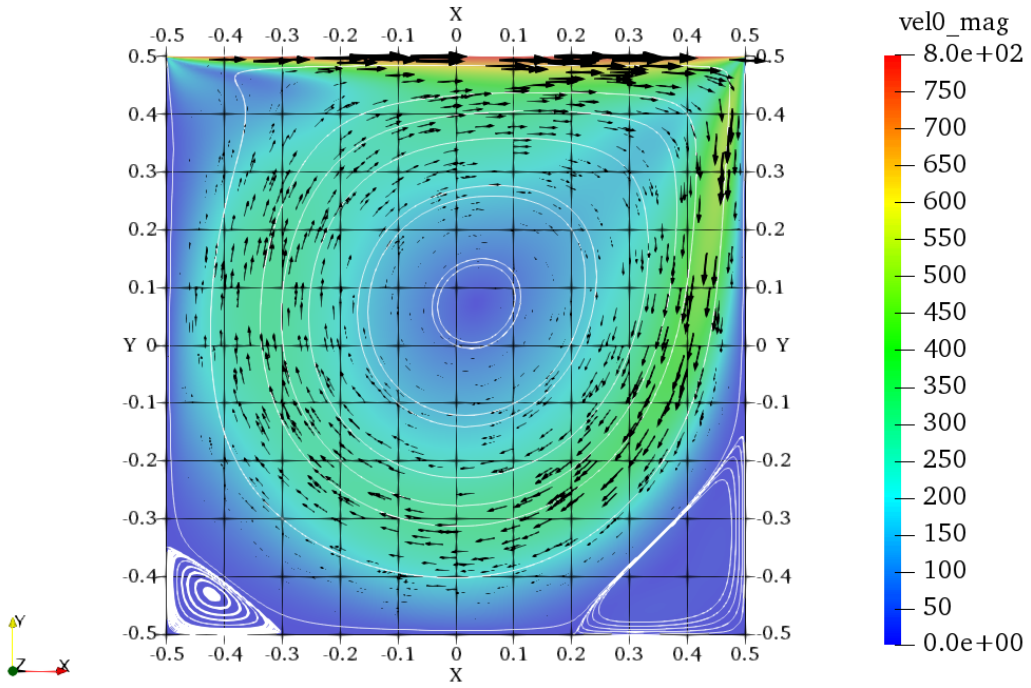


FIGURE 3.5: $Re = 800$ flow. The arrows denote the velocity field and the contour plot its magnitude with the respective legend. The white lines show the streamlines.

wavenumbers, which will be analyzed by a subsequent mesh refinement, to check for convergence of the generalized eigenvalue with mesh size.

Intermediate Reynolds numbers

The previous calculations have shown, which mesh resolution is necessary in order to obtain an accurate result. Here we will use sufficiently dense meshes to discuss the impact of the Reynolds number on the basic flow for $\Gamma = 1$. Increasing the Reynolds number results in a larger vertical penetration depth of the magnitude of the flow velocity inside the cavity, which is shown in Figure 3.7. Comparing the behaviour of the components of the velocity field for different Reynolds numbers along the diagonals of the cavity, we also see that the extrema of u and v shift more towards the boundaries of the cavity, the higher the Reynolds number. The corresponding Figures B.6 and B.7 are presented in Appendix B. Up to now, we obtained an estimate for the necessary mesh density within the method of finite elements. In the next sections, we will take a look at the impact of the computational method in use and whether the calculated results are reproducible with another method.

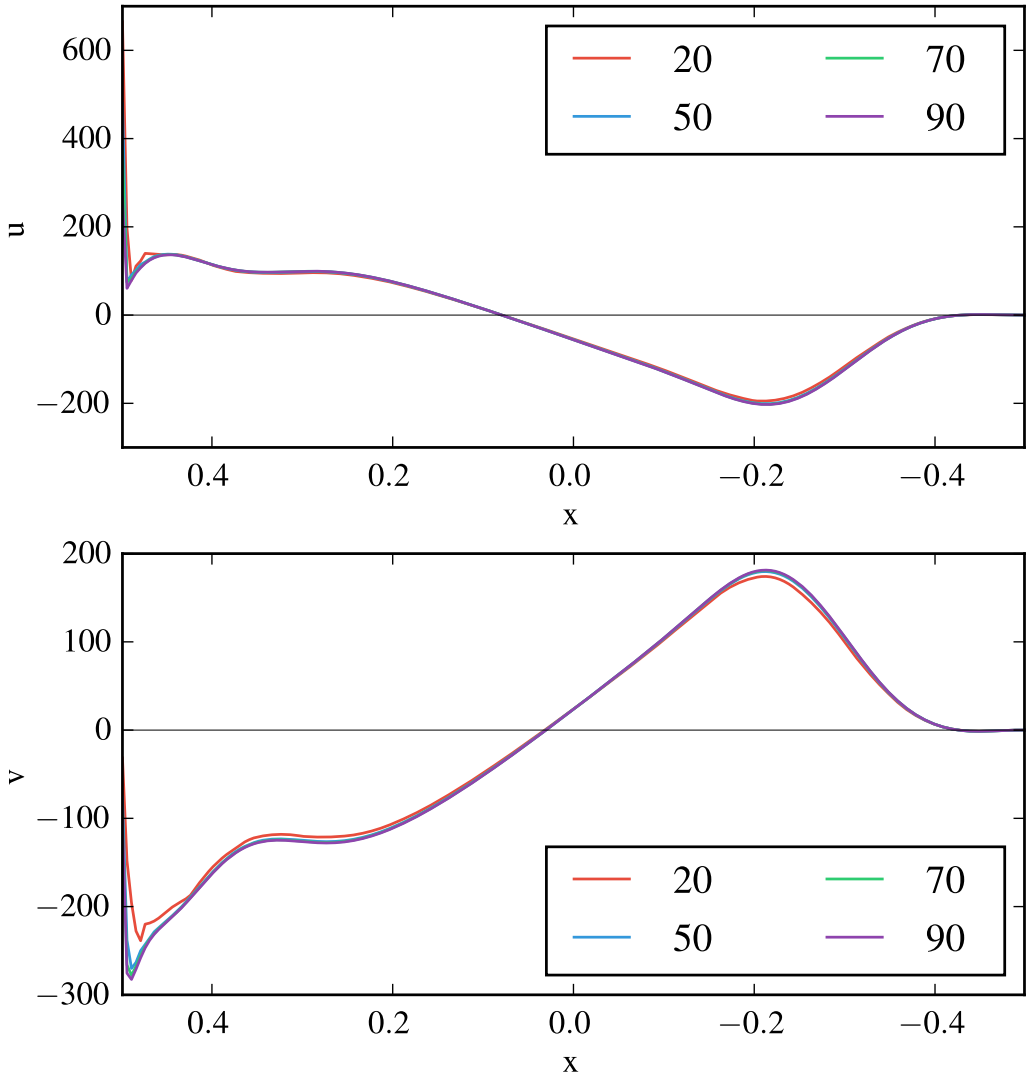
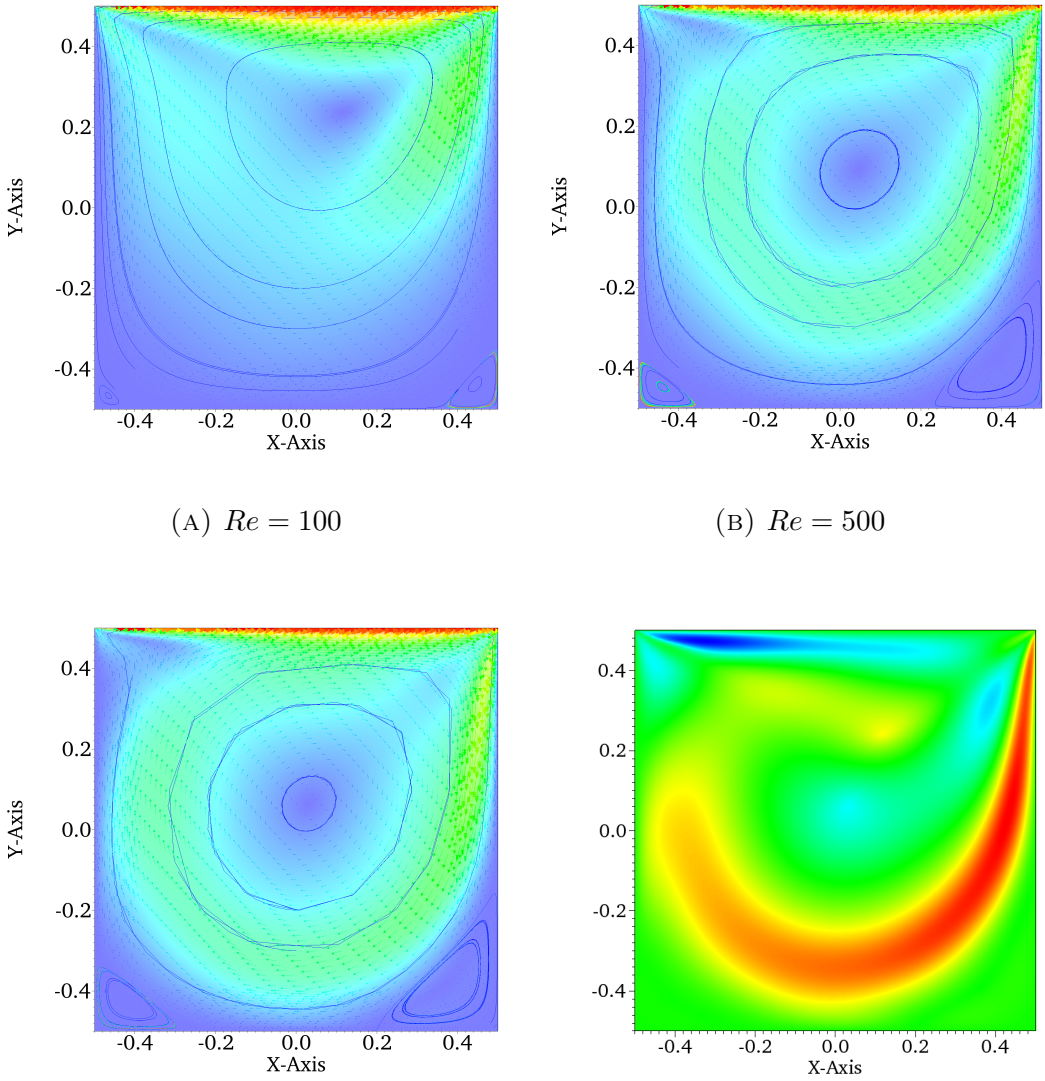


FIGURE 3.6: Grid convergence. The two plots depict the x- and y-components of the velocity field along the line $y = x - 0.5$ with refinement towards the boundary.



(A) $Re = 100$

(B) $Re = 500$

(C) $Re = 1000$

(D) Difference between $Re = 1000$ and $Re = 100$

FIGURE 3.7: Basic flows. The stationary flows for three Reynolds numbers (A-C) and the comparison between two flows (D, normalized with respect to the lid velocities). In (A-C) the colors denote the magnitude of the velocity field, where red corresponds to big and blue to small values. In (D) red depicts positive and blue negative differences in the velocities, where the maximal relative difference is 0.3649 and the minimal is -0.38 .

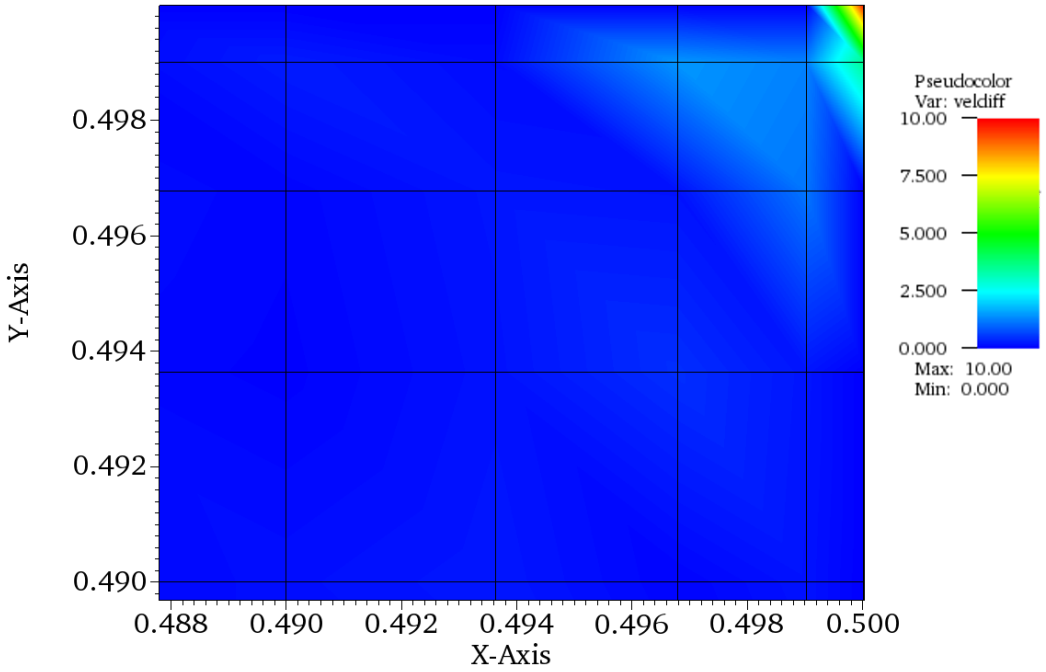


FIGURE 3.8: $Re = 10$ differences SEM-FEM. The difference of the magnitude of the velocities on the right top corner of the cavity is shown ($\sqrt{\vec{v}_0^{SEM} \cdot \vec{v}_0^{SEM}} - \sqrt{\vec{v}_0^{FEM} \cdot \vec{v}_0^{FEM}}$). We see the top right element and 5 Legendre-Gauss-Lobatto points in each direction.

3.1.2 Comparison with spectral elements

In order to show the independence of the obtained results on the chosen method, we consider the flows of the previous sections and compare them with results from a spectral element method using the code NEK5000. For the calculations of the flows, we use 9th order polynomials inside the elements on Legendre-Gauss-Lobatto points. The comparison between the flows using a 20×20 and a 50×50 element cavity shows, that convergence is obtained for the case of 20×20 elements. Choosing this resolution, we see that the difference between the finite element approach and the spectral element method is negligible inside the cavity. The biggest difference is visible on the top corners, shown in Fig. 3.8 and may as well be a consequence of the interpolation algorithm used by VisIt similar to the case of in previous section. Also in the case of higher Reynolds numbers, we only encounter a difference in the magnitude of the velocity in the corners of the cavity, as depicted in Figure 3.9.

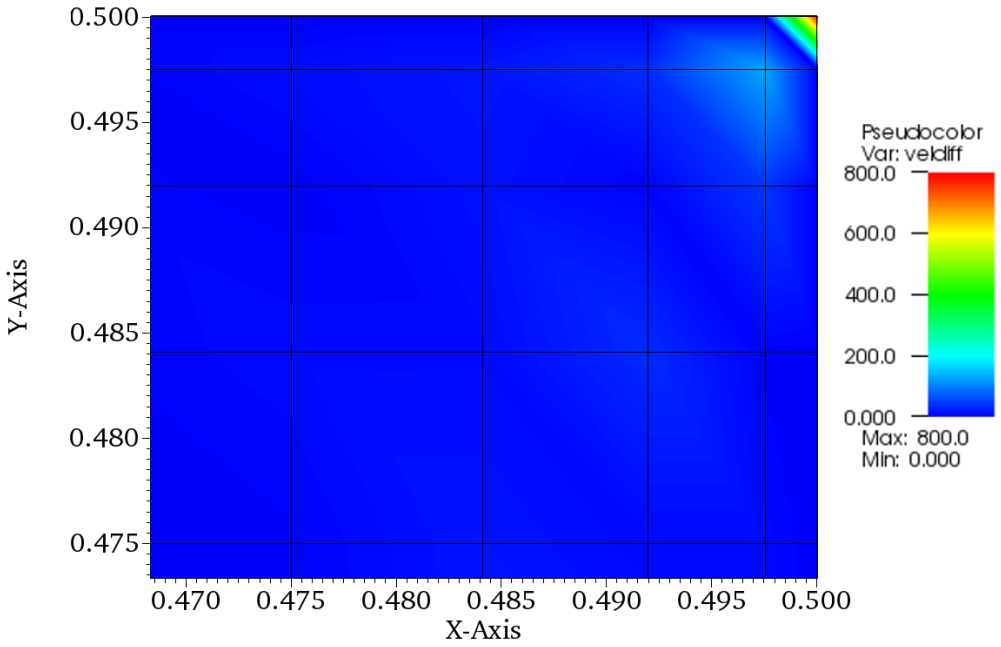


FIGURE 3.9: $Re = 800$ differences SEM-FEM, The difference of the magnitude of the velocities on the right top corner of the cavity is shown. We see the top right element and 4 Legendre-Gauss-Lobatto points in each direction.

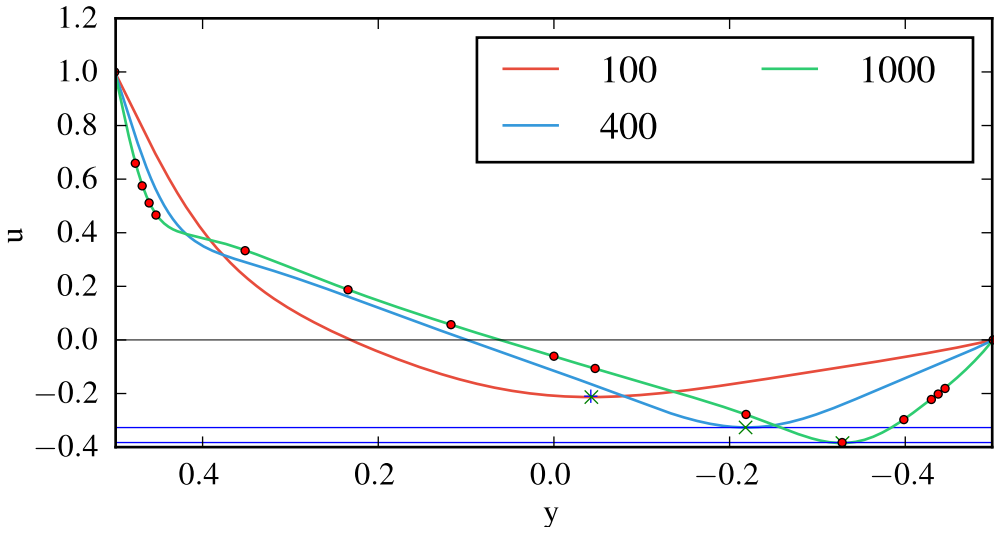


FIGURE 3.10: Literature comparison u on the vertical center line, the solid green, blue and red line show the calculated component of the u velocity component at the vertical center line. Green crosses mark the minimum of u along the line. The red dots are the results given in Ghia, Ghia, and Shin (1982), the blue horizontal lines mark the minimum of u for $Re = 400$ and $Re = 1000$ and the blue cross gives the position and minimal value as calculated by Botella and Peyret (1998) and Deng et al. (1994)

Basic flows - comparison with literature

To triple-check our results, we compare the calculated flows with some already published results. For the case of $Re = 100$, $Re = 400$ and $Re = 1000$, Botella and Peyret (1998) and Deng et al. (1994) gave a number of values for the magnitude of the stream function and the vorticity [4]. We will only compare the stream function, since we have chosen a formulation of the equations, where velocity components are most easy to evaluate. The results in Figure 3.10 and Figure 3.11 show that the agreement with previous studies is given to graphical accuracy, because all the literature values lie on the calculated lines.

3.2 $\Gamma = 2$ Basic flows

To shorten the lengthy convergence studies, we will present only the most relevant facts here and the interested reader is provided with additional plots in Appendix B. First, we state that the results of the convergence study show, that

[4]The vorticity $\vec{\omega}$ is given by the curl of the flow velocity: $\vec{\omega} = \vec{\nabla} \times \vec{u}$. It is a measure for the local rotation of a fluid.

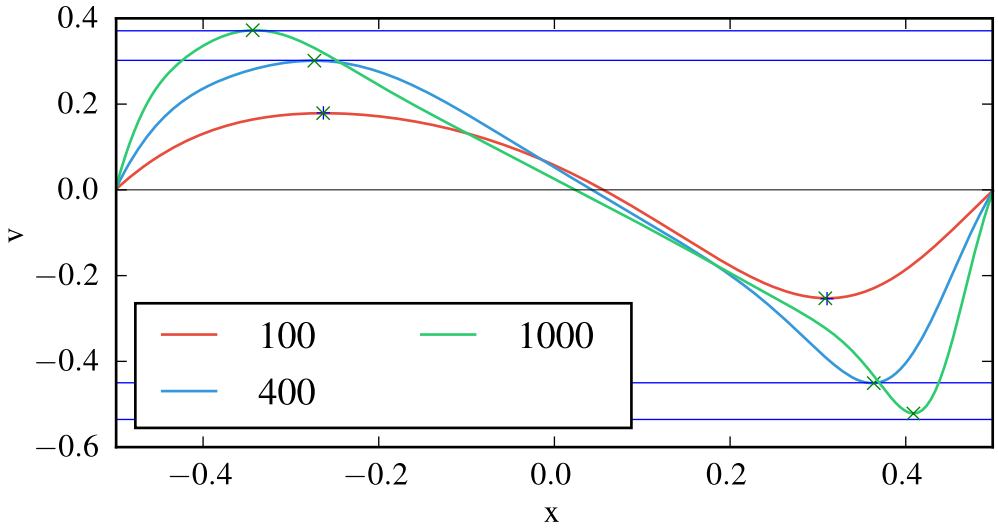


FIGURE 3.11: Literature comparison v on the horizontal center line, the solid green, blue and red line show the calculated component of the v velocity component at the horizontal center line. Green crosses mark the calculated extrema of v along the line. The blue horizontal lines mark the minima and maxima of v for $Re = 400$ and $Re = 1000$ and the blue crosses give the position and minimal value as calculated by Botella and Peyret (1998) and Deng et al. (1994)

a mesh of 70×70 grid points before the refinement procedure gives converged results for the velocity components along the diagonals and center lines. The refinement is again performed in a threefold way, s.t. that the points which are 0.6, 0.3 and 0.1 away from the boundary are doubled in the first, second and third refinement, respectively. This shows, that a proper resolution at the boundary allows for a quite coarse grid in the center. With the converged results, we obtain flows for the $\Gamma = 2$ case, which are shown in Figure 3.12 and we see that the deeper cavity has enough space to allow for a second counter-rotating vortex, whose size and center is dependent on the Re-number and marches towards the boundary for higher Reynolds numbers as the line-plots in Appendix B.2.1 illustrate.

3.3 $\Gamma = 3$ Basic flows

As physicists' induction would suggest, the cavity with an aspect ratio $\Gamma = 3$ consists of three vortices, counter-rotating from top to bottom, whose sizes are given by the Reynolds numbers. The flows and the change of the flow pattern with increasing Reynolds number is given in Figure 3.13. The mesh convergence shows, that we need to be careful, when it comes to deeper cavities because a

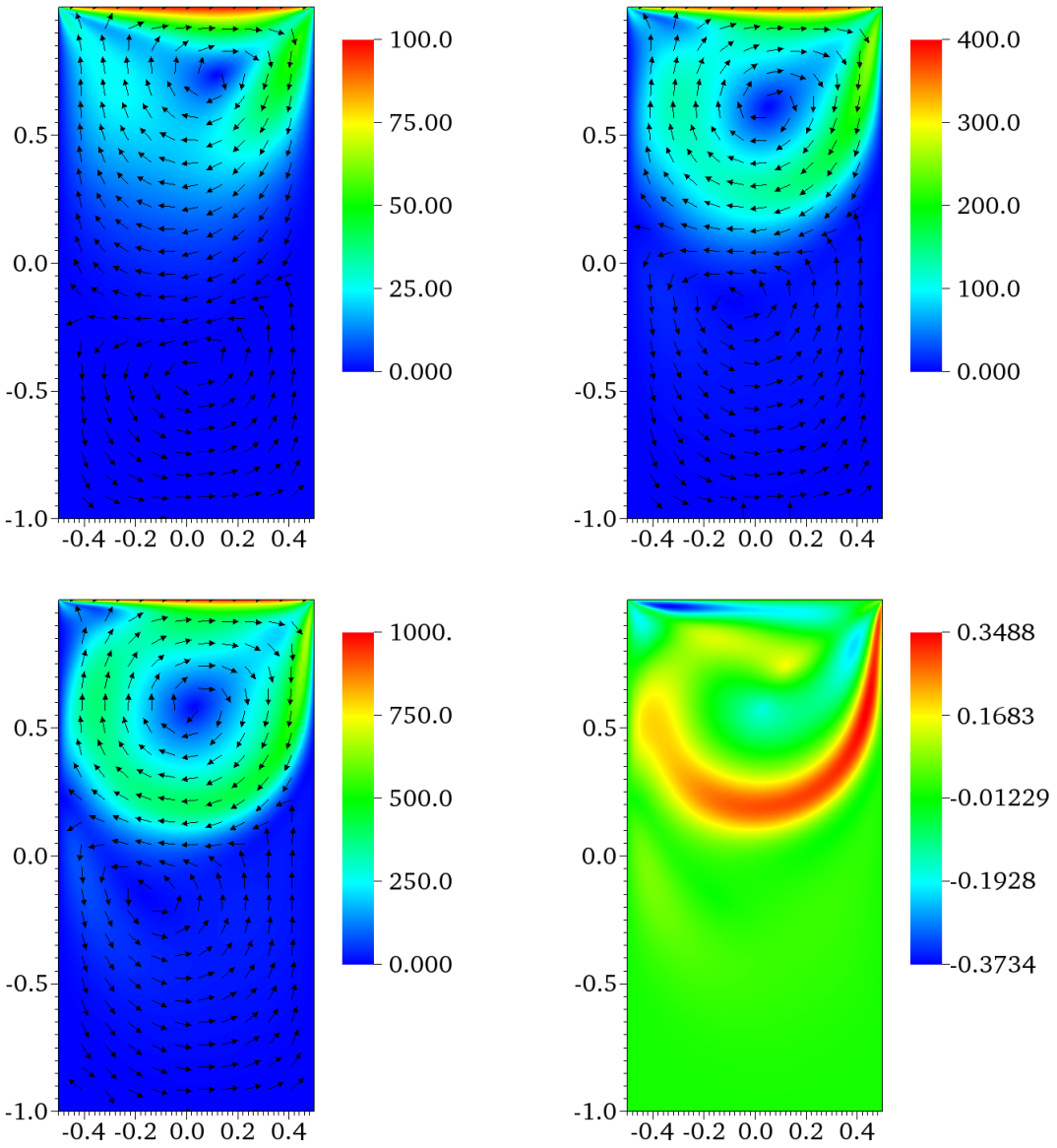


FIGURE 3.12: Re comparison for $\Gamma = 2$
top left: $Re = 100$
top right: $Re = 400$
bottom left: $Re = 1000$
bottom right:$(Re = 100) - (Re = 1000)$
The colours denote the magnitude of the velocity and the arrows the direction. The size of the arrows does not represent the magnitude of the velocity but only its direction.

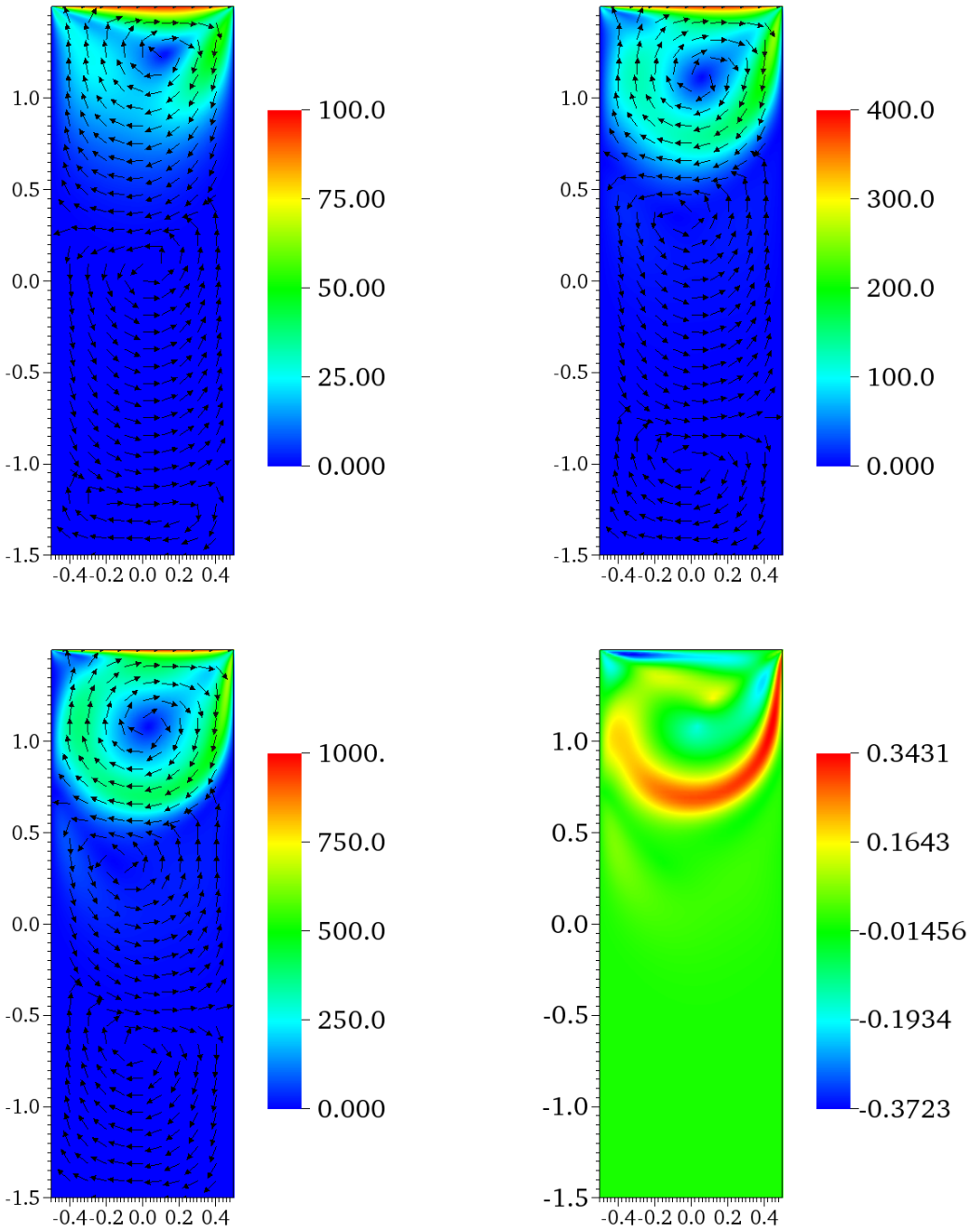


FIGURE 3.13: Re comparison for $\Gamma = 3$
top left: $Re = 100$
top right: $Re = 400$
bottom left: $Re = 1000$
bottom right:$(Re = 100) - (Re = 1000)$
The colours denote the magnitude of the velocity and the arrows the direction. The size of the arrows does not represent the magnitude of the velocity but only its direction.

mesh of 70×70 grid points is not converged up to graphical accuracy and may therefore yield wrong results for the stability analysis. When we compare a grid with 70×140 grid points before refinement, with a 100×100 grid, we see that convergence is obtained, when the number of grid points in y-direction is doubled. The corresponding plots are shown in Appendix B.3.1.

3.4 $\Gamma = 0.5$ basic flows

The last cavity we investigate has an aspect ratio $\Gamma = 0.5$. The convergence study in Appendix B.4.1 shows, that a 70×70 starting mesh results in well converged results, which are in agreement with the spectral element method. This aspect ratio cavity is interesting because of the evolution of a second vortex on the left side of the cavity for higher Reynolds numbers as Figure 3.14 reveals.

3.5 Conclusion – convergence studies

The convergence studies have shown that the basic flows are properly resolved by a refinement of the mesh towards the boundary, where a starting mesh of 70×70 grid points seems to be sufficient to get an estimate for critical Reynolds numbers. Care has to be taken when the aspect ratio is increased above $\Gamma = 2$, where a larger number of elements in y-direction seems to be appropriate to keep the resolution constant and obtain converged results. We saw that the numerical code developed is able to reproduce published results to graphical precision and can therefore be confident that the basic flows we calculate are correct. For the actual calculations we obtained an impression on how dense the mesh has to be. We will, however, always double check if the obtained result for critical Reynolds numbers and wave vectors is correct, by doing a mesh refinement to investigate the change in the eigenvalues of the generalized eigenvalue equation (2.27).

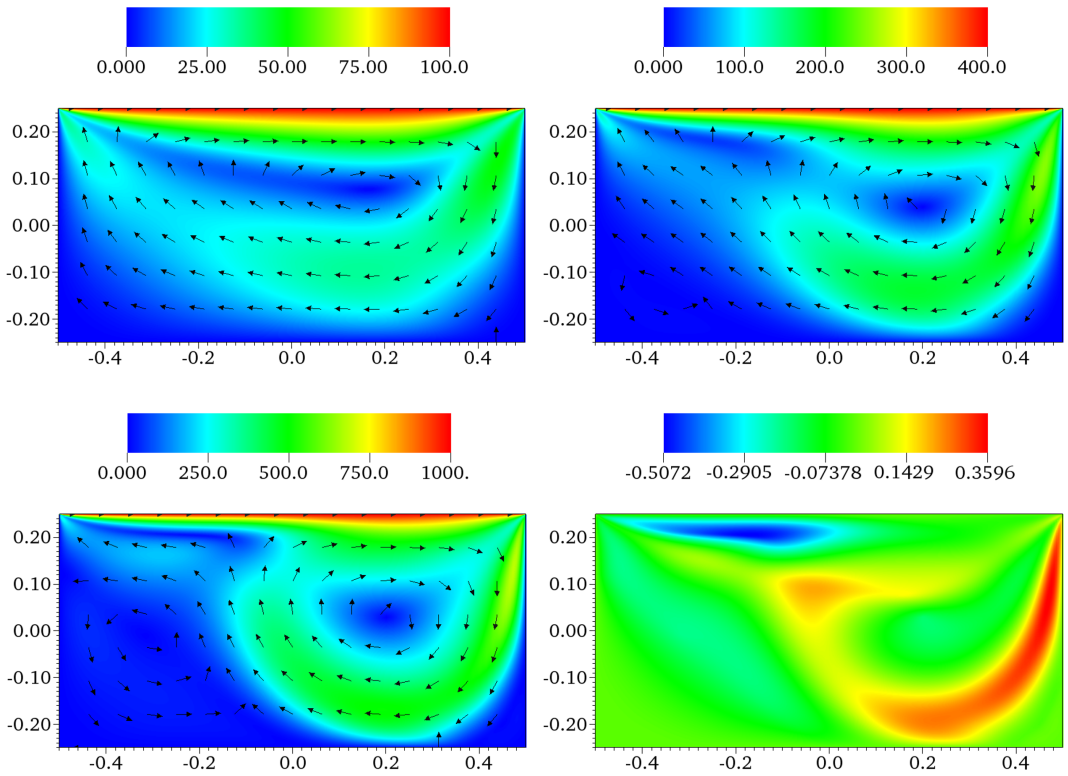


FIGURE 3.14: Re comparison for $\Gamma = 0.5$
top left: $Re = 100$
top right: $Re = 400$
bottom left: $Re = 1000$
bottom right: $(Re = 100) - (Re = 1000)$
The colours denote the magnitude of the velocity and the arrows the direction. The size of the arrows does not represent the magnitude of the velocity but only its direction.

Chapter 4

Stability analysis

In this chapter, we will use the knowledge of the previous convergence studies and basic flow calculations to disturb the basic flows and obtain the critical modes and Reynolds numbers of the flow. First, we will start to reproduce already published results for $\Gamma = 1$ and $\alpha = 0$ from Albensoeder, Kuhlmann, and Rath (2001), to check, if the linear stability code works correctly and then we will start to produce new results for different aspect ratios and drive angles. For this purpose a Python class has been developed (the source code is given in Appendix D), where the necessary functions are implemented to

- find Re_c for a given aspect ratio and drive angle by subsequent bisectioning of a given interval of Reynolds numbers. The borders of the input interval Re_{low} and Re_{high} should span a region, where critical modes are expected. The resolution of the mesh is adapted, such that we verify the values, where the critical values are arising by a second calculation with a refined mesh (function: findRe_c)

- perform a stability analysis for an array of Reynolds numbers at a given aspect ratio to extract the functional dependence of σ on the wavenumber. If a sign change occurs, a subsequent mesh refinement for the wavenumber interval of interest is also implemented in this function (function: analyze).

In order to call these functions in a more user friendly way, a bash script was written, which allows for a hybrid of the previously mentioned functions by a manual change of the wavenumbers of interest in an input file, such that the Python code does not have to be modified for each particular cavity.
These tools allow us to find the critical Reynolds numbers and the values of k, where the real part of the growth rate σ changes its sign. The size of the parameter space we will investigate in this thesis depends on the computation time of the calculations. If all the calculations converge at the first trial, which is usually not the case, the calculation of 4 angles for a single aspect ratio takes about 3 days on a single core with a very coarse grid in the wavenumber range.

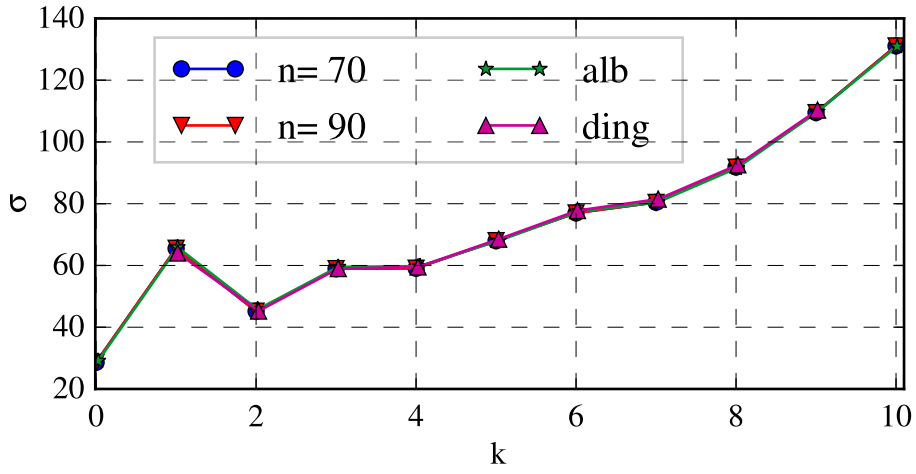
4.1 Orthogonal lid motion for $\Gamma = 1$

The linear stability analysis of this standard cavity was performed by Albensoeder, Kuhlmann, and Rath (2001), using finite volumes, and by Ding and Kawahara (1999), using finite elements. A comparison between our minimum damping rates σ and angular frequencies ω and theirs are provided in Figure 4.1 and Figure 4.2. The energy transfer rates also agree well with the results obtained in Albensoeder, Kuhlmann, and Rath (2001), both in the magnitude as well as in the localization (see Figure 4.3). The comparison reveals that our code is working well and is able to reproduce published results.

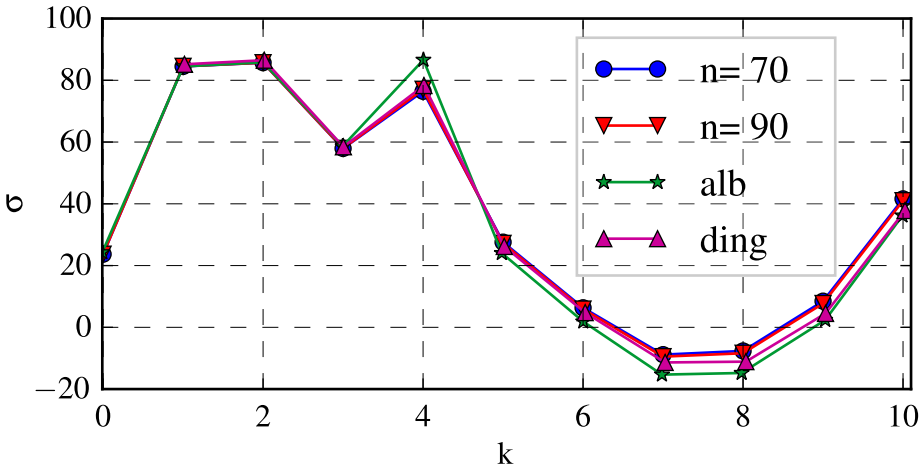
4.2 Stability of oblique cavity flow

The flows corresponding to lid motions with inclination angles $\alpha = 22.5°, 45.0°$ and $67.5°$ in the x-z plane were analyzed by Theofilis, Duck, and Owen (2004). In their paper, they did not find any critical mode below $Re = 800$ for these drive angles, which is in disagreement with this study. As Figure 4.4 reveals, we find critical modes below $Re = 800$ for the drive angles $22.5°$ and $67.5°$. Due to this discrepancy with published results, we also perform an independent three-dimensional non-linear flow simulation of the full Navier–Stokes equation in NEK5000 for the drive angle of $22.5°$. From Figure 4.5 we can extract that indeed the velocity at a fixed position starts oscillating [2]. The use of the second method clearly confirms our result by showing that for the $Re = 630.7$ flow oscillations arise and grow in time. In addition, the energy analysis suggests, that the production rate exceeds the dissipation rate for the mode of interest, which is another hint for the correctness of the calculation. We also see in Figure 4.4, that the behaviour of the critical parameters follows a non-trivial curve: Increasing the angle from $\alpha = 0°$ to $\alpha = 7°$ does not induce a big change in the critical parameters. This plateau is followed by a decrease in Re_c, which reaches a minimum at an angle of $\alpha \approx 22.5°$. The subsequent rise towards a maximal value of $Re_c \approx 880$ at an angle of $\alpha = 32°$ is followed by a decrease towards $Re_c \approx 640$ for an angle of $\alpha = 55°$. There is another plateau between $\alpha = 55°$ and $\alpha = 75°$, where there is no rapid change in Re_c, followed by a small jump at a value of $\alpha = 80°$.
The limit of an angle of $\alpha = 90°$ corresponds to a wall bounded Couette flow.

[2] The preliminary three-dimensional simulations were performed for an increased Reynolds number than the predicted critical one to assure the appearance of the instability. The simulations. For $Re = 750$ and $k = 7.24$ the three-dimensional simulation gives $\omega = 586.8 \pm 8.2$, which is reproduced by our calculation for this setting, yielding $\omega = 579.18$.

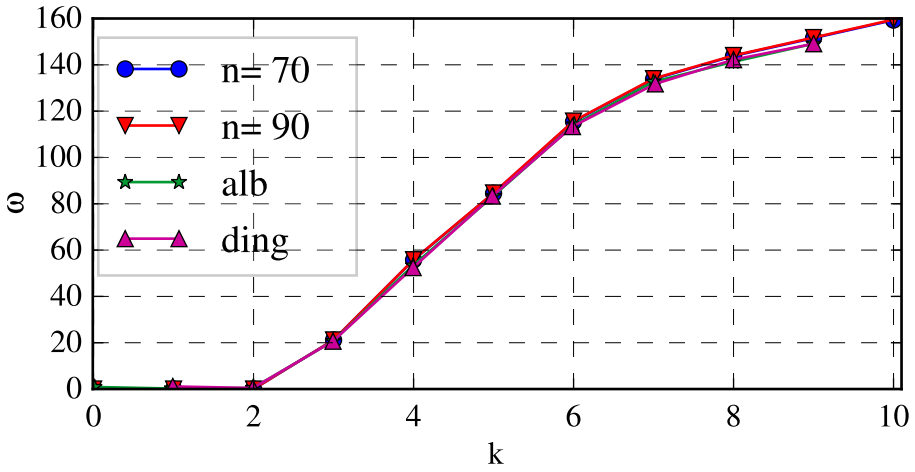


(A) $Re = 200$

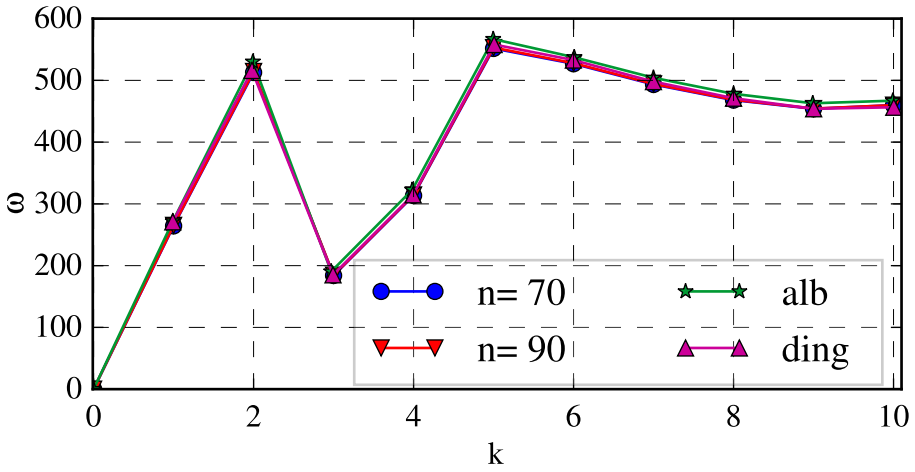


(B) $Re = 1000$

FIGURE 4.1: σ comparison with literature, the calculated damping rate σ as a function of the wavenumber k is compared with the results of Ding and Kawahara (1999) and Albensoeder, Kuhlmann, and Rath (2001). We see that our calculations are closer to the results of Ding and seem to be converged for a starting mesh of 70×70 grid points before refinement.

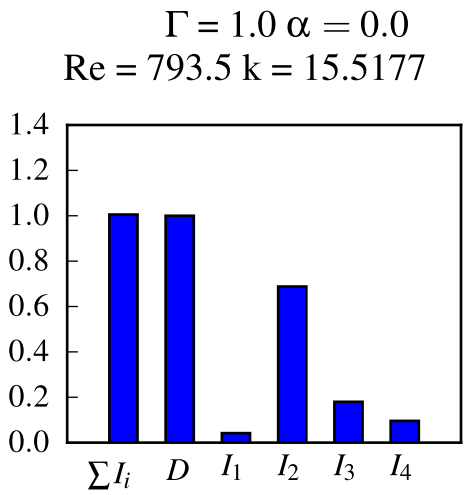


(A) $Re = 200$

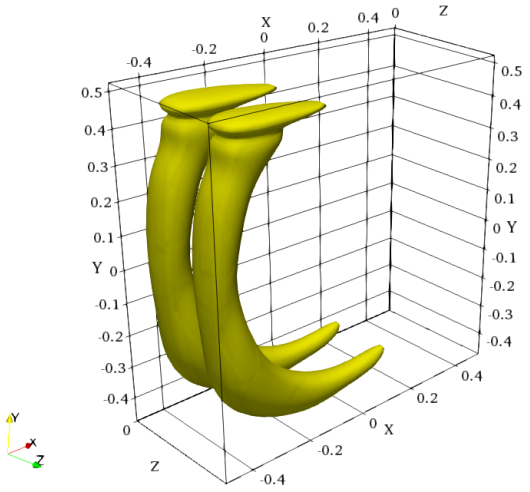


(B) $Re = 1000$

FIGURE 4.2: ω comparison with literature. The calculated circular frequency is plotted versus k and compared with Ding and Kawahara (1999) and Albensoeder, Kuhlmann, and Rath (2001) for two different Reynolds numbers. All results agree within a few percent.



(A) Integral energy contributions for the critical mode.



(B) The total energy production rate for the critical mode at $Re_c = 793.5$ and $k = 15.5117$. Shown is the isosurface corresponding to 1/5 of the maximal value. The overall maximum of the energy production is located near the moving lid and the isosurfaces agree with Albensoeder, Kuhlmann, and Rath (2001).

FIGURE 4.3: Energy analysis for the critical mode.

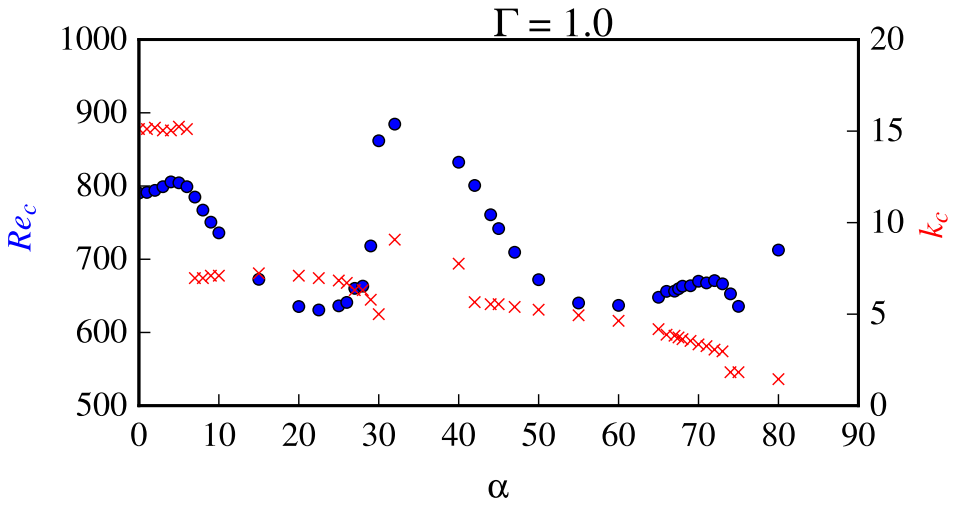


FIGURE 4.4: Critical Reynolds numbers and wavenumbers as a function of α for $\Gamma = 1$. The red crosses mark the critical wavenumbers and the blue dots the critical Reynolds numbers.

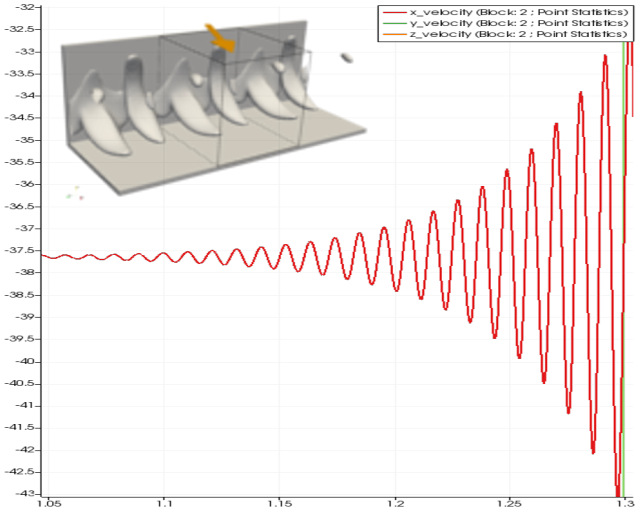


FIGURE 4.5: Three-dimensional simulation results with NEK5000 for the drive angle $\alpha = 22.5°$ with $\Gamma = 1$ and $Re = 750$. The plot shows the simulation box of the three-dimensional calculations with the lid velocity as an orange arrow. An isosurface of the magnitude of the perturbation velocity is presented. The line plot shows the rising velocity at a fixed position in the cavity and confirms the results obtained with the linear stability analysis. The x-axis corresponds to the time and the y-axis to the magnitude of the velocity field [1].

This flow was analyzed by Theofilis, Duck, and Owen (2004), where they did not find any criticality. We also performed a stability analysis for this angle and were likewise unable to find any critical modes for $Re < 3000$.

An analysis of all the critical modes for such a large parameter space is impossible and therefore we will focus on the main features apparent, when the angle is varied. Apart from the already clarified discrepancy with Theofilis, Duck, and Owen (2004), there are two interesting jumps occurring in the behaviour of $Re_c(\alpha)$ for $\Gamma = 1$, which shall be analyzed in the following: One jump is occurring in the behaviour of the critical wavenumbers: We see a change in the critical wavenumber arising in the small angle regime: For angles $\alpha < 7°$, the critical mode is around $k = 15$, whereas the spatial periodicity for larger angles is given by wavenumbers with $k_c < 5$. Taking a look at the k-σ curves we realize, that there are two minima of σ, which are close to criticality and the change of the critical mode is arising between these two angles (Figure 4.6). Since the plot in Figure 4.6 includes points, where the eigenvalue changes rapidly, when k is increased, we will check for this behaviour by a refinement of these critical regions: this is done exemplary for the jump of the second eigenvalue of the $\alpha = 7°$ case in the regime between $k = 9.3$ and $k = 9.46$, because we want to

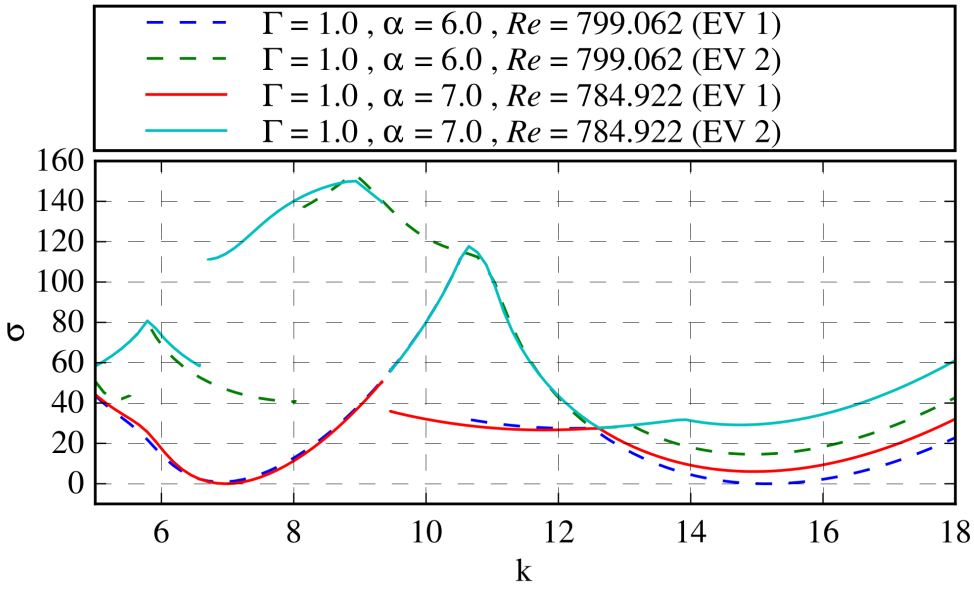


FIGURE 4.6: $\alpha = 6°$ and $\alpha = 7°$, the k - σ plots for the analysis of the jump in critical wavenumbers.

see, if our eigenvalue solver is working correctly: We compare the two lowest eigenvalues for eight wavenumbers in this region in Table 4.1 and deduce that the jump is an abrupt change of σ with k, which means that the eigenvalue solver suddenly finds a new mode. Whether this is of physical origin or owed to the numerics can not be answered at this point.

Due to the fact, that the periodicity of the modes differs a lot, we are also interested in the energy transfer terms and a localization of the energy gain. Figure 4.7 shows the isosurfaces of the production rate in three dimensions. The shapes of the isosurfaces look different: As in the case of $\alpha = 0$ in Figure 4.3,

k	σ_1	σ_2	ω_1	ω_2
9.33	50.618	139.72	2.41	232.60
9.37	53.64	142.62	53.82	283.68
9.39	54.40	141.93	53.18	283.07
9.41	55.16	141.24	52.56	280.47
9.43	55.93	140.55	51.95	278.88
9.44	54.81	137.04	1.24	223.67
9.45	55.21	136.79	1.57	222.84
9.46	35.86	55.81	674.42	2.05

TABLE 4.1: Jumps in eigenvalues. The table depicts the behaviour of the eigenvalues for $\alpha = 7°$ and $Re = 784.922$ and corresponds to a section of the turquoise line in Figure 4.6.

there is a banana shaped isosurface at the left side of the cavity. Due to the oblique lid motion these bananas are altered according to the periodicity, given by the wavenumber. The main difference between the two modes are arising in the top right corner. If we take a closer look at the slice $y = 0$, as shown in Figure 4.8, we can learn about the mechanism that drives the instability and see the different locations of the production rate between the two modes: The energy production is dominated by the term I_2. There are two vortices in the surrounding of positive energy production. This may be understood from the energy analysis: Due to the boundary conditions, we have a shear layer with big velocity gradients in x-direction and a basic flow in y-direction which gives gives a big contribution to I_2 because the perturbation between the vortices is orthogonal to the basic flow and we see in Figure 4.8 that also the parallel contribution is significant at this location, as the maximal isosurfaces of v are in the vicinity of the maximal production rate. The effect of the different localization on the energy production rate contribution is very small, as depicted in Figure 4.9, which means that the above mentioned mechanism is dominant in both flows. There is only a small shift of weight from I_1 to I_3, going from $\alpha = 6°$ to $\alpha = 7°$, but the most prominent contribution still comes from I_2, which resembles the behaviour that was analyzed in Albensoeder, Kuhlmann, and Rath (2001). There is a second interesting slice to look at for the critical mode of $\alpha = 7$, which is at an height of $y = 0.2$, where the nose shaped feature is located. As we show in Appendix C, this feature stems from the energy transfer rate I_3.

The second drastic change in the behaviour of the flow is arising between the angles of $\alpha = 22.5°$ and $\alpha = 30°$, where Re_c is changing from 631 to 862 without a big change in k_c. As illustrated in Figure 4.10, the energy transfer term distribution is similar for the two observed modes. The critical wavenumber for $\alpha = 22.5°$ is $k_c \approx 7$, whereas the critical mode for $\alpha = 30°$ is arising at $k \approx 5$ (see Figure 4.11 and Figure 4.12). The spatial localization of the energy gain for the modes is clarified in Figure 4.12. There we show that the energy gain is again along the bananas, as in the case of $\alpha = 0°$, but at the angle of $\alpha = 30°$, the energy production at $y = 0$ decreases and is shifted to the right side of the cavity (towards $x = 0.5$, downstream). Taking a closer look at the slice $y = 0$ allows to deduce that the vortex structure of the perturbation changes significantly going from $\alpha = 22.5°$ to $\alpha = 30°$. The vortices at $x = -0.5$ vanish for $\alpha = 30°$. The corresponding Figure 4.13 shows, that the absence of these vortices is accompanied by a reduction of I_2. This behaviour is explained by a reduced shear as the moving lid induces a flow, whose contribution in z-direction

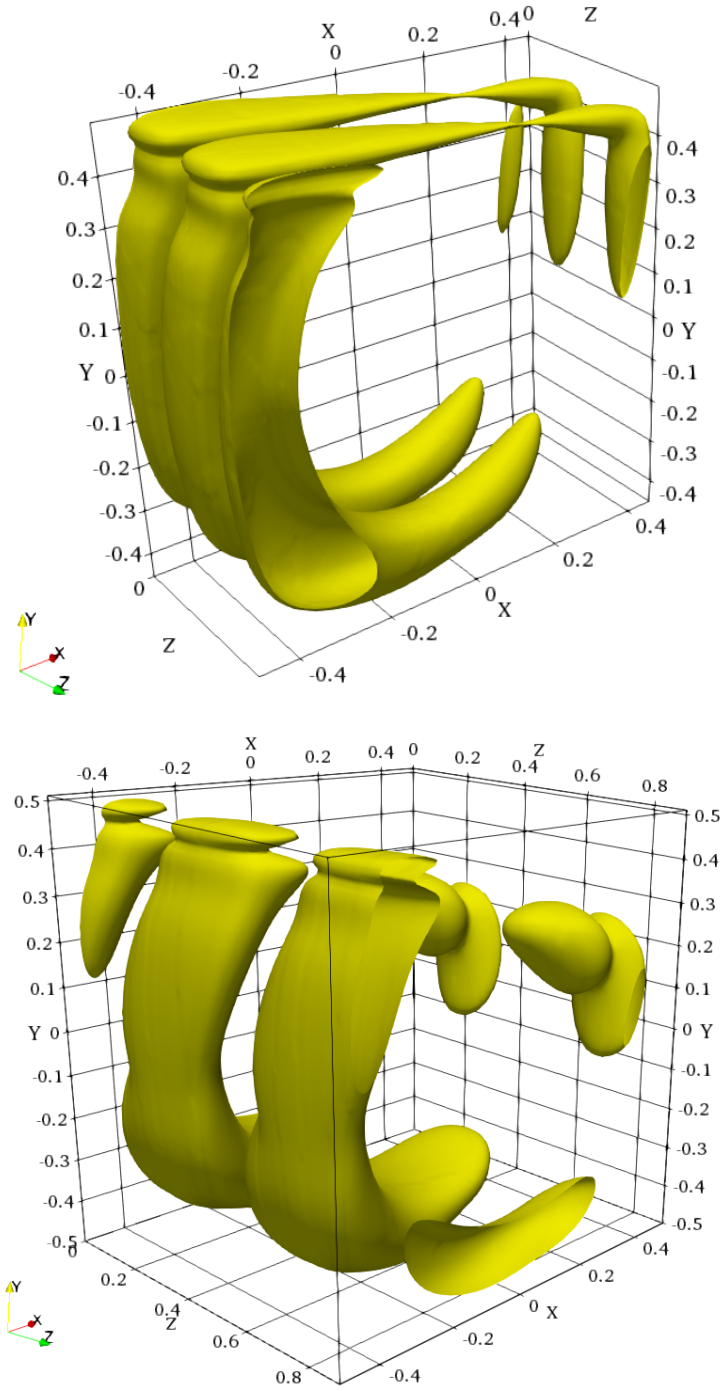


FIGURE 4.7: Energy productions of the critical modes for $\alpha = 6°$ (top) and $\alpha = 7°$ (bottom). An isosurface for a constant value of $\sum_i I_i / \max(\sum_i I_i)$ is shown.

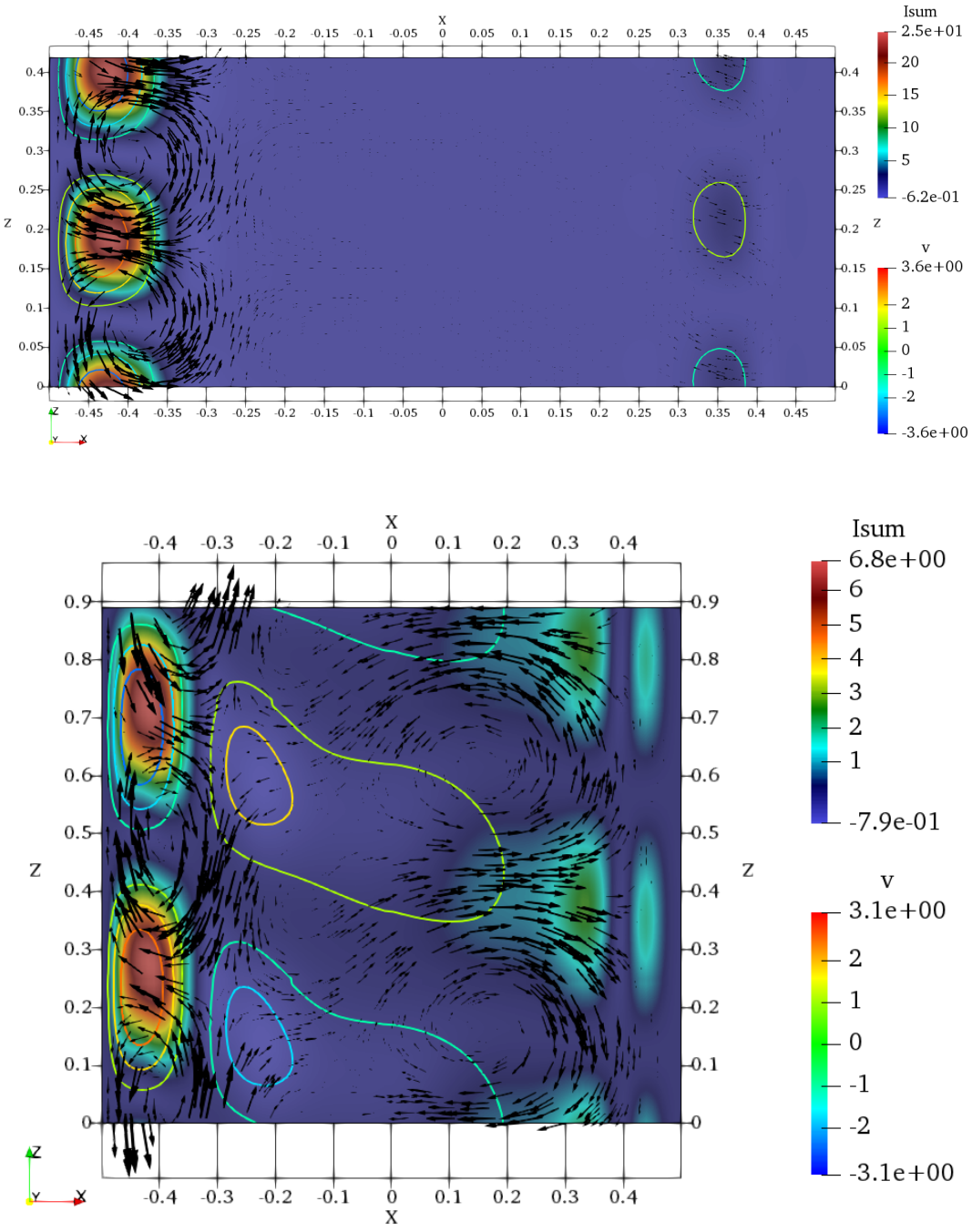


FIGURE 4.8: Energy productions of the critical modes for $\alpha = 6°$ (top) and $\alpha = 7°$ (bottom) at $y = 0$. The color denotes the production rate, the arrows denote the u and w components of the velocity and the isolines show the value of v, whose extrema are located at the maxima of the production rate.

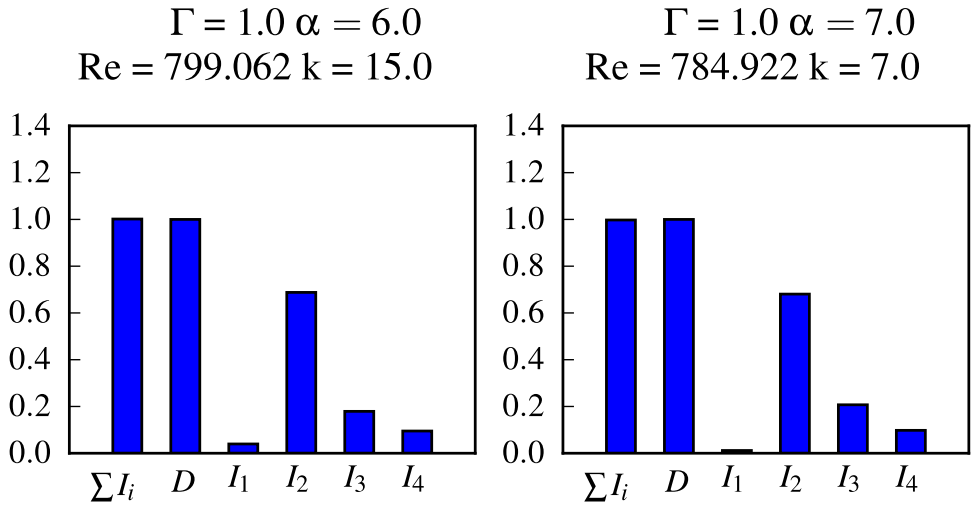


FIGURE 4.9: Energy contributions of the critical modes for $\alpha = 6°$ (left) and $\alpha = 7°$ (right).

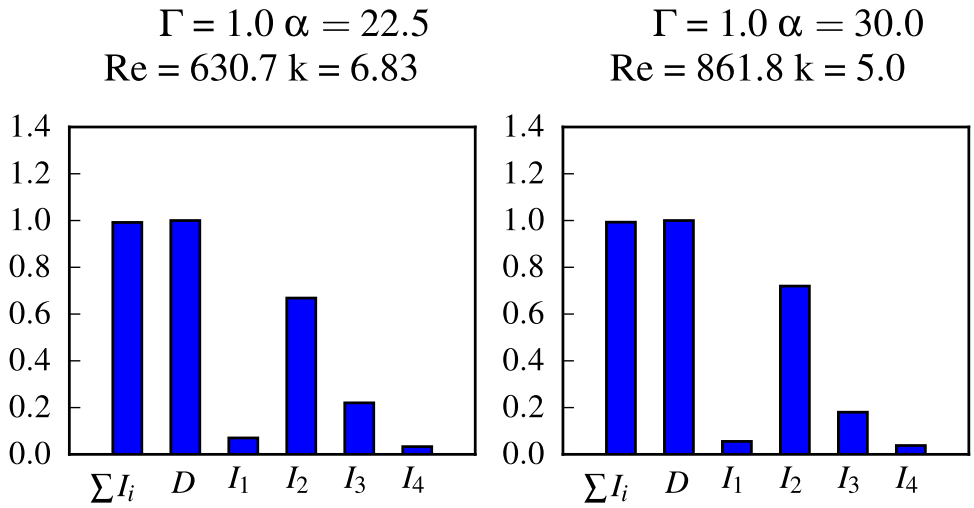


FIGURE 4.10: Energy contributions of the critical modes for $\alpha = 22.5°$ (left) and $\alpha = 30.0°$ (right)

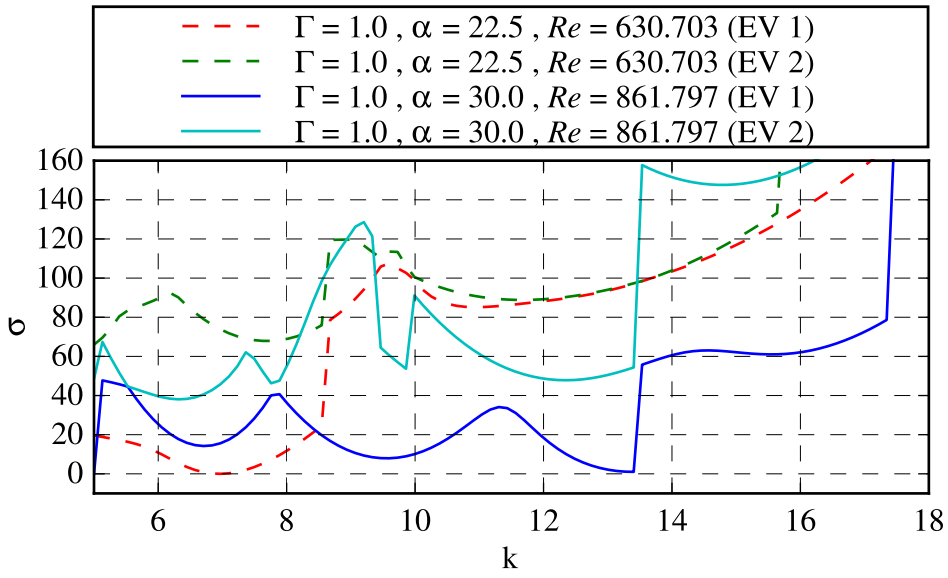


FIGURE 4.11: σ versus k for $\alpha = 22.5°$ and $\alpha = 30°$. We see that the eigenvalues show jumps, where the eigenvalue solver goes to the next eigenvalue, if k is increased. The red line shows, that there is a second mode, where we nearly obtain instability at $k \approx 13$.

increases.

As the shear layer in the $x - y$ plane is diminished are reduced for an angle of $\alpha = 30°$, we are interested in flows with a very large drive angle. So the next cavity flow we analyze is the one with $\alpha = 60°$: The energy contributions show that I_2 is even more dominant than for smaller angles and the energy production is happening solely in the vicinity of the upstream wall of the cavity, in contrast to the cavity flows analyzed so far.

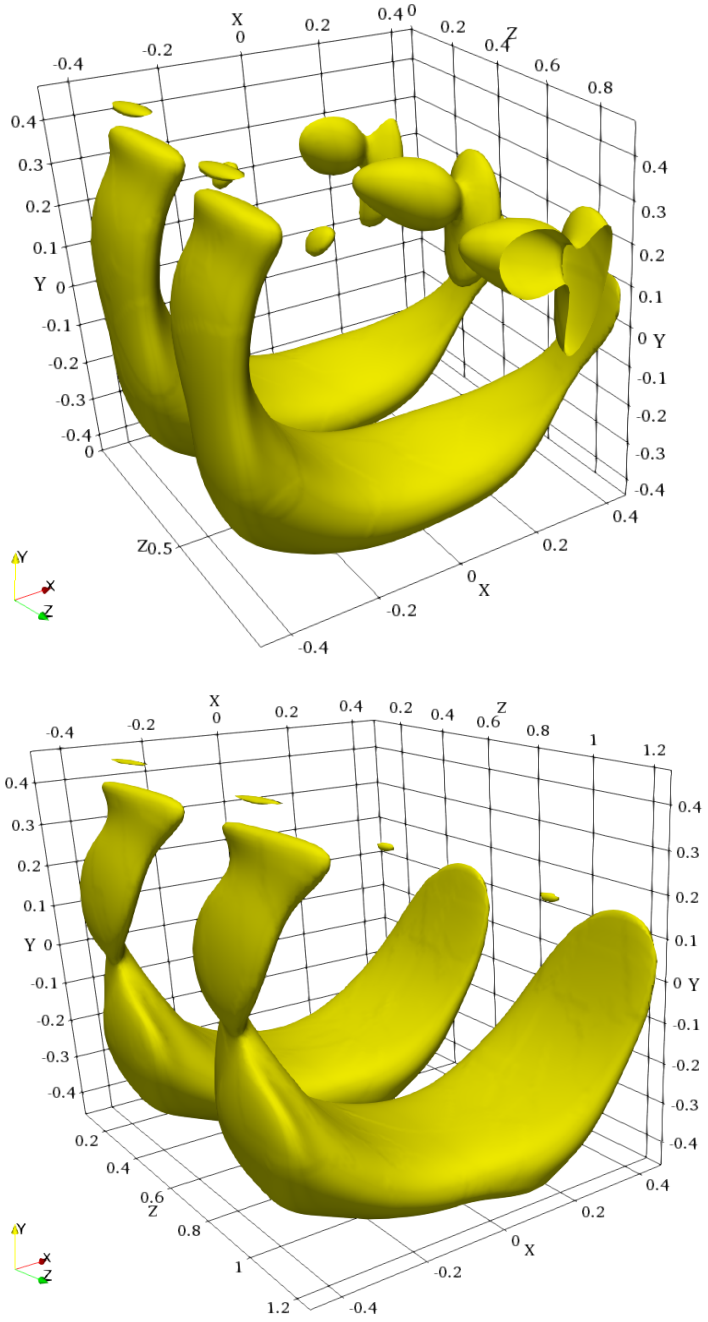


FIGURE 4.12: Energy productions of the critical modes for $\alpha = 22.5°$ with $k = 6.83$ (top) and $\alpha = 30.0°$ with $k = 5$ (bottom). An isosurface for a constant value of $\sum_i I_i / \max(\sum_i I_i) = 1/10$ is shown.

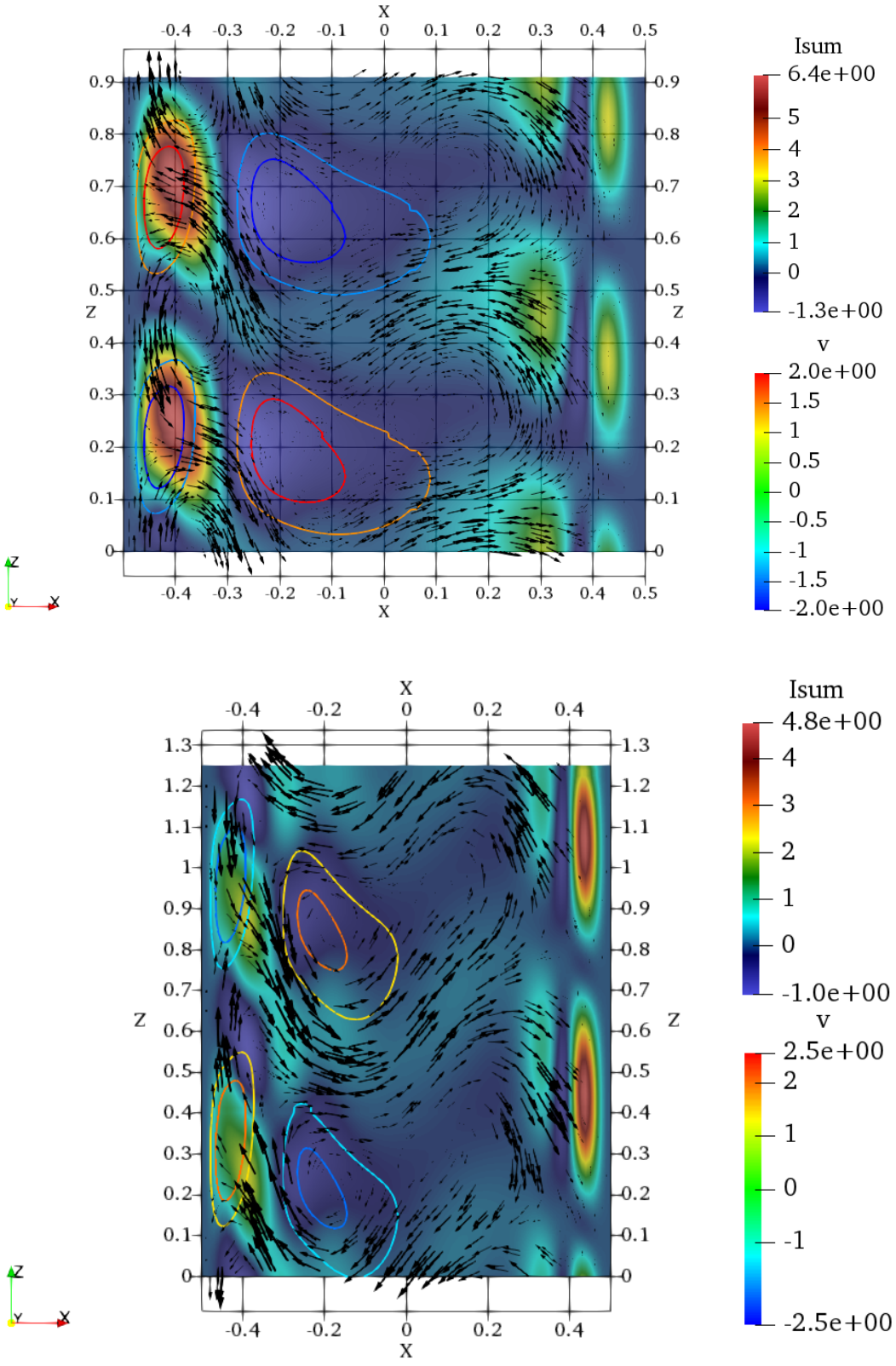


FIGURE 4.13: Energy productions of the critical modes for $\alpha = 22.5°$ (top) and $\alpha = 30.0°$ (bottom) for $y = 0.0$. The arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

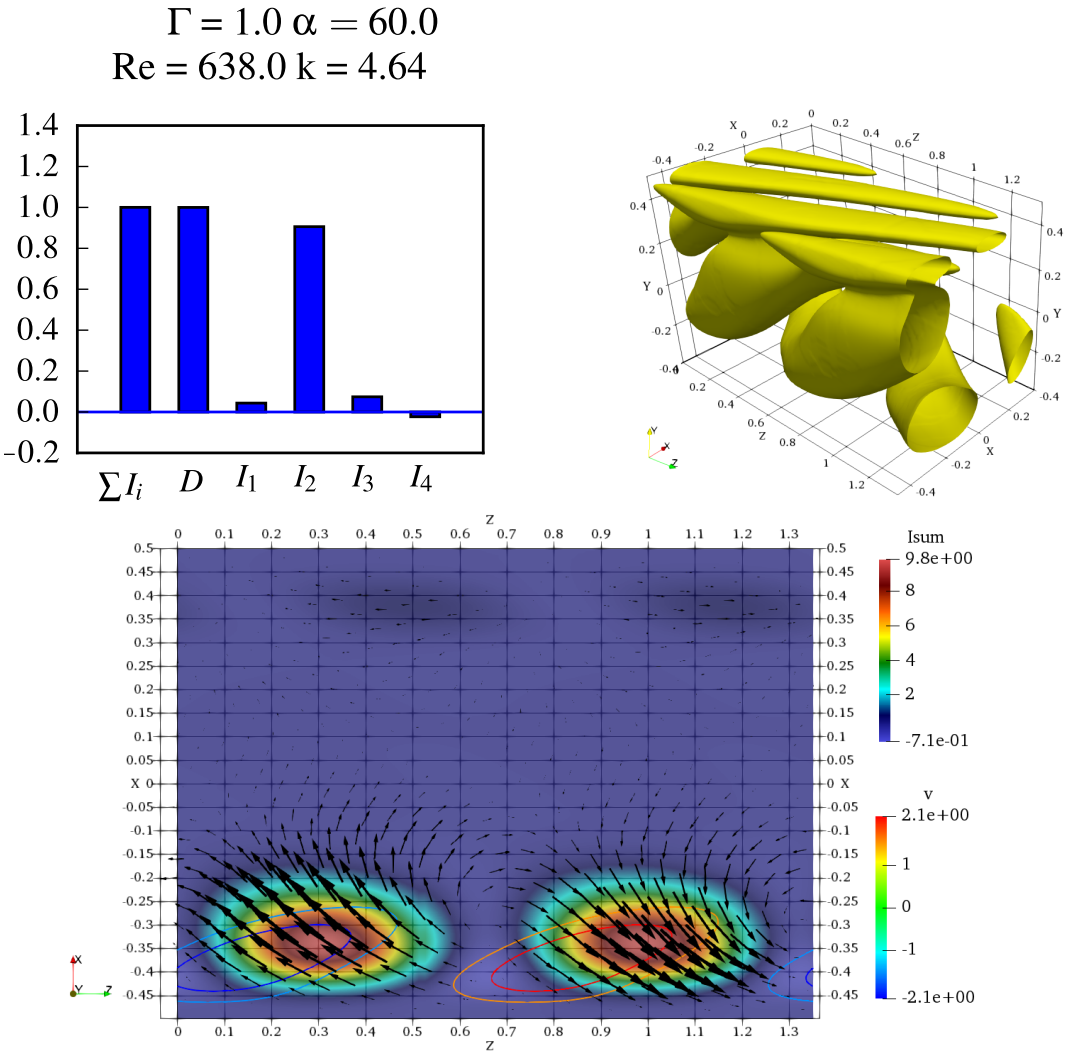


FIGURE 4.14: Energy production rate of the critical mode for $\alpha = 60°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.0$ (bottom). The arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

After the detailed analysis of the stability for $\Gamma = 1$, we now turn to the parameter dependence of the stability boundary by a variation of Γ.

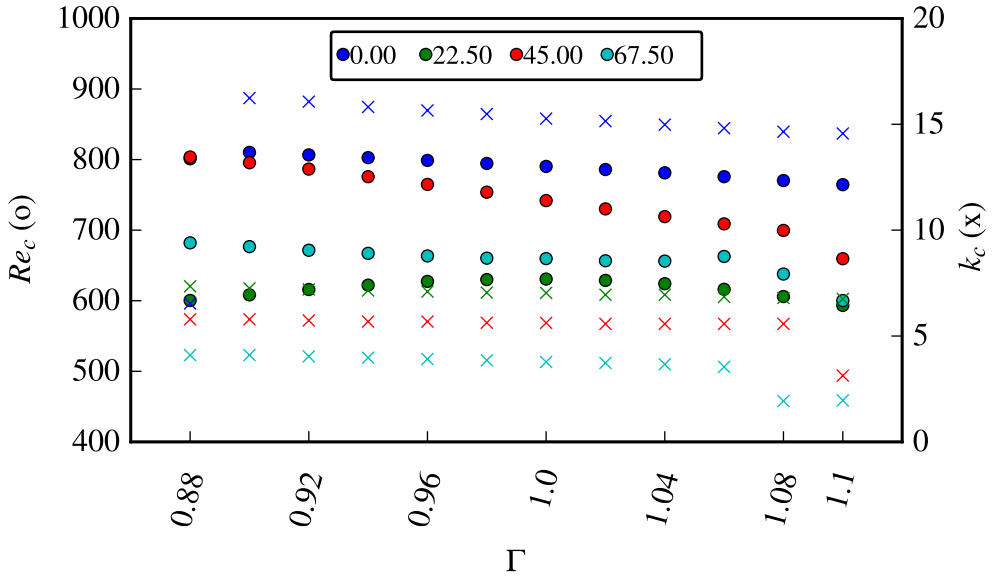


FIGURE 4.15: Variation of Γ. The critical Reynolds numbers are depicted by circles and the critical k values are plotted as crosses. For the cavity $\alpha = 0$ and $\Gamma = 0.88$ we see an interesting behaviour, as a new critical mode starts to arise.

4.3 Variation of Γ between 0.88 and 1.11

Before we start to investigate the criticality in cavities with a completely different aspect ratio, we will vary Γ only in a small range around $\Gamma = 1$, to investigate the effect of a change in geometry. For this purpose, we consider the same angles as Theofilis, Duck, and Owen (2004), because a denser sampling of α would drastically prolong the computation time. The result of the calculation is shown in Figure 4.15, where we see, that Re_c has an interesting behaviour in the Γ, α)-plane: For $\alpha = 0°$, $\alpha = 45°$ and $\alpha = 67.5°$, Re_c decreases for larger aspect ratios and increases for smaller values (with the exception of $\alpha = 67.5$ and $\Gamma = 1.06$). For the case of a drive angle $\alpha = 22.5°$, a maximum of Re_c is found for $\Gamma = 1$. The exact quantities are provided in Table C.1 in Appendix C. The order of k_c stays the same over the calculated range of angles, with an interesting exception for the case of $\Gamma = 0.88$, where k_c of $\alpha = 0°$ goes down to $k_c \approx 6.5$ from a value $k_c \approx 16.2$ for $\Gamma = 0.90$. This is explained by taking a look at Figure 4.16: At an aspect ratio of $\Gamma = 0.88$, the mode at $k \approx 6.5$ starts to govern the criticality of the cavity. Our findings are in agreement with Albensoeder, Kuhlmann, and Rath (2001), where the critical mode for $\Gamma = 0.888$ was calculated to be the stationary one we obtain in our regime of aspect ratios for the angle $\alpha = 0°$. If this oscillatory mode is the critical also for the small

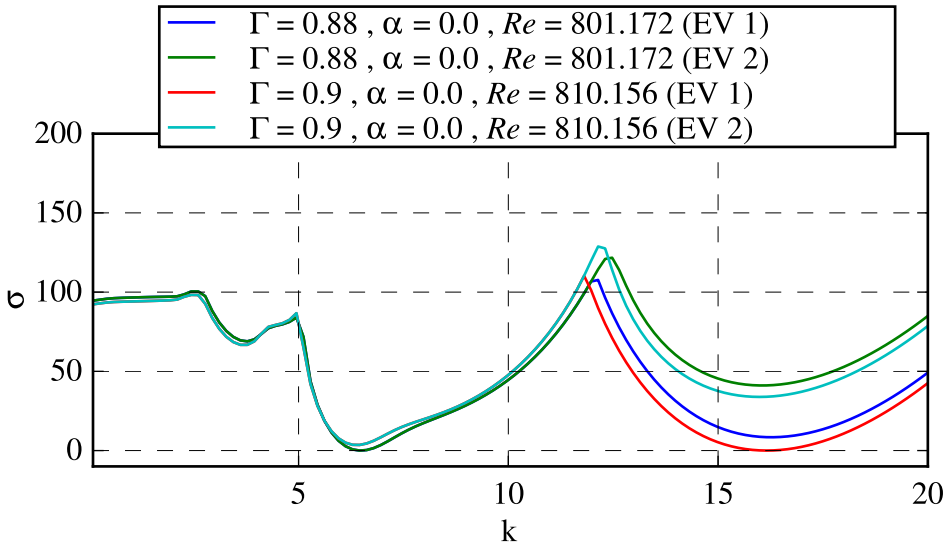


FIGURE 4.16: k - σ for $\Gamma = 0.88, \alpha = 0.0$ and $\Gamma = 0.90, \alpha = 0.0$. The first and second eigenvalue of each cavity is shown for the k-values of interest. In the lower k-regime, the first and second eigenvalues are indistinguishable.

aspect ratio cavities is an interesting question, which we will answer in the next section.

4.4 $\Gamma = 0.5$

The cavity with $\Gamma = 0.5$ is of special interest, because the literature shows strong deviations in the arising critical modes: Theofilis, Duck, and Owen (2004) obtained values for 4 neutral modes for the $\alpha = 0$ case, where the lowest Re_n was calculated to be 1467 for a mode with $k_c = 13.33$, which is in big contrast with the calculations of Albensoeder, Kuhlmann, and Rath (2001), who obtained the value $Re_c = 706.1$ for $k_c = 10.63$. Our first approach did not find the critical mode of Albensoeder, Kuhlmann, and Rath (2001) because our variation steps of Re were too large and we skipped the region of interest, so without the knowledge about the existence of the mode, we would have missed it. This finding creates the need for a very proper search of the Reynolds numbers in order to find the lowest neutral mode.
Taking a look at Figure 4.17, we see that we can reproduce the published results of Albensoeder, Kuhlmann, and Rath (2001). We obtain the criticality for $k_c = 10.6$ and $Re_c = 712$, which is a transient mode with $\omega = 819$. Thus, the critical mode from the previous section is not responsible for critical one for the aspect ratio of $\Gamma = 0.5$. An analysis of the aforementioned mode at $k \approx 6.5$, which was important in the $\Gamma = 0.88$ cavity, yields an eigenvalue of $\sigma = 97$, far away from being critical. As we have a lot of interesting modes arising for this new geometry, we will not focus on the presence of the $k \approx 6.5$ mode, even though it would be interesting, where the transition from this mode toward ours for $\Gamma = 0.5$ occurs. A comparison between Figure 4.17 and Figure 4.4 reveals that the behaviour of the critical Reynolds numbers and wavenumbers with increasing angles shows similarities with the results for $\Gamma = 1$, as k_c and Re_c lower, if we go from $\alpha = 0°$ to $\alpha = 22.5°$. The jump of Re_c between $\alpha = 22.5°$ and $\alpha = 30°$ is also present, whereas the rest of the plot of $Re_c(\alpha)$ does not resemble the $\Gamma = 1$ case: First, there is no drop in Re_c, when the drive angle grows from $\alpha = 30°$ to $\alpha = 60°$ and the small enhancement of Re_c at $\alpha = 67.5°$ for $\Gamma = 1$ is more pronounced for the geometry with $\Gamma = 0.5$. As there are three regions of interest, we will analyze one representative of each:
For the analysis we choose the angles $\alpha = 0°$, $\alpha = 30°$ and $\alpha = 67.5°$. Comparing the Figures 4.18 , 4.19 and 4.20, we learn, that the bigger the angle, the more production rate shifts towards I_2 and the energy production is more localized. While the lid-driven cavity flow for an angle of 0° has a contribution I_4, which accounts for approximately 20% of the production rate, this value decreases for higher angles, such that I_3 has a bigger contribution for the $\alpha = 30°$ flow. The influence of the shear layer on the right side of the cavity ($x \approx 0.5$, downstream)

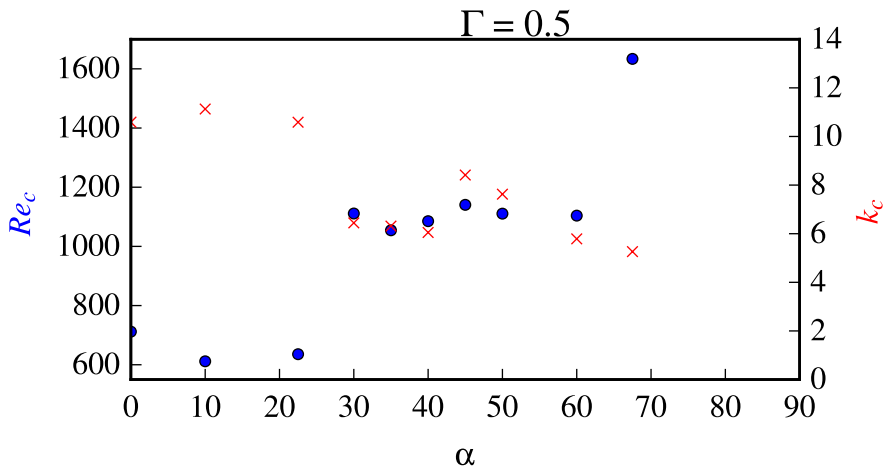


FIGURE 4.17: $\Gamma = 0.5$ – Re_c and k_c versus α. The calculated results for the cavities with the given aspect ratio show higher critical Reynolds numbers than the standard cavity with $\Gamma = 1$.

is diminished and the vortices in the $x - y$-plane, which are responsible for the energy production mechanism discussed above, vanish for the angle $\alpha = 67.5°$. The loss of this energy production results in a much higher Re_c for the large angle. This completes the treatment of the $\Gamma = 0.5$ lid-driven cavity flows and we focus our attention to geometries with bigger aspect ratios and investigate on the stability behaviour also for geometries with $\Gamma > 1$.

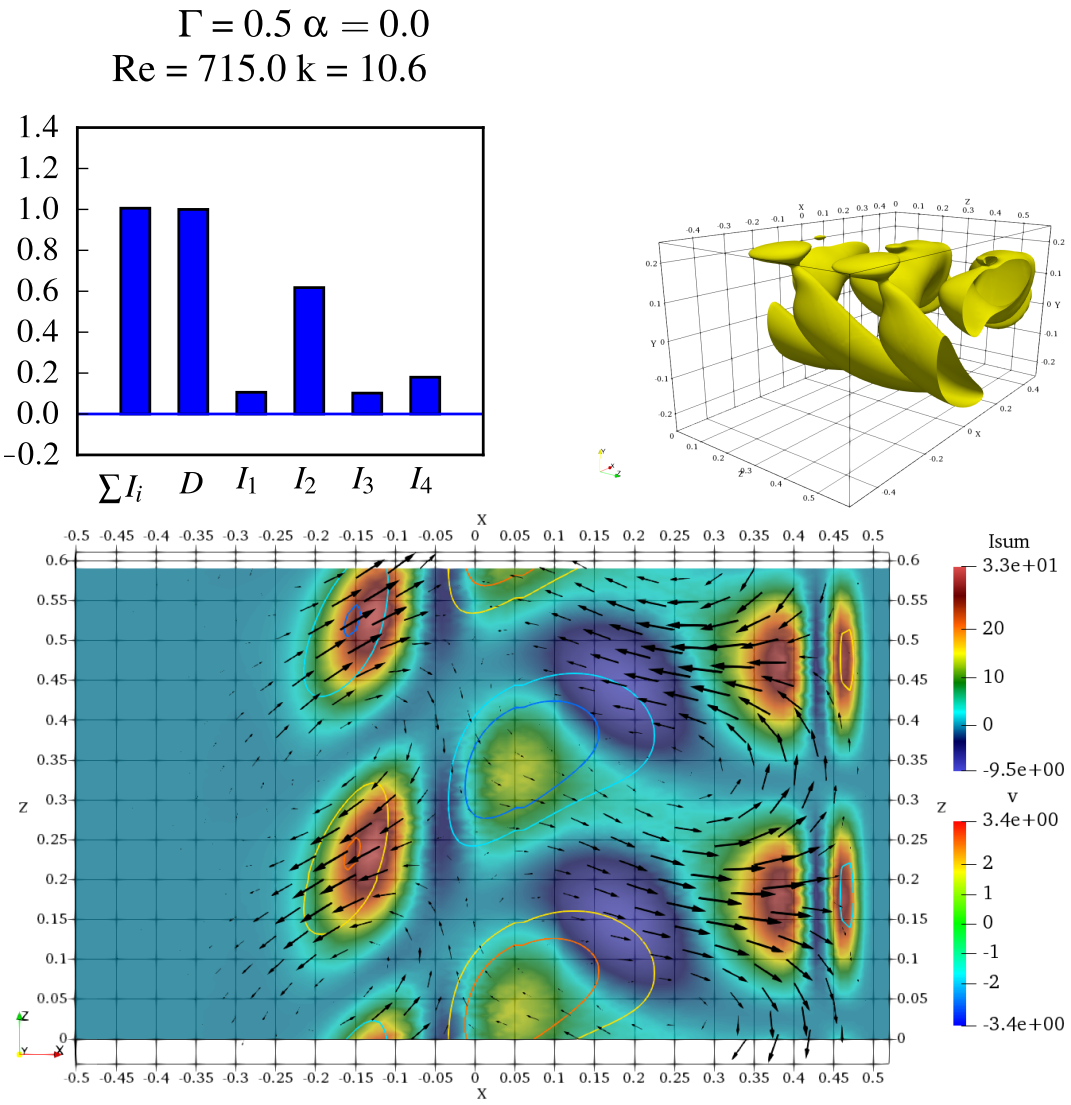


FIGURE 4.18: Energy production rate of the critical mode for $\Gamma = 0.5$ and $\alpha = 0°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.0$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

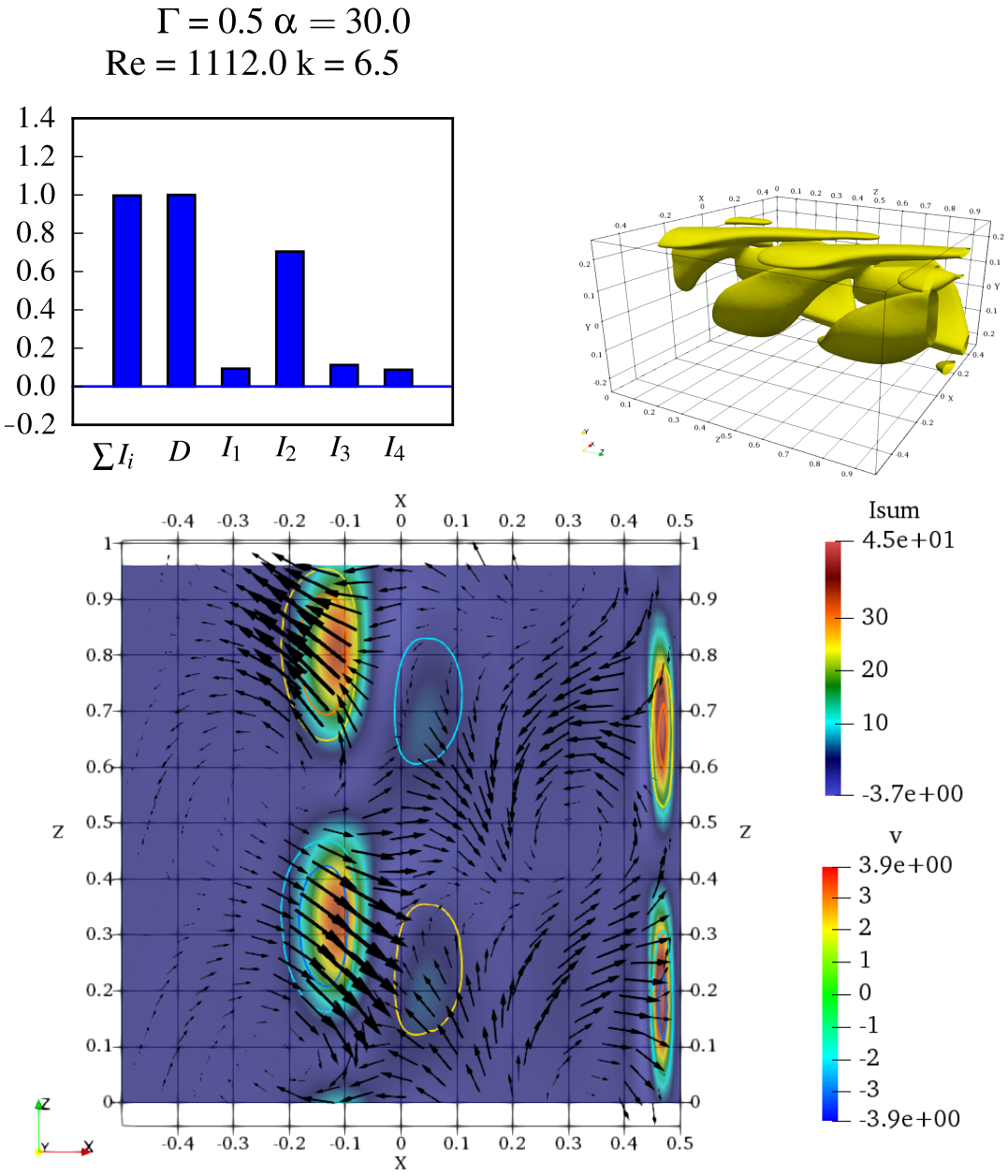


FIGURE 4.19: Energy production rate of the critical mode for $\Gamma = 0.5$ and $\alpha = 30°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.0$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

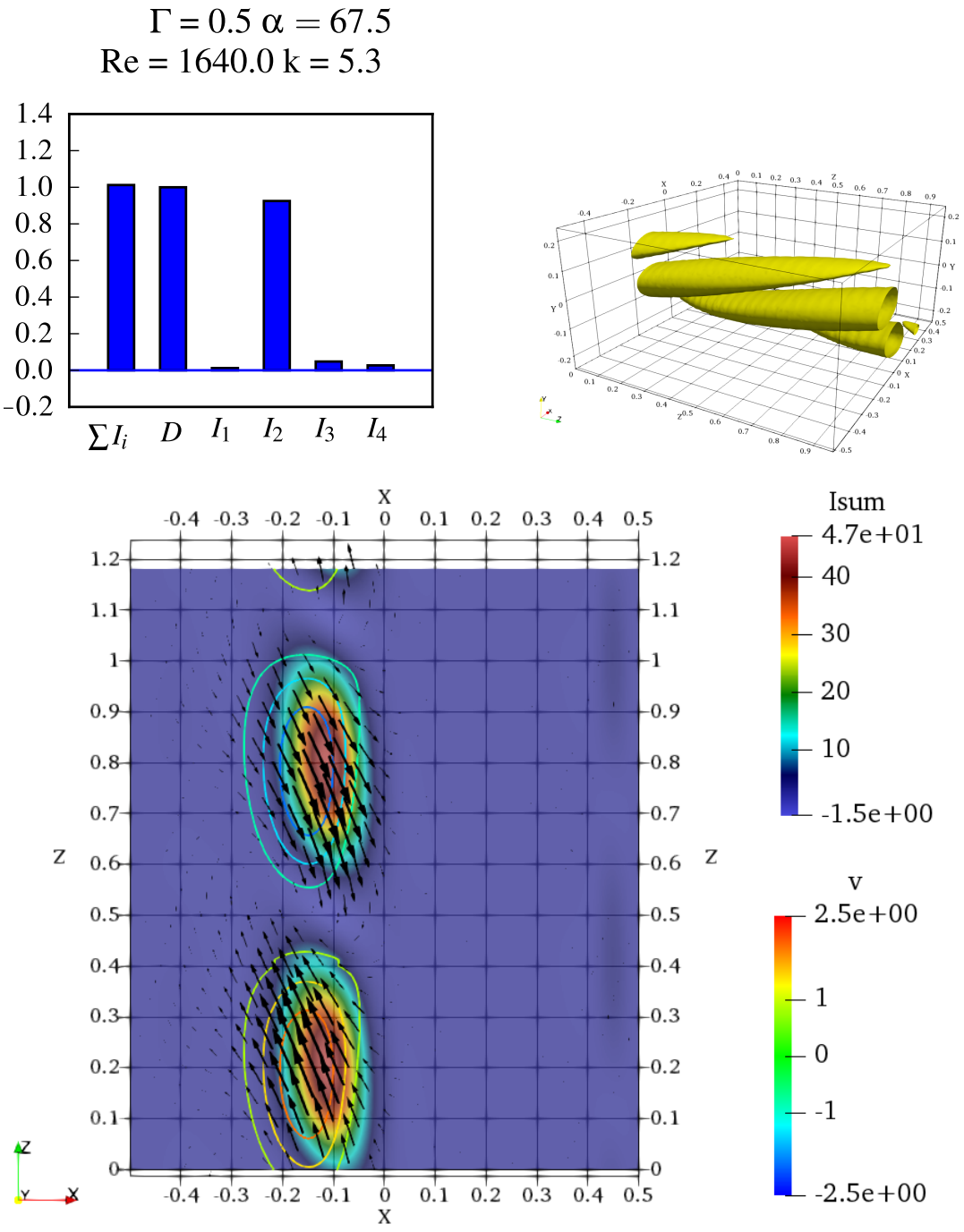


FIGURE 4.20: Energy production rate of the critical mode for $\Gamma = 0.5$ and $\alpha = 67.5°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.0$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

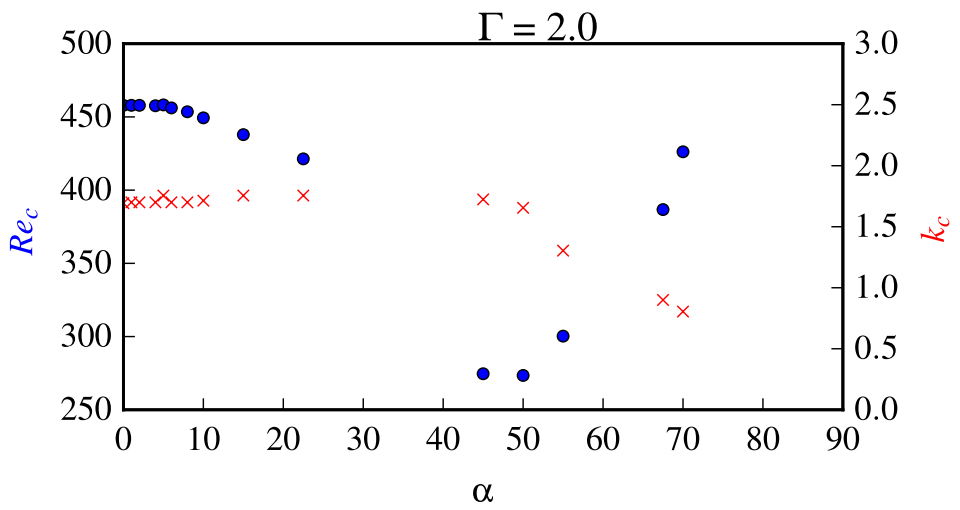


FIGURE 4.21: $\Gamma = 2.0$ – Re_c and k_c versus α. The blue dots denote the critical Reynolds numbers (scale on the left) and the red crosses the critical wavenumbers (scale on the right).

4.5 $\Gamma = 2$

We start the analysis for the lid-driven cavity flow with $\Gamma = 2$ by looking at Figure 4.21: First, we notice that the critical Reynolds numbers are lower than for the quadratic cavity with $\Gamma = 1$. In contrast to the $\Gamma = 0.5$ aspect ratio case, the critical Reynolds numbers decrease, if the drive angles are increased as long as $\alpha < 55°$. For the largest drive angles analyzed, the critical Reynolds numbers increase, as has been the case for all the aspect ratios we have analyzed up to now. The calculated values for $\alpha = 0$ ($Re_c = 458\,, k_c = 1.7$) are in good agreement with Albensoeder, Kuhlmann, and Rath (2001) [$Re_c = 446.3 \pm 10\,, k_c = 1.71$]. Again, we take three candidates for a graphical representation of the flow and an analysis of the corresponding energy production mechanism. We consider the angles $\alpha = 0°$, $\alpha = 45°$ and $\alpha = 67.5°$.

For the angle $\alpha = 0$, the analysis is presented in Figure 4.22: The main energy production contribution stems again from I_2 and the mechanism for energy gain is governed by the existence of two counter-rotating vortices, where the velocity field of the perturbation has the biggest magnitude in the intermediate region. When the fast perturbation reaches the walls, where the shear layer causes a big change of v_0 with respect to x, a negative orthogonal gradient and the parallel $\tilde{v}$ are responsible for a large energy production due to I_2. This mechanism is the same as in the cavity flow for $\Gamma = 1$ and we see that the lower half of the

cavity flow does not give a significant contribution to the energy budget.

The cavity flow for $\alpha = 45°$ is summarized in Figure 4.23: It has a much lower $Re_c = 275$ and the vortex structure at $y = 0.5$ is removed. The negative energy production at the left (upstream) side of the cavity has diminished in comparison with the flow for the inclination angle $\alpha = 0°$. The positive and negative extrema of the energy production rates exhibit little spacial separation. The energy production is again strongly correlated with the velocity contribution $\tilde{v}$, orthogonal to the shown plane. The energy contributions I_1, I_3 and I_4 for the critical mode differ strongly from the $\alpha = 0°$ critical flow, since most of the weight of I_4 shifts to I_1 and I_3. In contrast to the $\Gamma = 0.5$ cavity flows, we do not find the strong decrease of the energy production on the right side (downstream), when the drive angle is increased. This difference in the behaviour of critical flows with increasing angles is manifested when looking at the critical cavity flow for the drive angle $\alpha = 67.5°$, whose properties are presented in Figure 4.24: We observe, that again the upper vortex (closest to the lid) is mainly responsible for the energy production and that the contribution I_2 is dominating for large angles, as was the case for all the cavity flows analyzed up to now. The slice at $y = 0.5$ reveals another property of the large angle cavity flows for $\Gamma = 2$: While the spots with a positive energy production on the slice located at the vertical center of the upper basic flow vortex have approximately the same size for all the cavity flows, the negative production rate emerges over an increased area, which may explain the higher Re_c for the large angle flows.

We saw in this section, that the flows for cavities with an aspect ratio $\Gamma = 2$ have a much lower Re_c than the smaller aspect ratio flows and that the energy production takes place mainly in the upper half of the cavity. The dependence on the angle turns out to be similar to the $\Gamma = 1$ cavity flows, with the difference that the rise of Re_c for angles $\alpha < 5°$ is less pronounced and the lowering of Re_c for the angle $\alpha = 45°$ is 1/3 of $Re_c(\alpha = 0°)$. The large angle regime behaviour differs, as there is a continuous rise in Re_c as the angle is increased. The curves for k_c are similar in the high angle regime, where we obtained a decrease of k_c with increasing α for all the aspect ratios.

We continue our analysis by further increasing the aspect ratio and since Figure 4.25 reveals, that the behaviour for $\Gamma = 2.5$ resembles the curve of $\Gamma = 2$, we will finish our investigations with the cavity flow for an aspect ratio $\Gamma = 3$.

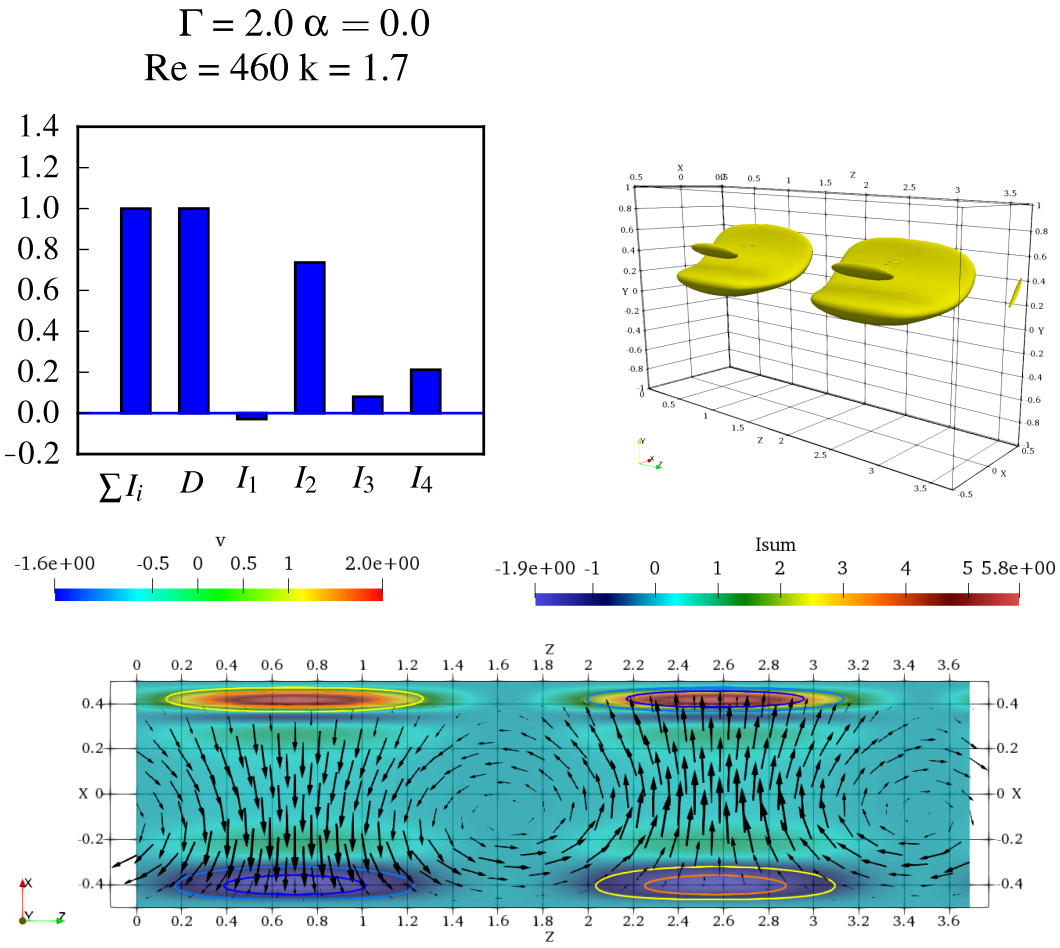


FIGURE 4.22: Energy production rate of the critical mode for $\Gamma = 2.0$ and $\alpha = 0°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.5$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.5$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

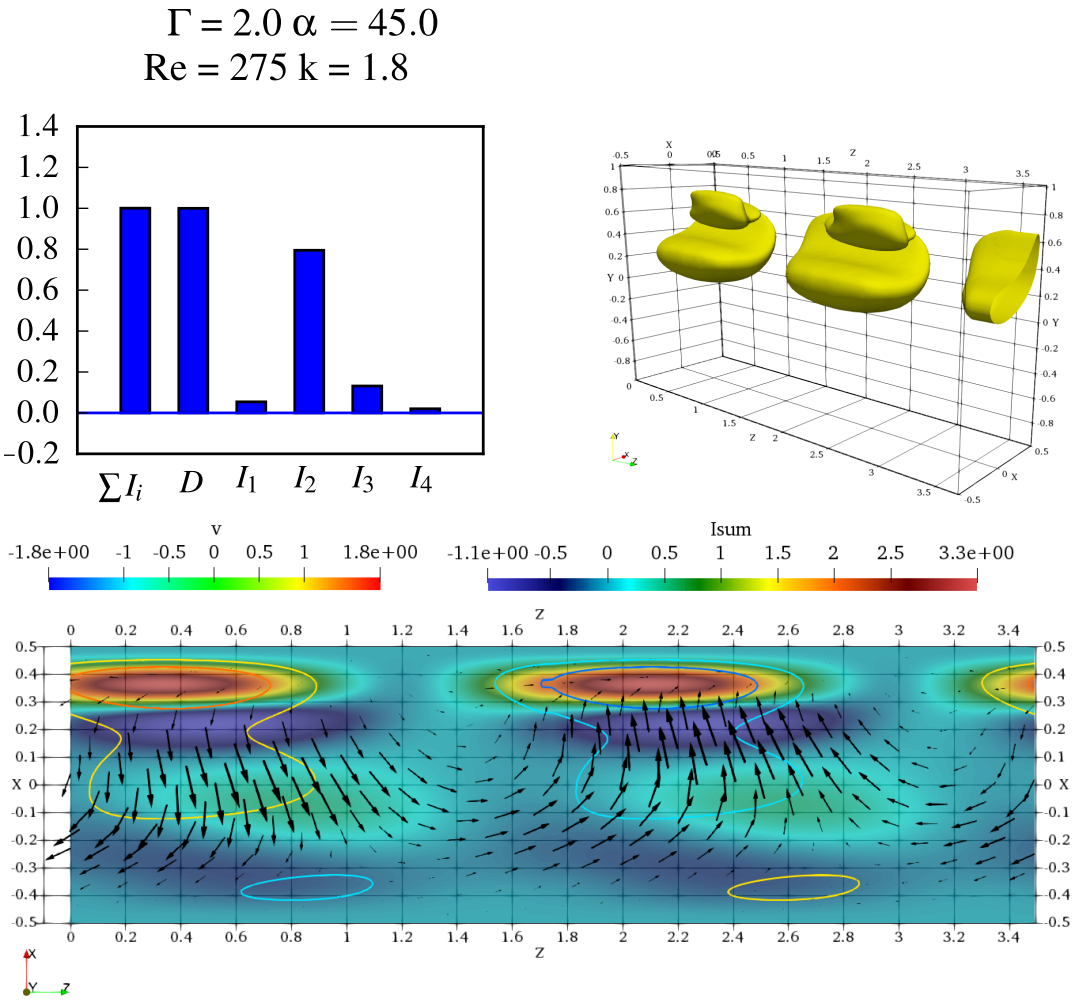


FIGURE 4.23: Energy production rate of the critical mode for $\Gamma = 2.0$ and $\alpha = 45°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.5$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.5$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

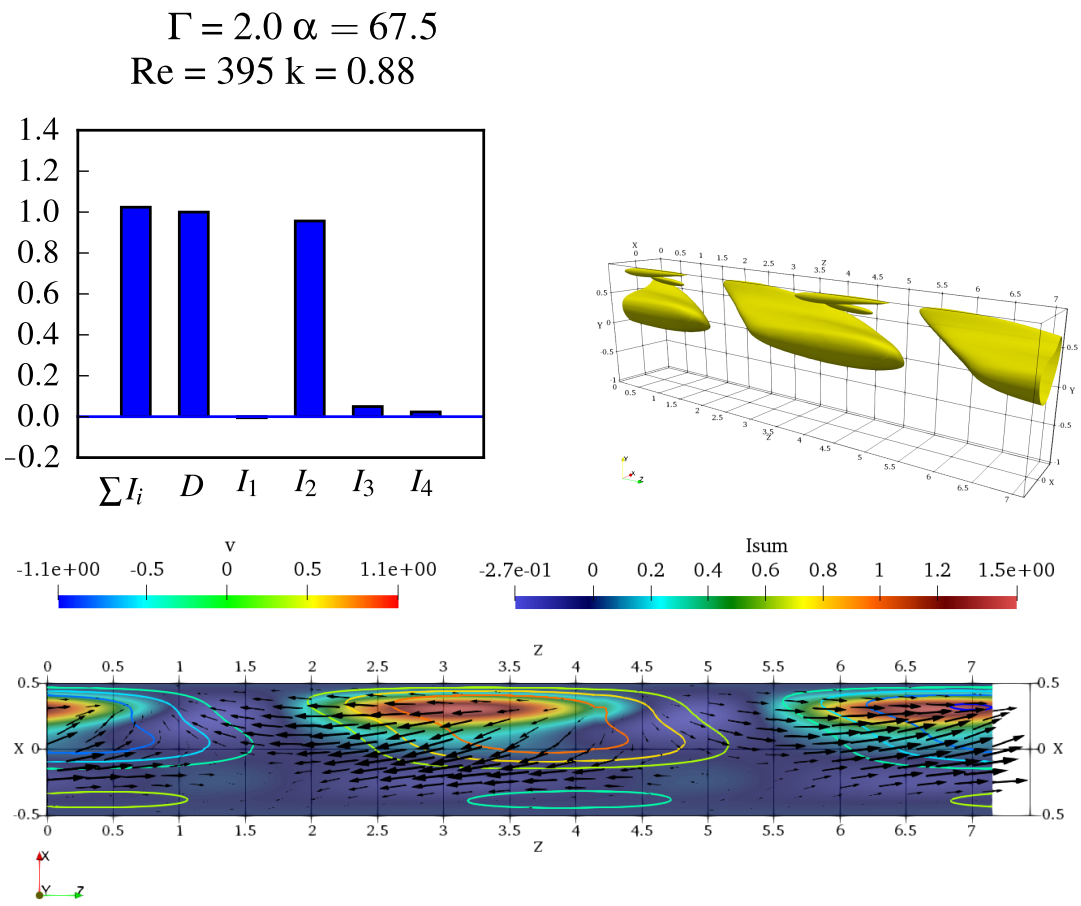


FIGURE 4.24: Energy production rate of the critical mode for $\Gamma = 2.0$ and $\alpha = 67.5°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.5$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.5$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

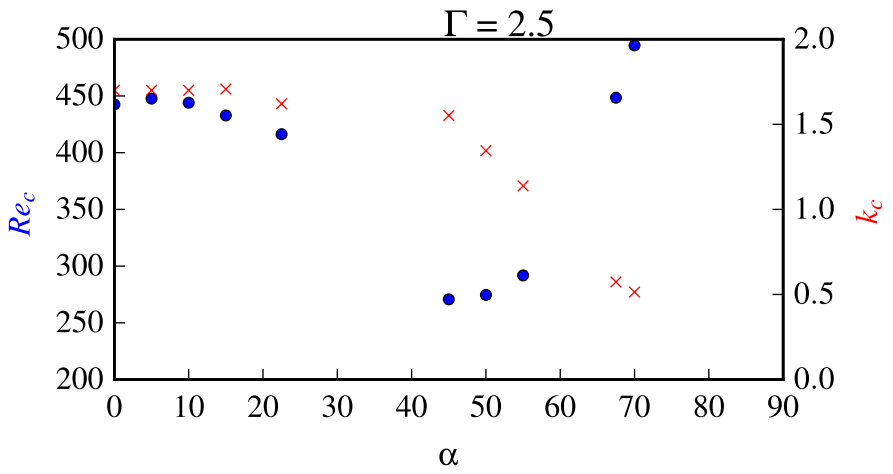


FIGURE 4.25: Γ = 2.5 – Re_c and k_c versus α. The blue dots denote the critical Reynolds numbers (scale on the left) and the red crosses the critical wavenumbers (scale on the right).

4.6 Γ = 3

For the aspect ratio Γ = 3, we obtain a higher $Re_c = 442$ for $\alpha = 0$ than Albensoeder, Kuhlmann, and Rath (2001) ($Re_c = 424.869$) but the trend of decreasing critical Reynolds numbers with increased aspect ratios is continued in accordance with their study. The results differ by 4%, which is not a dramatic difference. This critical perturbation flow is shown in Figure 4.26, where we see that I_2 is again the biggest energy transfer term and that the energy production is located at the upper basic flow vortex closest to the moving lid. The presented slice at $y = 0.75$ on the bottom of Figure 4.26 reveals that the vortex structure of the perturbation in this high energy production surface looks different from the one obtained for the Γ = 2 case in Figure 4.22 as the vortices in the center have vanished. However, this may also be owed to the particular choice of the projection slice at $y = 0.75$.

The influence of an increasing drive angle on the critical Reynolds numbers for Γ = 3 cavities was investigated for six angles. The results resemble the behaviour of the Γ = 2 and Γ = 2.5 cavities, with an interesting difference of the critical Reynolds numbers for large angles: As we see in Figure 4.27, there is a very big change in Re_c going from $\alpha = 50°$ to $\alpha = 67.5°$ ($Re_c(\alpha = 67.5°) - Re_c(\alpha = 50°) = 666$) and a significant reduction of Re_c going from $\alpha = 67.5°$ to $\alpha = 70°$ ($Re_c(\alpha = 67.5°) - Re_c(\alpha = 70°) = 181$). A more profound analysis of the critical modes for the critical flows at these two angles is presented

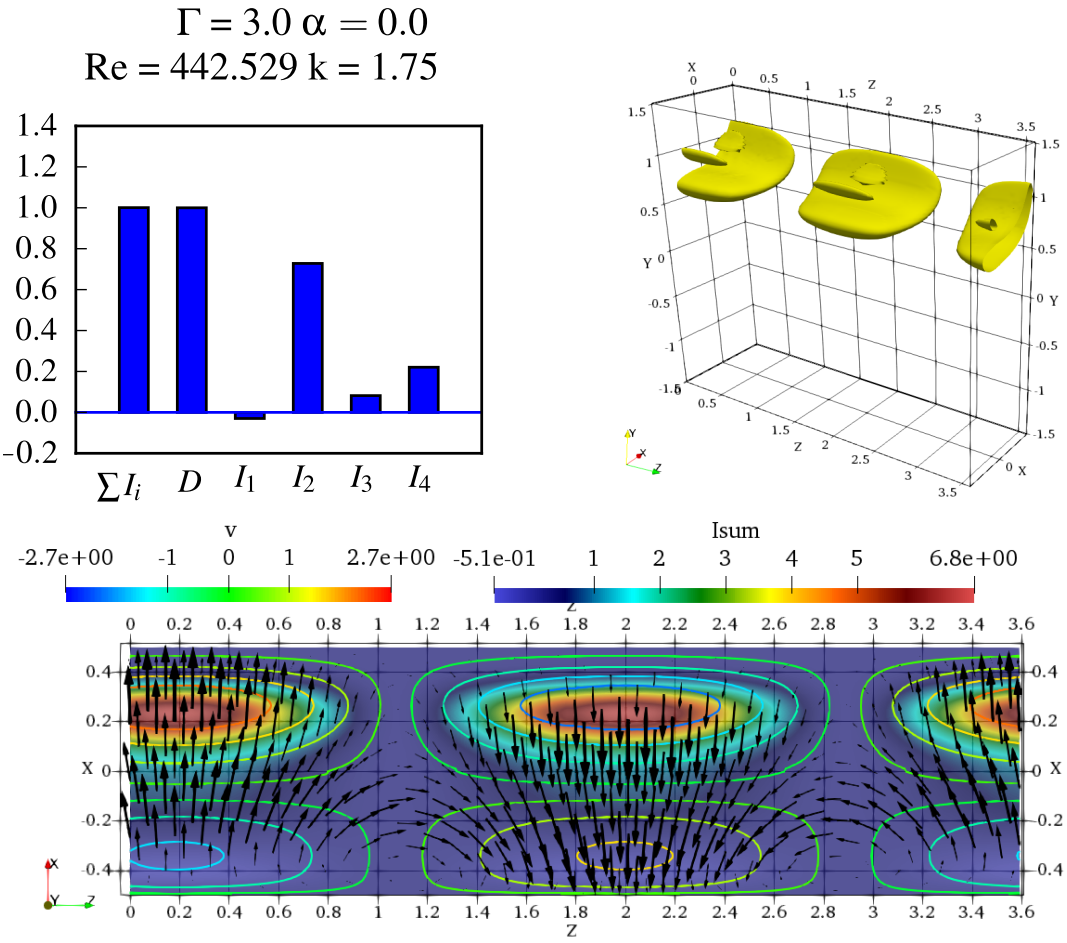


FIGURE 4.26: Energy production rate of the critical mode for $\Gamma = 3.0$ and $\alpha = 0.0°$. The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.75$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.75$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

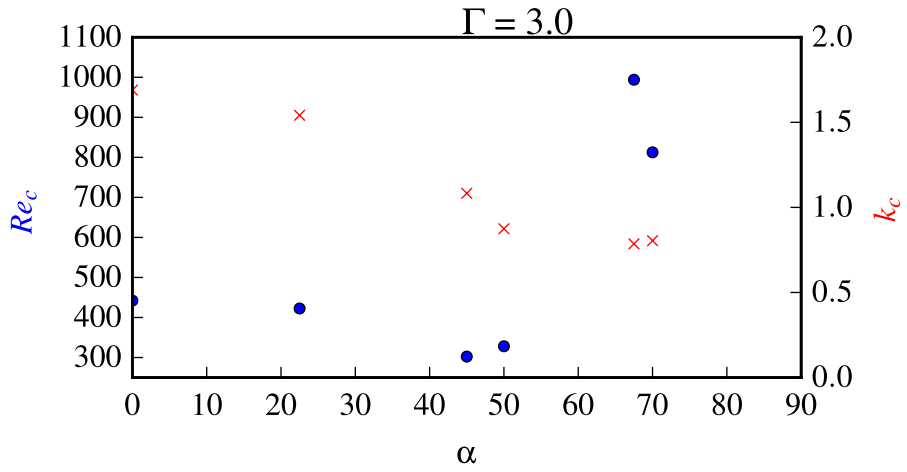


FIGURE 4.27: $\Gamma = 3.0$ - Re_c and k_c versus α. The blue dots denote the critical Reynolds numbers (scale on the left) and the red crosses the critical wavenumbers (scale on the right).

in Figure 4.28 and Figure 4.29, where we extract that they seem to differ only in Re_c, since the energy production localizations as well as the energy transfer term contributions are similar. Due to this strange behaviour, these calculations were redone with a finer mesh and different parameters for the eigenvalue solver, which did not change the result. If there is physics involved has to be clarified by a three-dimensional simulation, solving the full Navier–Stokes equations.

4.7 Overview of the results of the linear stability analysis

We finish our analysis with three-dimensional illustrations of the dependence of the critical values of all the calculated cavity flows on the aspect ratio and drive angle. The results are shown in Figure 4.30. However, it has to be mentioned that some cavity flows were not analyzed as deeply as the ones described above and we may have missed criticality for lower Reynolds numbers, if our Re sampling was not dense enough.

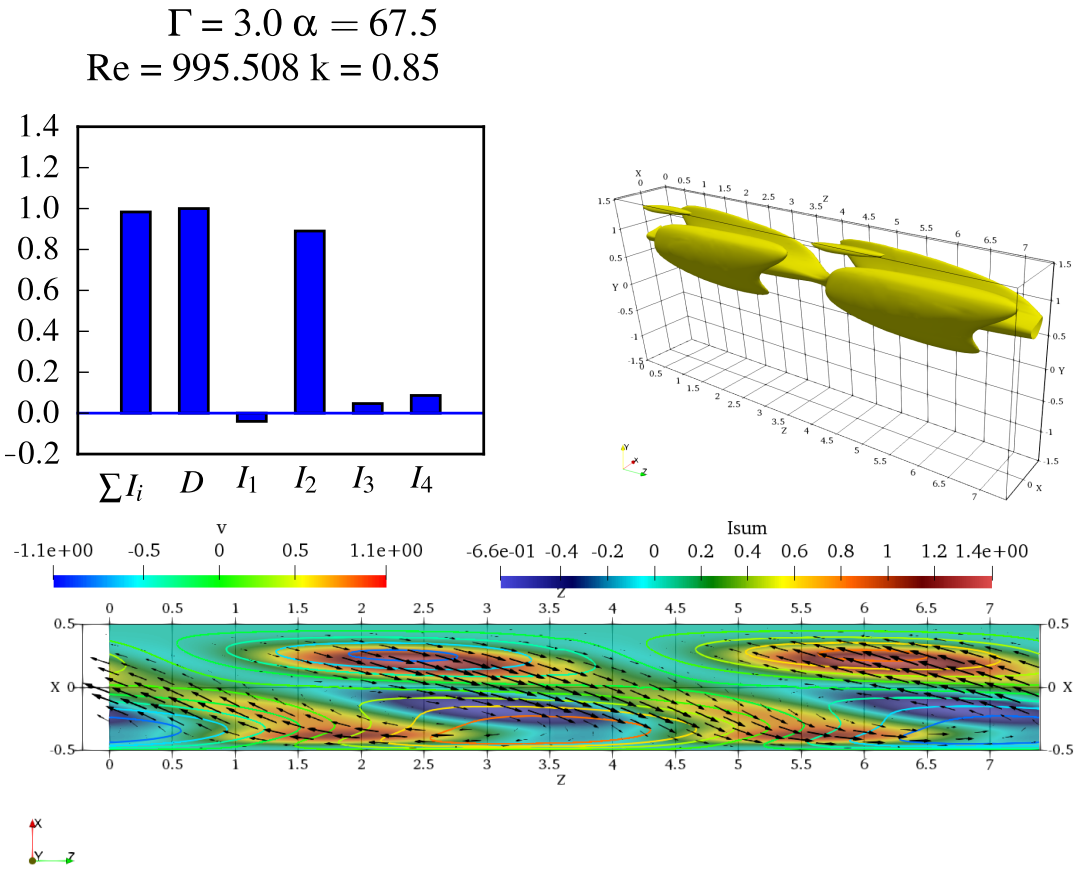


FIGURE 4.28: Energy production rate of the critical mode for $\Gamma = 3.0$ and $\alpha = 67.5°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.75$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.75$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

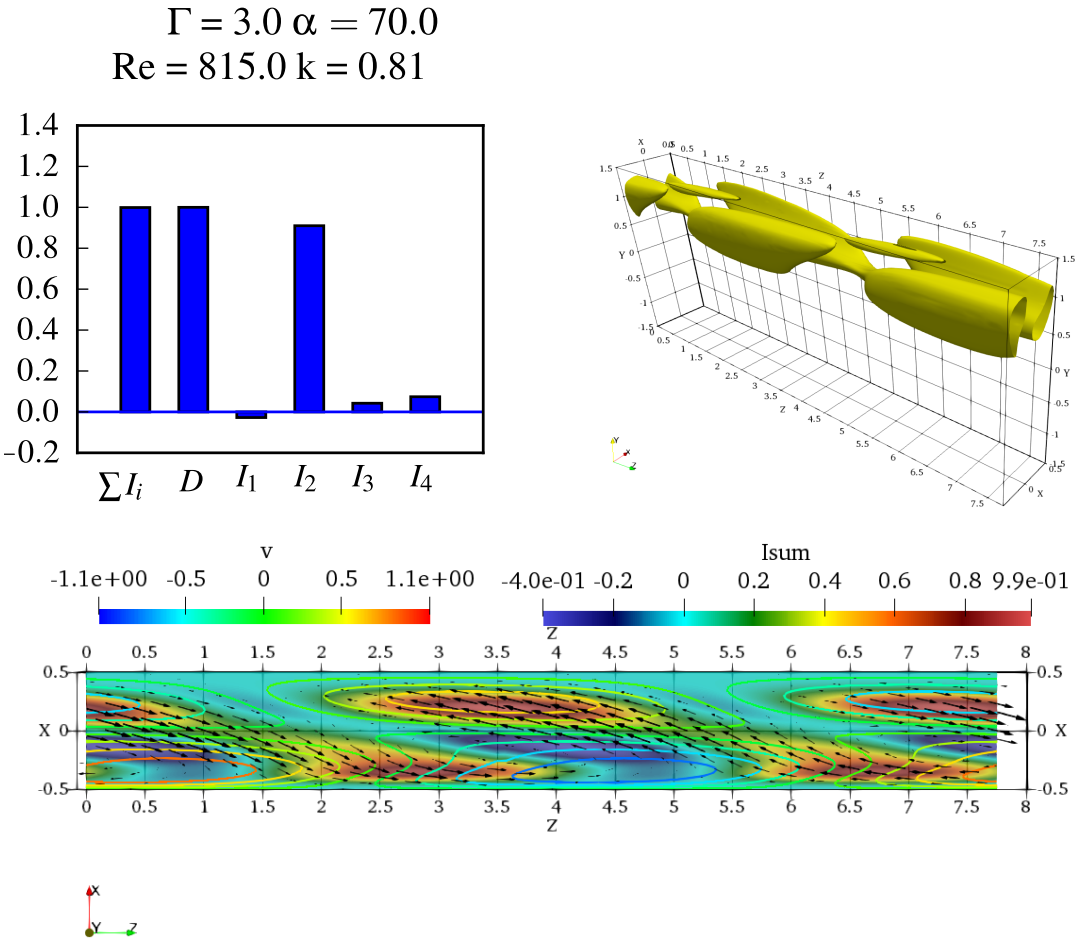


FIGURE 4.29: Energy production rate of the critical mode for $\Gamma = 3.0$ and $\alpha = 70.0°$: The contributions (top left), the sum of the production rates in three dimensions (top right) and on the slice with $y = 0.75$ (bottom). The isosurface for the 3-dimensional plot of the energy production corresponds to the value $\sum_i I_i / \max(\sum_i I_i) = 1/10$. For the slice at $y = 0.75$ the arrows denote the two-dimensional projection of the perturbation velocity. The colours represent the total local energy production and the isolines $\tilde{v}$.

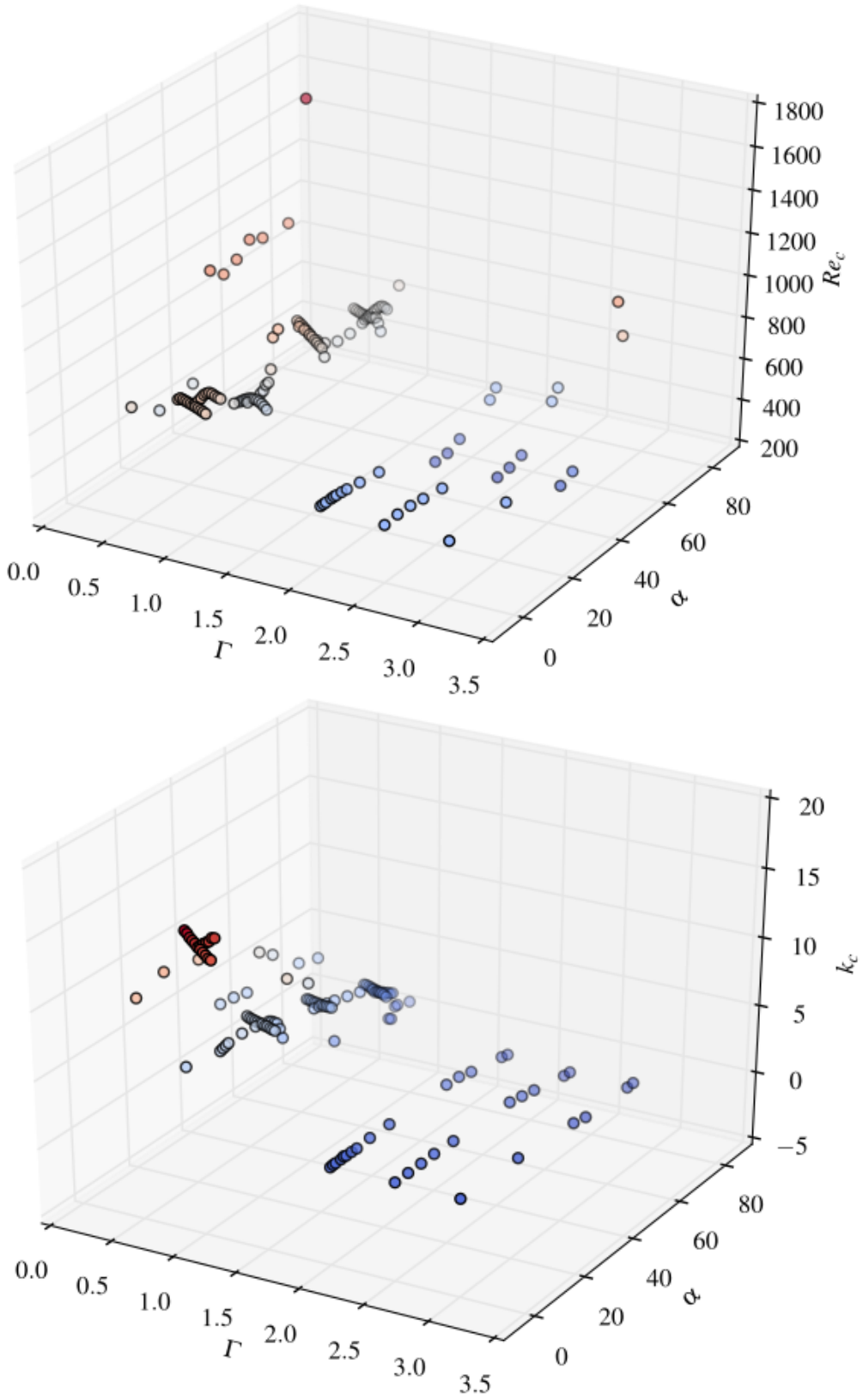


FIGURE 4.30: Results overview. The plots show a three-dimensional representation of Re_c (top) and k_c (bottom) as functions of Γ and α. The colors of the dots denote the value [blue-low , red-high] and are scaled logarithmically for the upper and linearly for the lower plot.

Chapter 5

Summary and Outlook

In this thesis, we introduced the lid-driven cavity problem as an important benchmark system for stability analysis. We presented the mathematical description of the linear stability analysis and the numerical implementation thereof. Subsequently, we developed a code, which is able to accurately determine the properties of the lid driven cavity flows and capable to deal with a change of the geometrical parameter Γ and the drive angle α of the moving lid. We showed, that the output of the calculations is comparable with published results for $\alpha = 0°$ and for the discrepancy with Theofilis, Duck, and Owen (2004) for $\Gamma = 1$ and $\alpha = 22.5°$, we verified our results with a three-dimensional simulation. Thus, we are therefore confident, that the prediction of new critical modes is correct. Whether we always found the lowest Reynolds number, where the transition towards instability occurs can not be guaranteed, because there appeared delicate situations, where our code overlooked modes. This happened, if the criticality of a mode only occurs in a small Re regime, where the search for critical modes by a subsequent bisectioning of the Re interval misses modes, if the guesses for Re_c span a big interval.

We found the critical modes, reported in literature and predicted new modes for an expanded parameter space in Γ and α. For a prediction of the lowest Reynolds numbers, a very profound search was performed to assure that indeed the lowest neutral Reynolds numbers were found. In addition, we were able to analyze the modes of interest with respect to their energy production, both in integral form as well as in local form, where the critical parameters of the flow could be determined and the mechanism that drives the instability could be described. The most prominent mechanism leading to a positive energy production was explained by the effect of a shear layer. For angles $\alpha < 45°$, counter rotating vortices develop and the intermediate region between the vortices turned out to be the main source for a positive energy production. When the angle was further increased, the vortex structure in the $x - z$ plane was lifted, but the big gradients of the velocities in the shear layer still were the

main sources of positive energy production. The most prominent transfer term was I_2 for all the analyzed cavity flows.

A further investigation of other neutral modes in the higher Re regime would be of great interest and the change of the critical mode upon a variation of the governing parameters to another might as well be calculated. It would be interesting to see, whether the predicted instabilities can be observed experimentally and in three-dimensional simulations, where the full Navier–Stokes equations are solved. This work provides a search direction for critical modes in such simulations, as guesses for the onset of instability were calculated with the linear stability analysis. This has already enabled us to find new modes, occurring for lower Reynolds numbers as reported by Theofilis, Duck, and Owen (2004).

Appendix A

Derivation of Reynold's transport theorem [1]

In this Appendix, Reynold's transport theorem from chapter 2 is derived. As shown there, it is needed to calculate the time derivative in the Lagrangian specification for an integral quantity. For this purpose we need to transform volume elements in the Lagrangian specification (the starting point of the particle is denoted as $\vec{X} = (X, Y, Z)$ to the Eulerian specification (the points in space are denoted as $\vec{x} = (x, y, z)$. This is achieved with the help of the determinant of the Jacobian

$$det\, J = \frac{\partial\,(x, y, z)}{\partial\,(X, Y, Z)} = \begin{vmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{vmatrix}, \tag{A.1}$$

in index notation J is given by $J_{ij} = (\partial x_i / \partial X_j)$. The transformation of the volume element from the Eulerian (dV) to the Lagrangian (dV_0) specification reads

$$dV = dV_0\, |det\, J| \tag{A.2}$$

This determinant may be calculated by the Laplacian expansion

$$det\, J = \sum_{k=1}^{3} J_{ik}\, \alpha_{ik} \quad \text{with} \quad \alpha_{ik} = (-1)^{i+k} \left(det\, \tilde{J} \right)_{ik}, \tag{A.3}$$

where $(det\, \tilde{J})_{ik}$ is the minor, which is the subdeterminant resulting by the deletion of the i-th row and the k-th column. Due to the antisymmetry of the determinant, the following relation holds

$$\sum_{k=1}^{3} J_{ik}\, \alpha_{jk} = \delta_{ij}\, det\, J, \tag{A.4}$$

[1] This derivatoin also follows the treatment given in (Braun 2001)

with δ_{ij} being the Kronecker-delta. The last needed ingredient for the time derivative of integral quantities, is the material derivative of the Jacobian determinant itself. Starting from the differential of the Laplacian expansion (A.3)

$$d\left(det\ J\right) = \sum_i \sum_j \underbrace{\frac{\partial\left(det\ J\right)}{\partial J_{ij}}}_{\alpha_{ij}} dJ_{ij} \tag{A.5}$$

the material derivative reads

$$\frac{D\left(det\ J\right)}{Dt} = \sum_i \underbrace{\sum_j \alpha_{ij}}_{\delta_{ki} det\ J} \frac{DJ_{ij}}{Dt} = \sum_i \sum_k \sum_j J_{kj}\alpha_{ij}\frac{\partial v_i}{\partial x_k} = det\ J \sum_i \underbrace{\frac{\partial v_i}{\partial x_i}}_{\vec{\nabla}\cdot\vec{v}}, \tag{A.6}$$

where the material derivative of the matrix elements was used

$$\frac{DJ_{ij}}{Dt} = \frac{D}{Dt}\left(\frac{\partial x_i}{\partial X_j}\right) = \frac{\partial v_i}{\partial X_j} = \sum_k \frac{\partial v_i}{\partial x_k}\frac{\partial x_k}{\partial X_j}. \tag{A.7}$$

Now there is enough equipment to transform the volume integrals and derive Reynold's transport theorem:

$$\frac{D}{Dt}\int_V b(\vec{x},t)\,dV = \frac{D}{Dt}\int_{V_0} b(\vec{X},t)\,\underbrace{|det\ J|\ dV_0}_{dV} \tag{A.8}$$

$$= \int_{V_0}\left(b\underbrace{\frac{D\,|det\ J|}{Dt}}_{|det\ J|\vec{\nabla}\cdot\vec{v}} + |det\ J|\,\frac{Db}{Dt}\right)dV_0 \tag{A.9}$$

$$= \int_{V_0}\left(\frac{Db}{Dt} + b\vec{\nabla}\cdot\vec{v}\right)|det\ J|\,dV_0 \tag{A.10}$$

$$= \int_{V}\left(\frac{Db}{Dt} + b\vec{\nabla}\cdot\vec{v}\right)dV \tag{A.11}$$

$$= \int_{V}\left(\frac{\partial b}{\partial t} + \vec{\nabla}(b\vec{v})\right)dV \tag{A.12}$$

q.e.d.

Appendix B

Convergence Plots

This appendix is meant to be a placeholder for the convergence plots, which did not fit in the main text for readability reasons.

B.1 $\Gamma = 1$

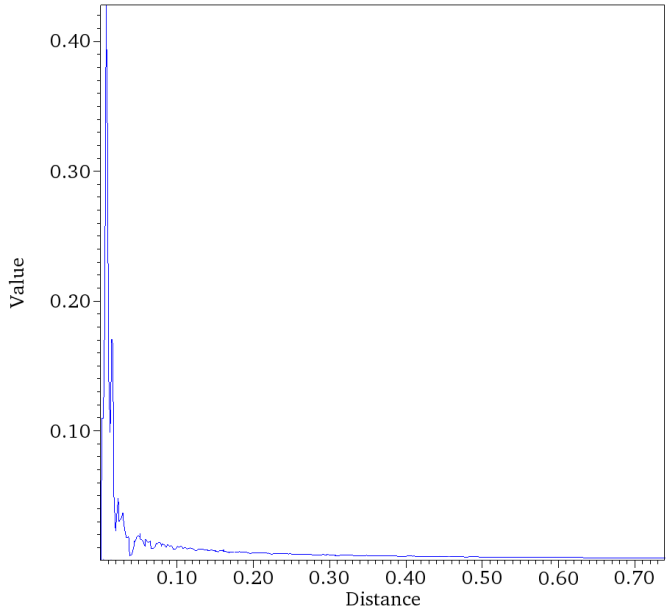


FIGURE B.1: The decay of the difference of the velocity magnitude between a 250×250 grid and a 200×200 grid from the top left edge ($x = -0.5, y = 0.5$) towards the center ($x = 0, y = 0$). The parameters of the compared data are $\Gamma = 1$ and $Re = 10$ and the values on the finer grid were interpolated on the coarser one.

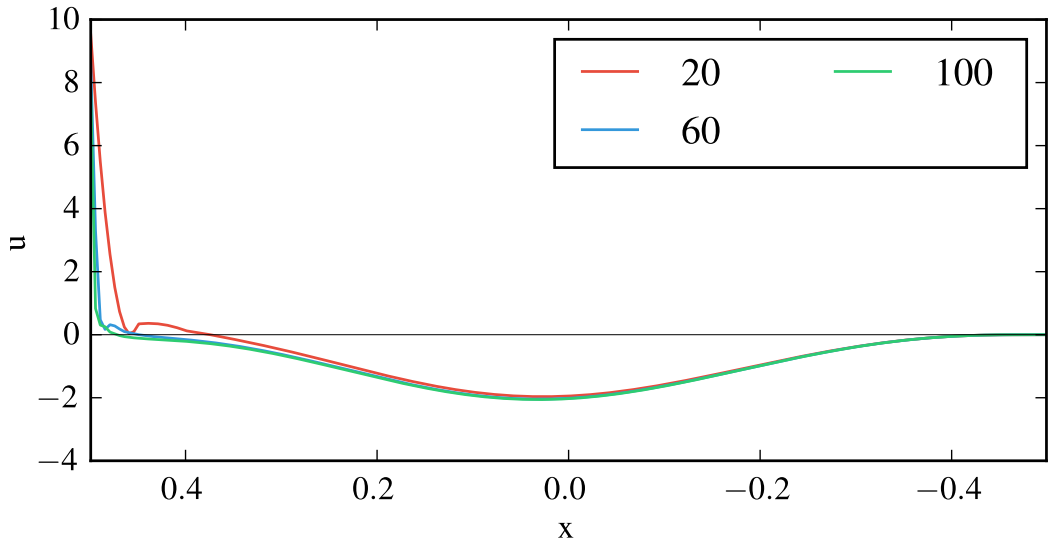


FIGURE B.2: Grid convergence along the diagonal $y = x - 0.5$ from the bottom left to the top right of the cavity.($Re = 10$, $\Gamma = 1$)

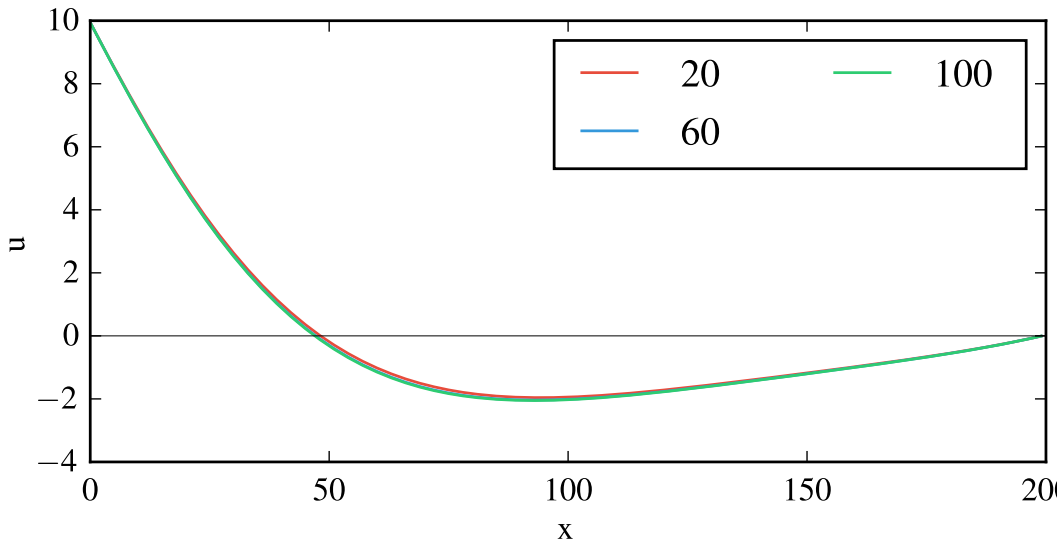


FIGURE B.3: Grid convergence along a vertical line in the center from the top lid to the bottom. ($Re = 10$, $\Gamma = 1$)

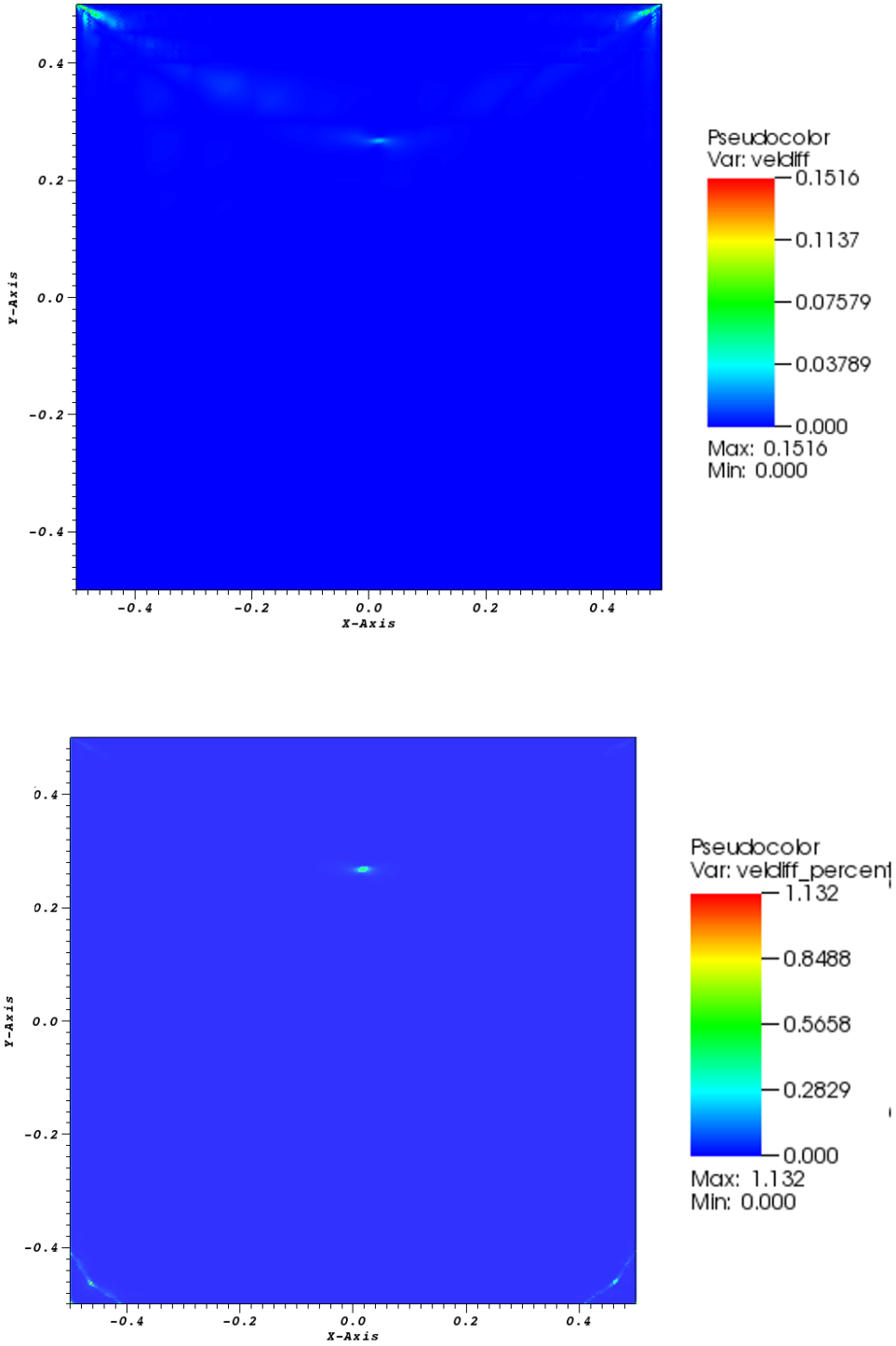


FIGURE B.4: Refined mesh comparison for the velocity field. *Top*: The difference in magnitude of the velocity between a starting mesh of 160×160 grid points and 150×150 grid points, which are refined 3 times as described in the main text.*Bottom*, The difference divided by the value of the magnitude of the velocity. We see that the maximal difference is about 1% of the value.($Re = 10$, $\Gamma = 1$)

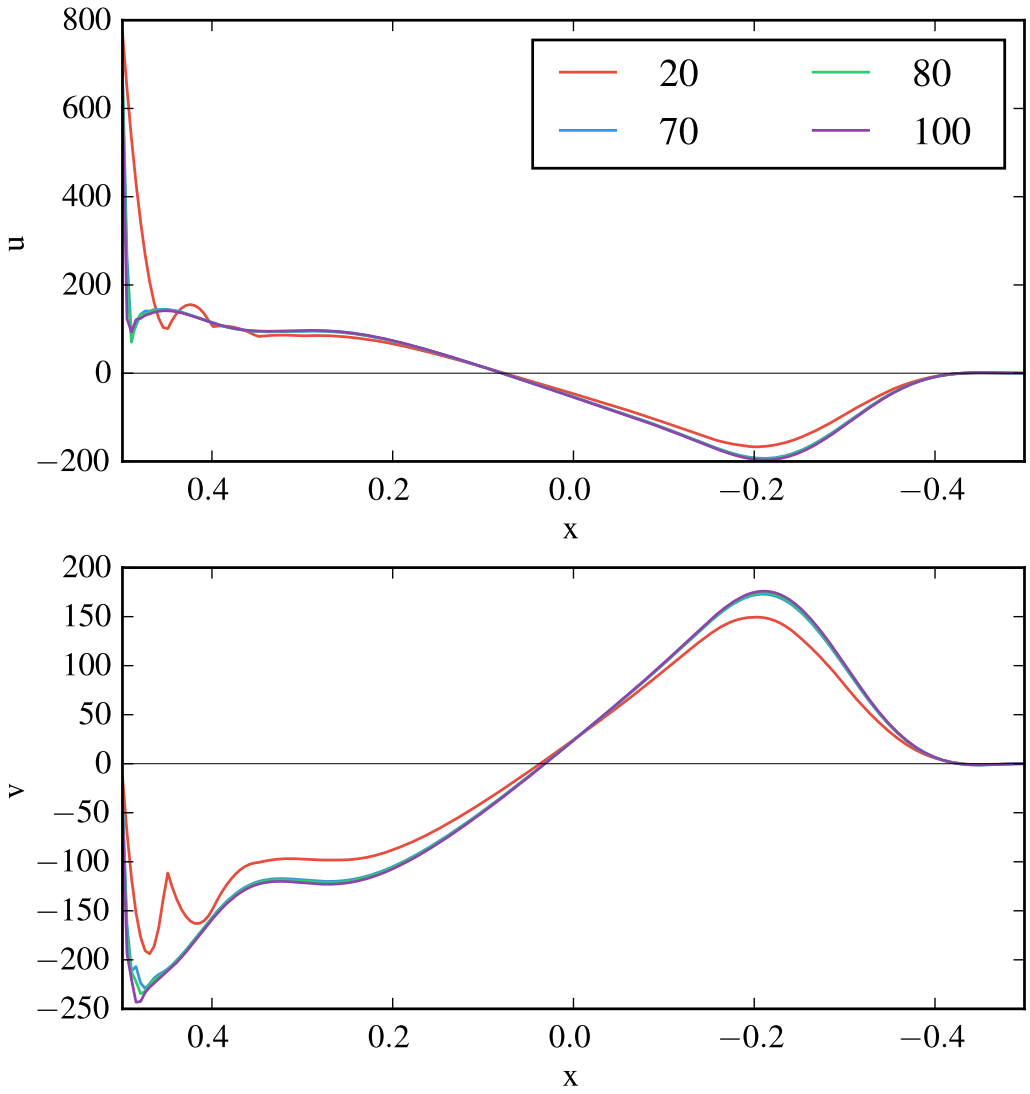


FIGURE B.5: $\Gamma = 1$, $Re = 800$, grid convergence. The two plots depict the x- and y-components of the velocity field along the line $y = x - 0.5$ without any refinement towards the boundary.

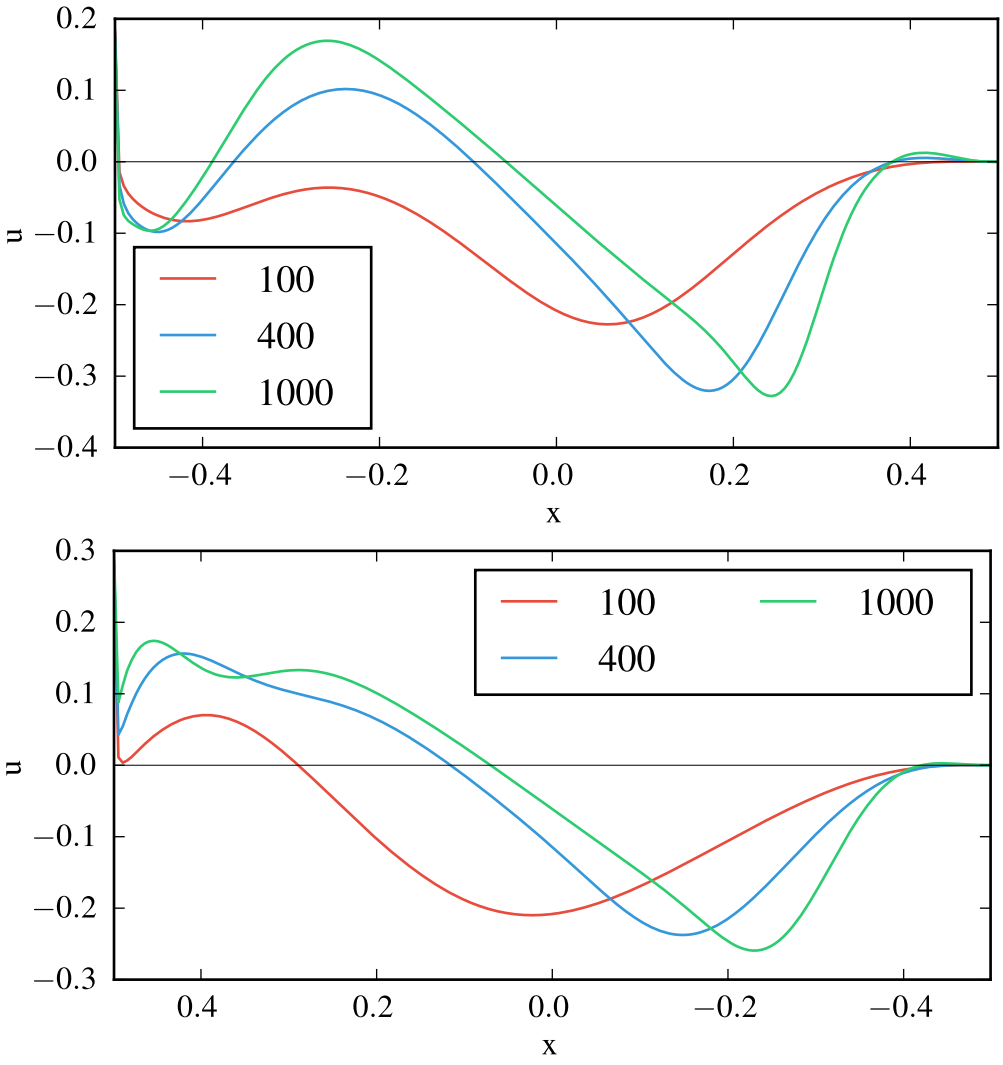


FIGURE B.6: $\Gamma = 1$, u along the diagonals of the cavity. The top graph shows u along the diagonal from the top left to the bottom right and the bottom graph depicts the values of u along the other diagonal. We see that the minimum of u shifts towards the boundary with higher Reynolds numbers.

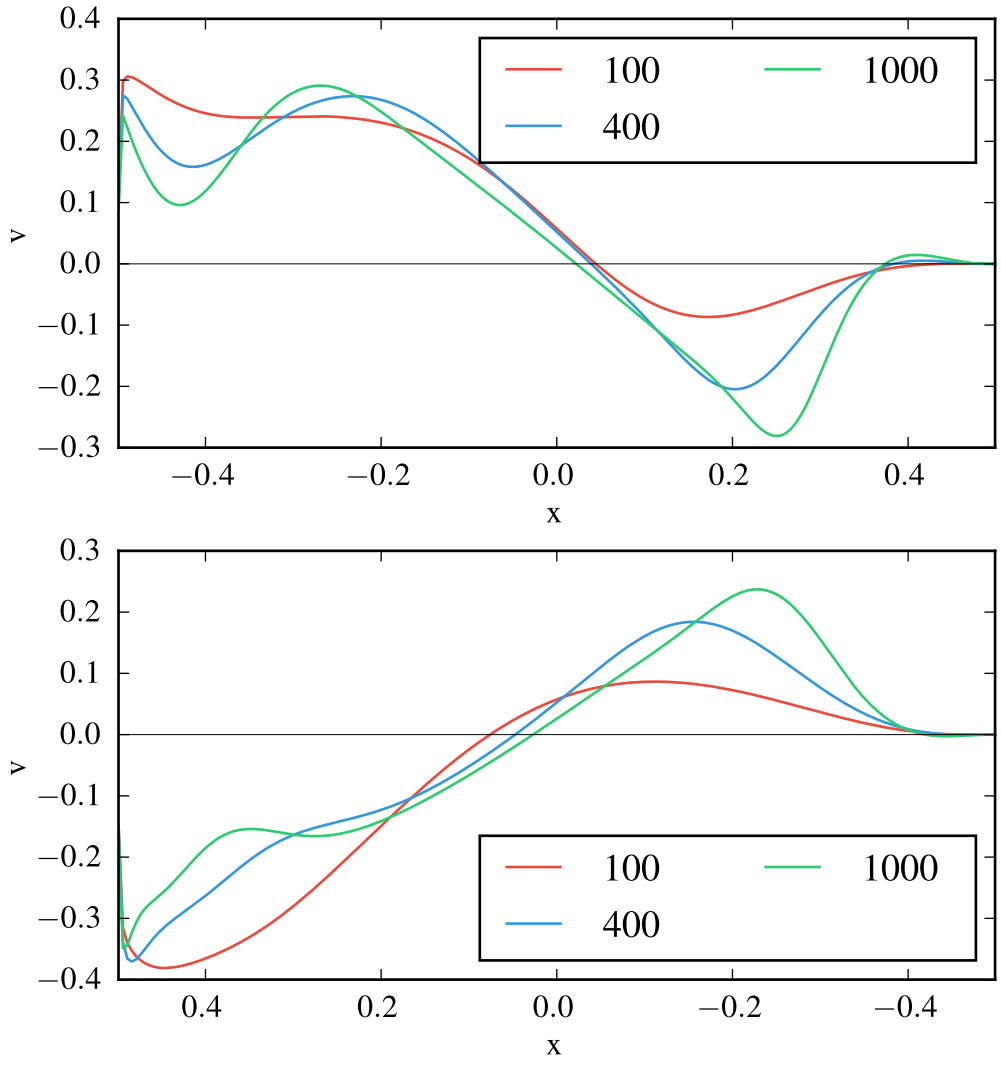


FIGURE B.7: $\Gamma = 1$, v along the diagonals of the cavity, The top graph shows v along the diagonal from the top left to the bottom right and the bottom graph depicts the values of v along the other diagonal. We see that the minimum of v shifts towards the boundary with higher Reynolds numbers.

B.2 $\Gamma = 2$

This section shows selected plots for the velocity components along the diagonals and center lines of the cavity for $Re = 100, 400$ and 1000. The selection is chosen as diverse is possible, s.t. the reader gets an impression for crucial regions and components. From now on we will denote the lines according to their direction, i.e. $\searrow$ and $\swarrow$ for the diagonals and $\downarrow$ and $\rightarrow$ for the two centerlines. We see that a mesh of 70×70 grid points before the refinement procedure is sufficient to grasp the features of the flow. Convergence up to graphical precision is reached in most of the cases. The only region, where the flows differ slightly is the top left corner but this difference does not have an influence on the flow in the center and may therefore yield good results also for the stability analysis.

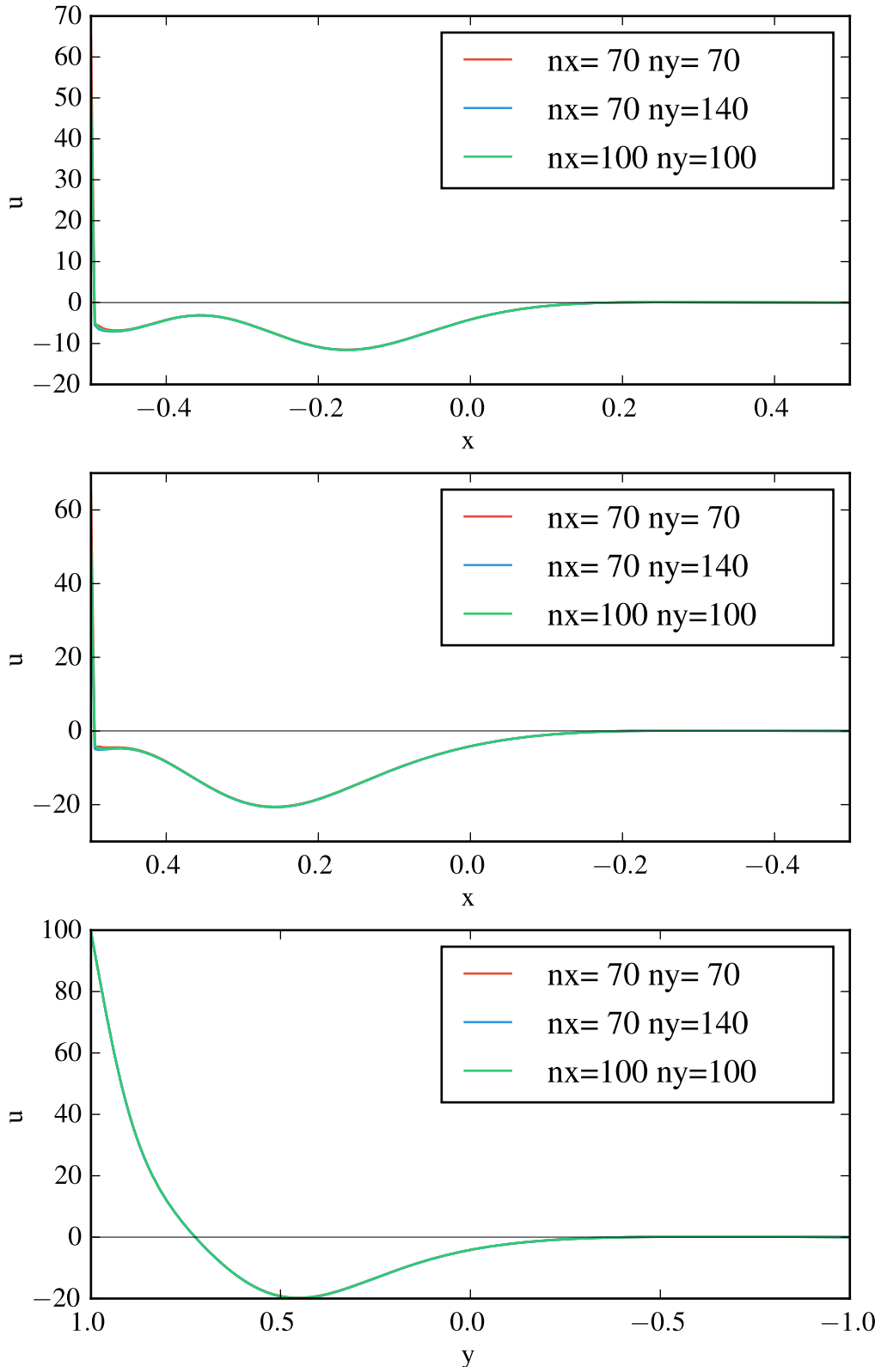


FIGURE B.8: $\Gamma = 2, Re = 100$ u. The three plots show the u-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\downarrow$, from top to bottom.

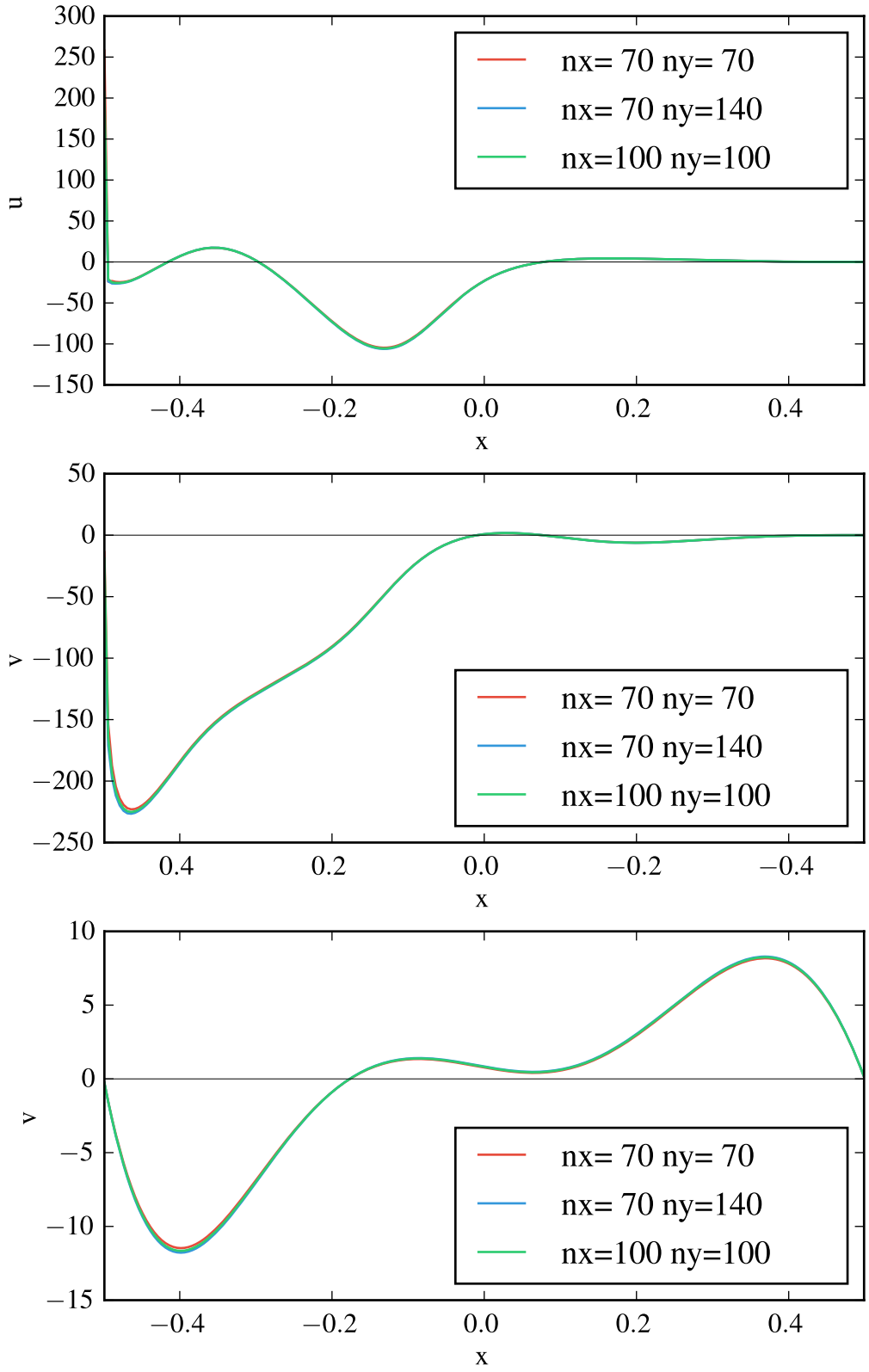


FIGURE B.9: $\Gamma = 2, Re = 400$ v, the three plots show the v-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\rightarrow$, from top to bottom.

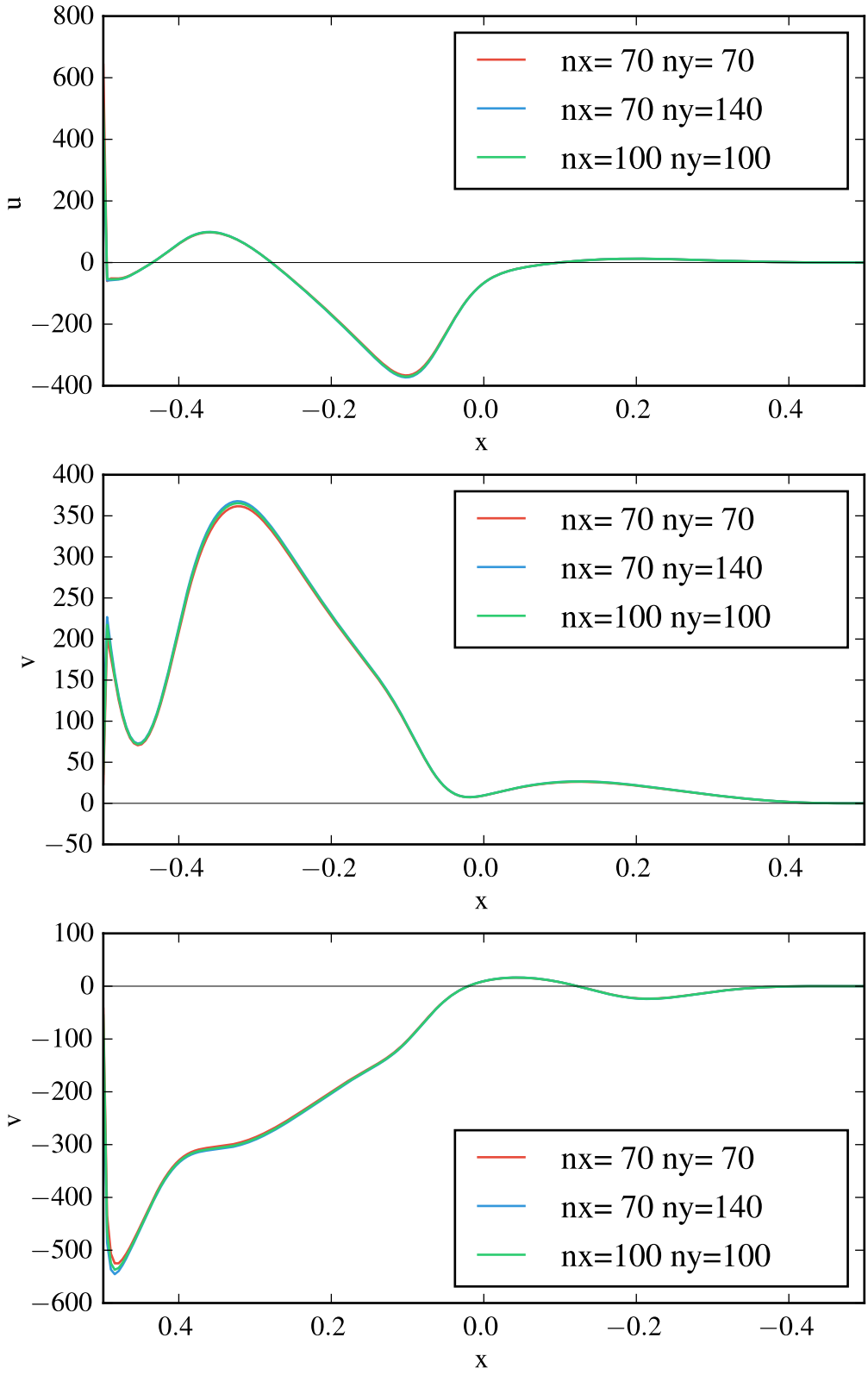


FIGURE B.10: $\Gamma = 2, Re = 1000$, the top panel shows the u component along $\searrow$, in the center v along $\searrow$ is shown and on the bottom v along $\swarrow$ is presented.

B.2.1 *Re* comparison

This section is meant to illustrate the two effects of an increasing Reynolds number on the flow for the $\Gamma = 2$ cavity: The first Figure B.11 shows that the penetration depth of the magnitude of the velocity increases if the Reynolds number is increased from 100 to 800 and the second Figure B.12 allows to emphasize the growth of the second vortex with increased *Re*.

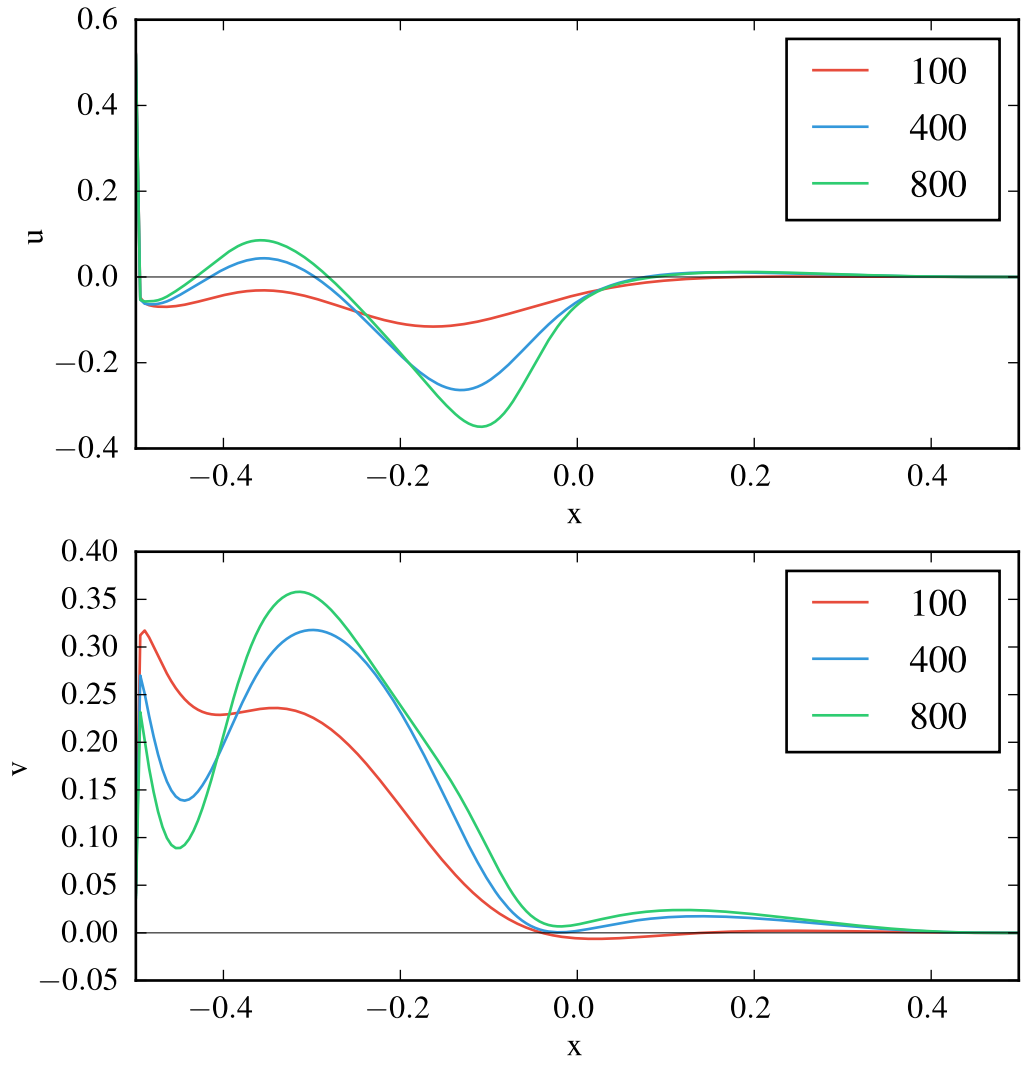


FIGURE B.11: u and v along the diagonal $\searrow$. We see that, as in the case of $\Gamma = 1$, the penetration into the cavity is enhanced, if the Reynolds number is increased.

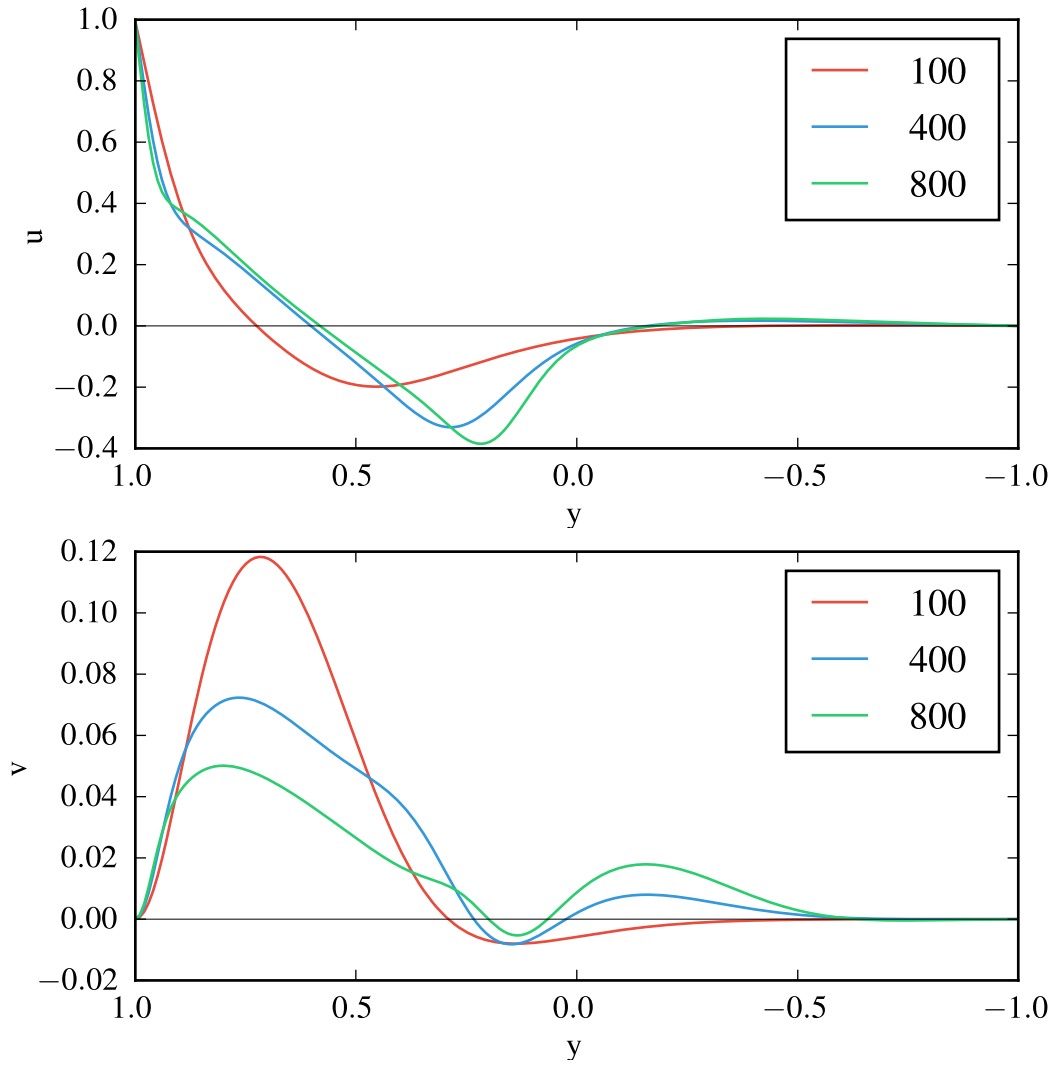


FIGURE B.12: u and v along the central line ↓. The second effect of an increased Reynolds number is visible in this plot: the evolution of a second vortex and its growth with higher Reynolds numbers.

B.3 $\Gamma = 3$

B.3.1 Grid convergence studies

This section is a repetition of the previous one for the case of $\Gamma = 3$. The convergence studies reveal, that a proper mesh needs at least 70×140 grid points in order to arrive at results, which are converged up to graphical accuracy.

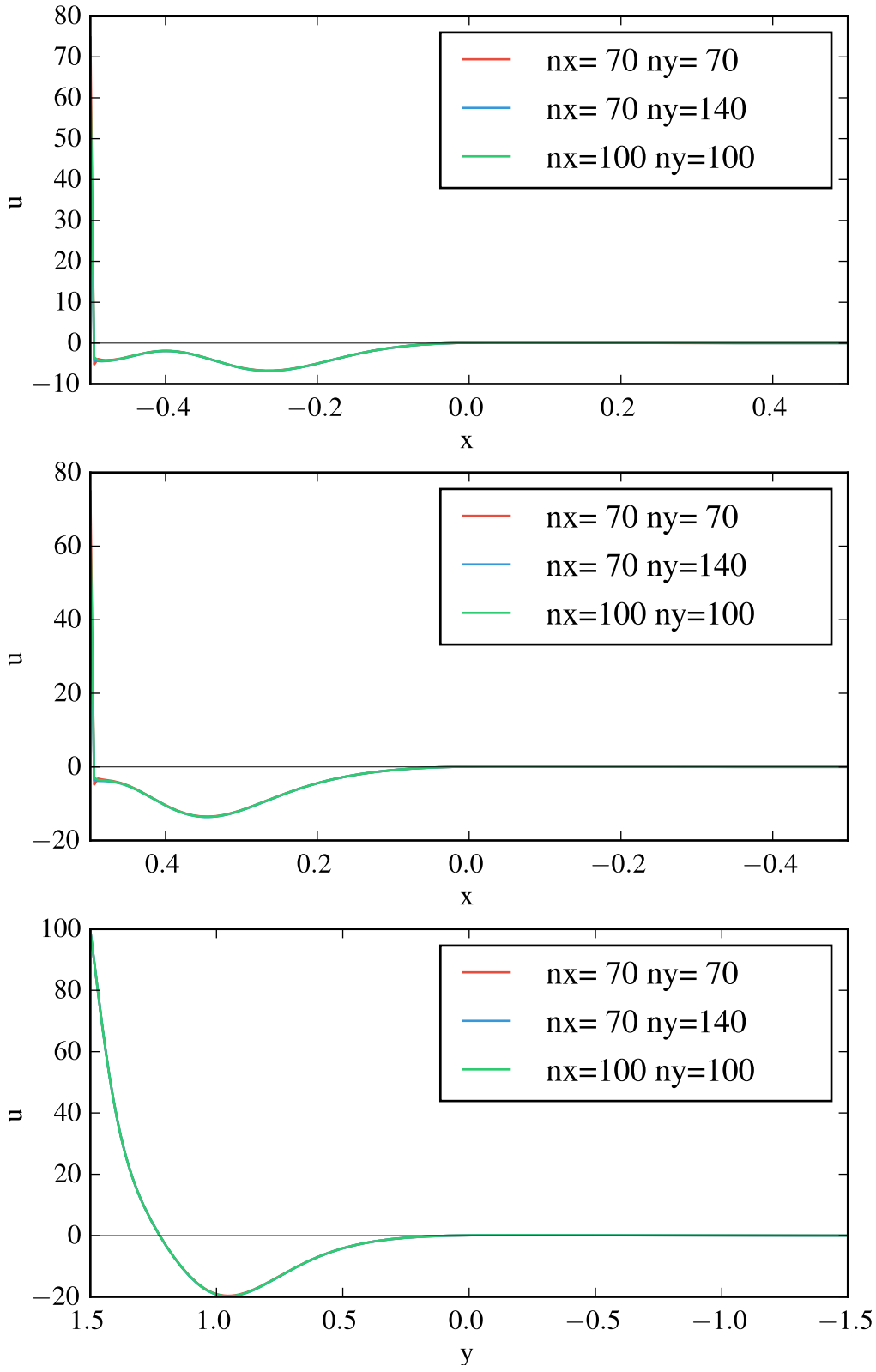


FIGURE B.13: $\Gamma = 3, Re = 100$ u. The three plots show the u-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\downarrow$, from top to bottom.

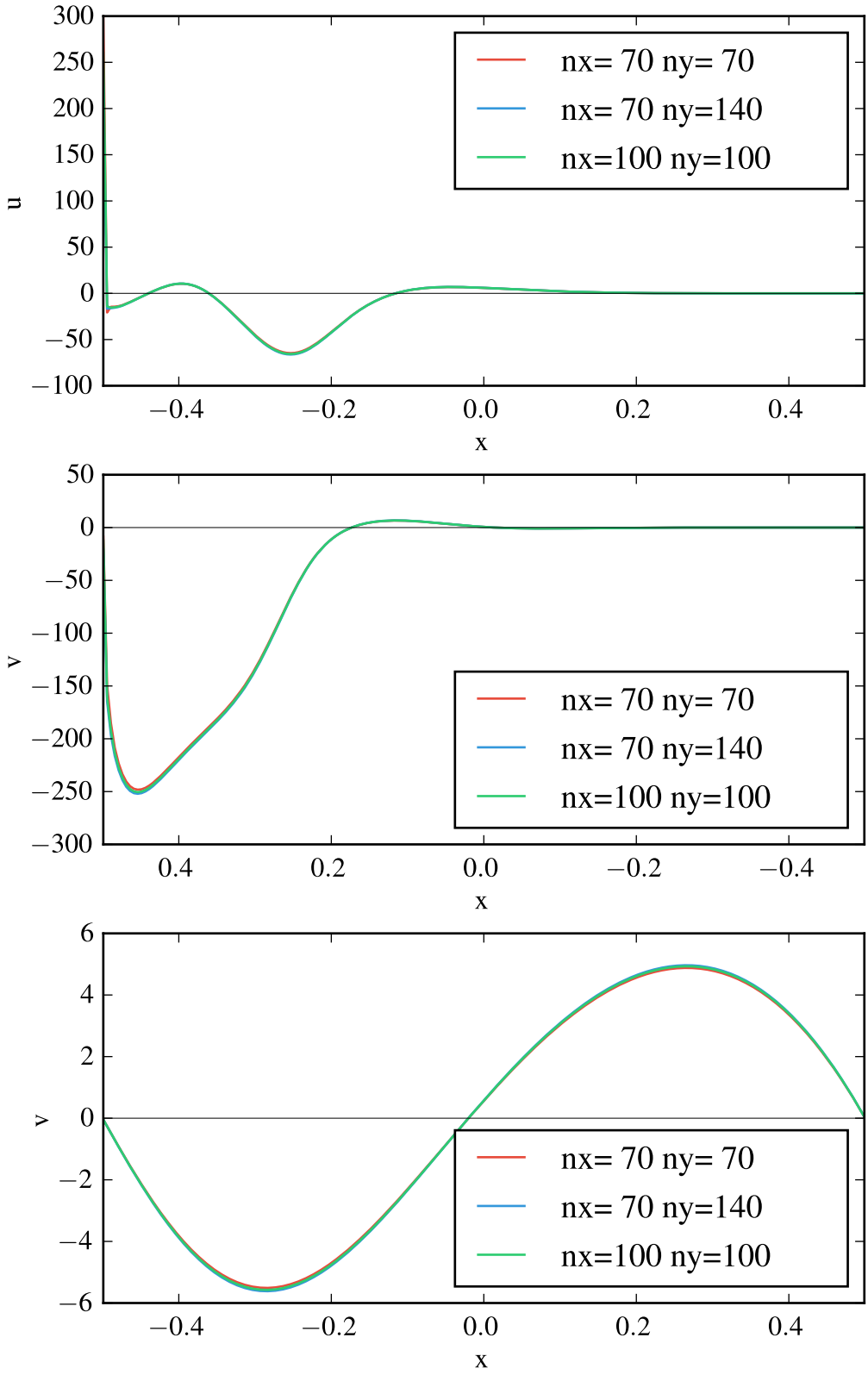


FIGURE B.14: $\Gamma = 3, Re = 400$ v. The three plots show the v-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\rightarrow$, from top to bottom.

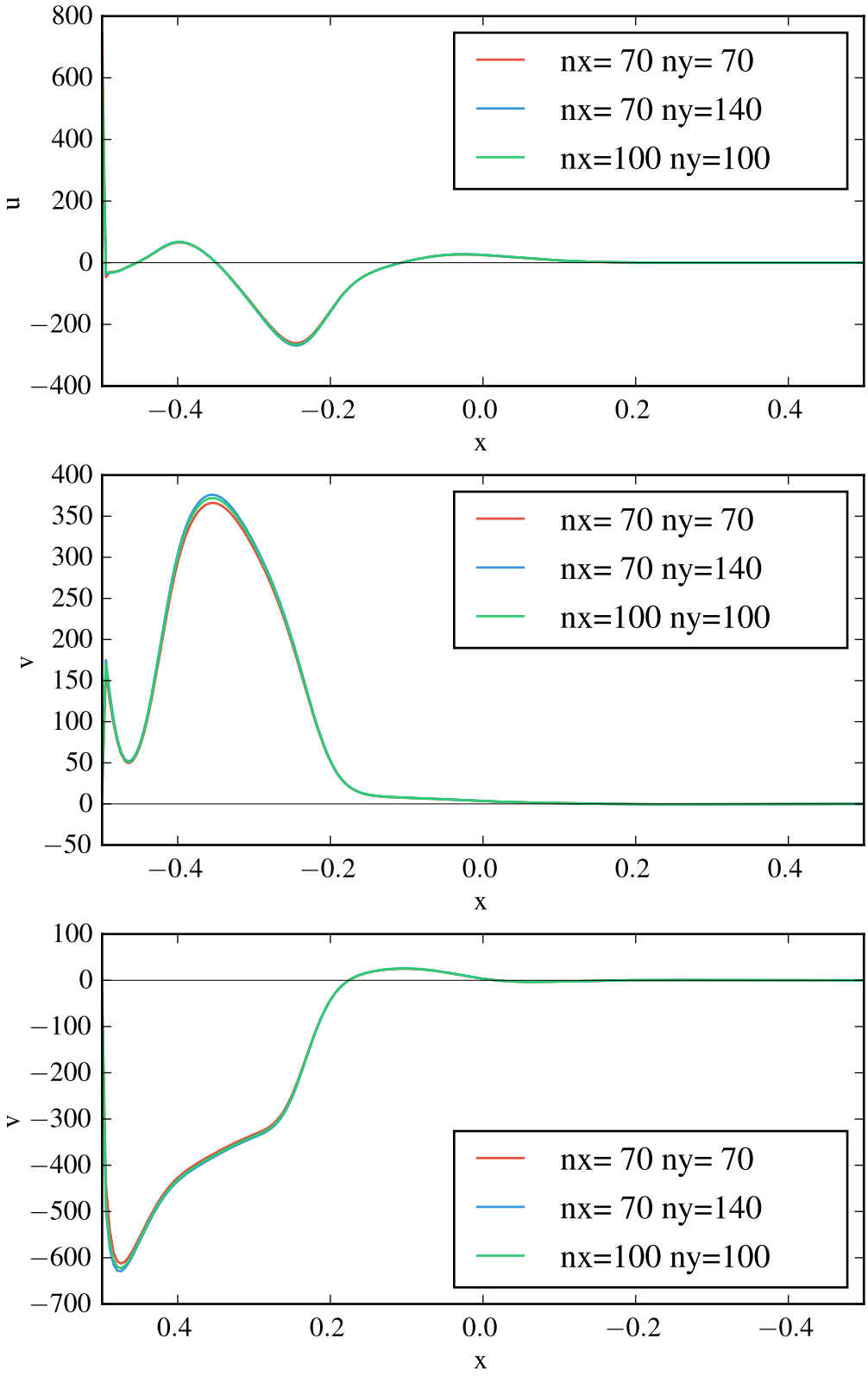


FIGURE B.15: $\Gamma = 3, Re = 1000$. The top panel shows the u component along $\searrow$, in the center v along $\searrow$ is shown and on the bottom v along $\swarrow$ is presented.

B.3.2 Re comparison

This section is meant to illustrate the two effects of an increasing Reynolds number on the flow for the $\Gamma = 3$ cavity: The first Figure B.16 indicates the penetration depth of the velocity magnitude with higher Reynolds numbers and the second Figure B.17 allows to emphasize the growth of the second and third vortices with increased Re.

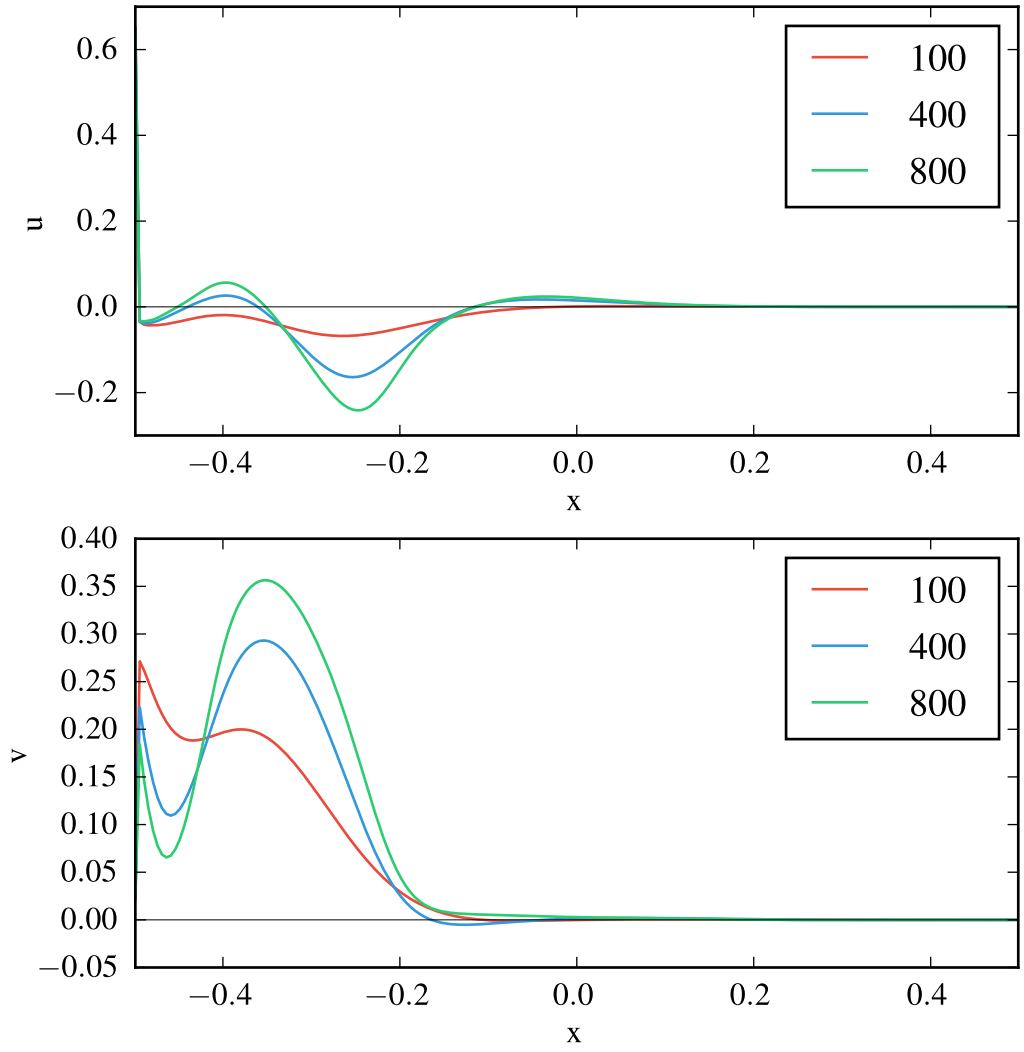


FIGURE B.16: u and v along the diagonal $\searrow$, we see that, as in the case of $\Gamma = 1$, the penetration into the cavity is enhanced, if the Reynolds number is increased.

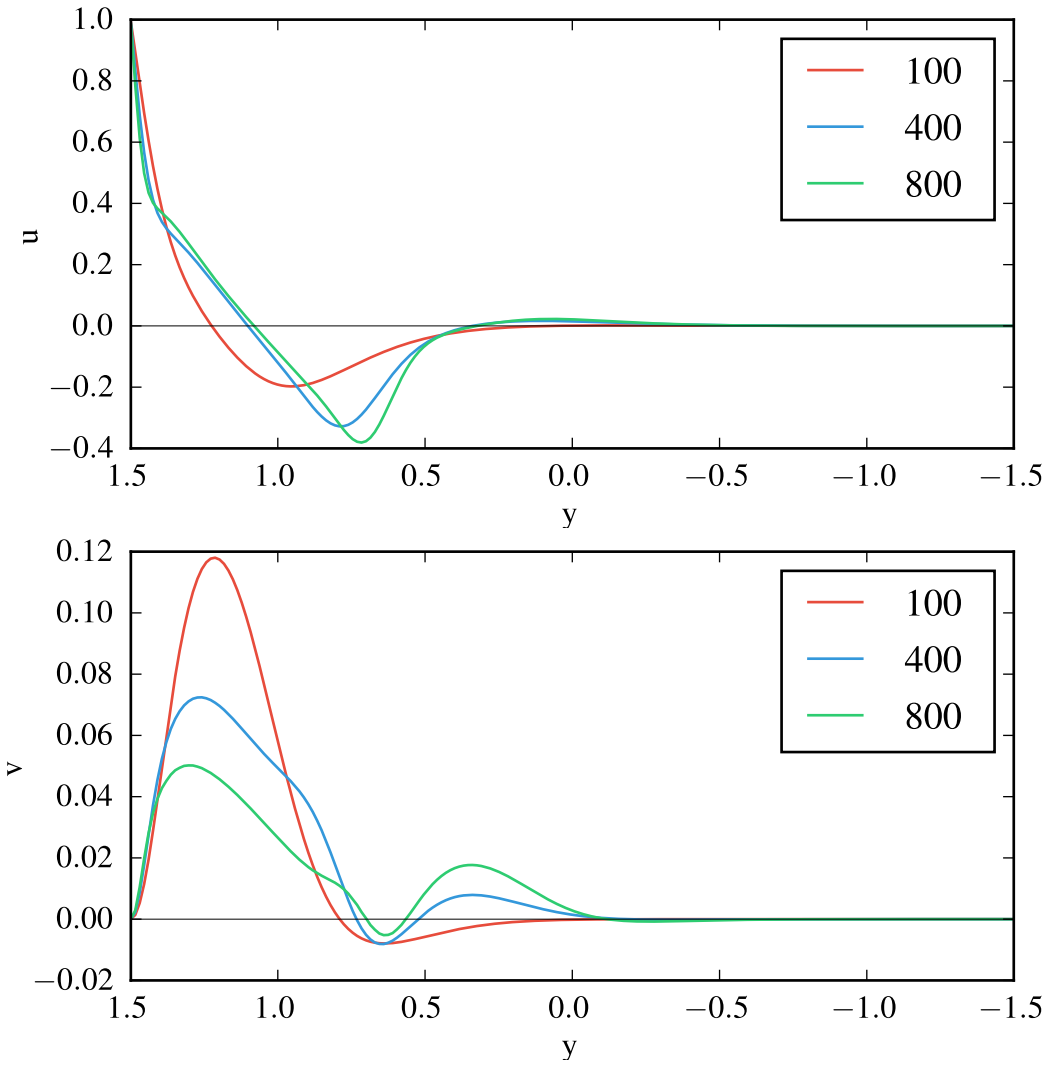


FIGURE B.17: u and v along the central line $\downarrow$. The second effect of an increased Reynolds number is visible in this plot: the evolution of a second vortex and its growth with higher Reynolds numbers.

B.4 $\Gamma = 0.5$

B.4.1 Grid convergence studies

This section provides the convergence studies for $\Gamma = 0.5$. It is seen that the quantities are converged for a $70x70$ grid. Only small differences are encountered in the very close vicinity of the boundary.

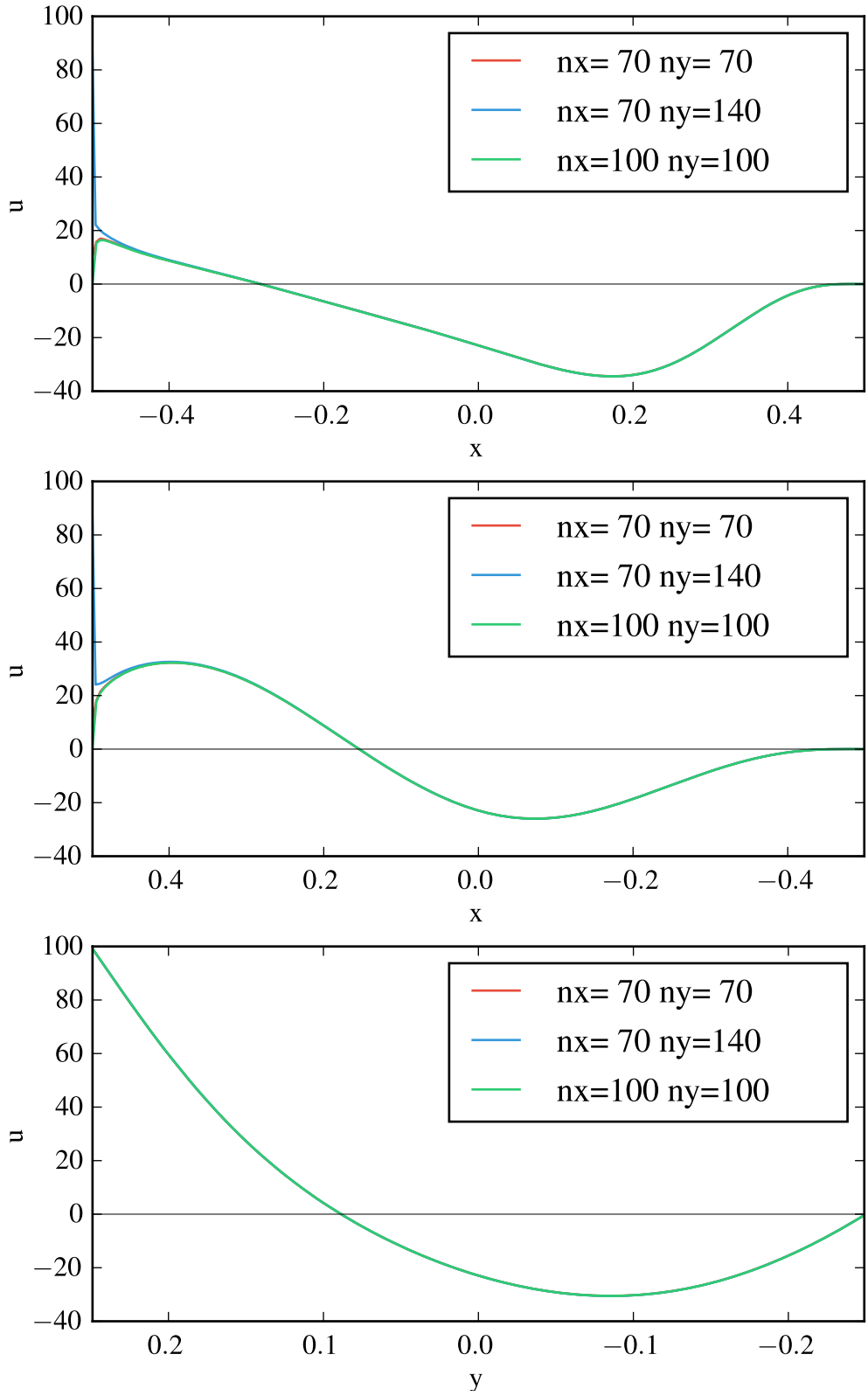


FIGURE B.18: $\Gamma = 0.5, Re = 100$ u. The three plots show the u-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\downarrow$, from top to bottom.

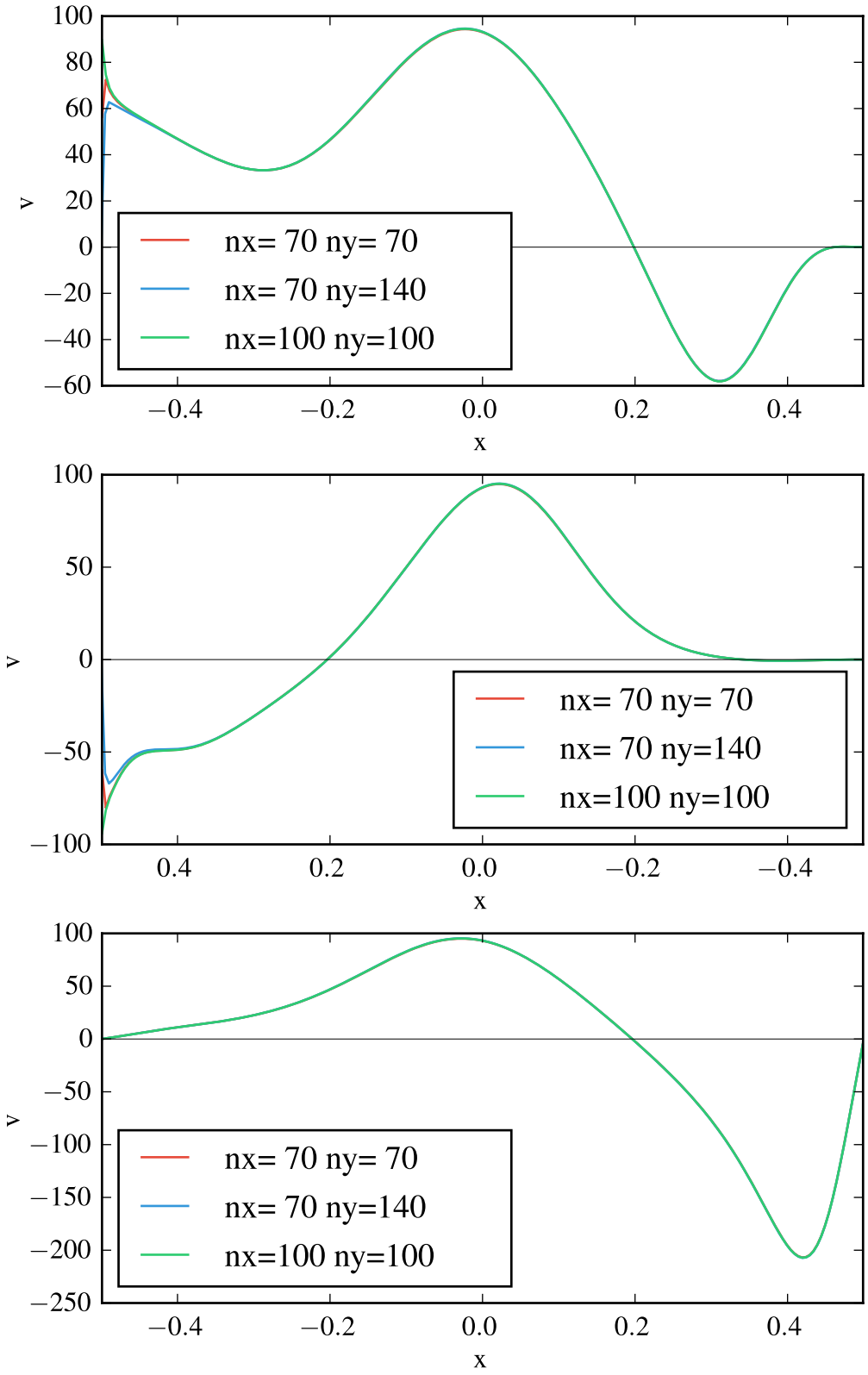


FIGURE B.19: $\Gamma = 3, Re = 400$ v. The three plots show the v-component of the velocity field along the lines $\searrow$, $\swarrow$ and $\rightarrow$, from top to bottom.

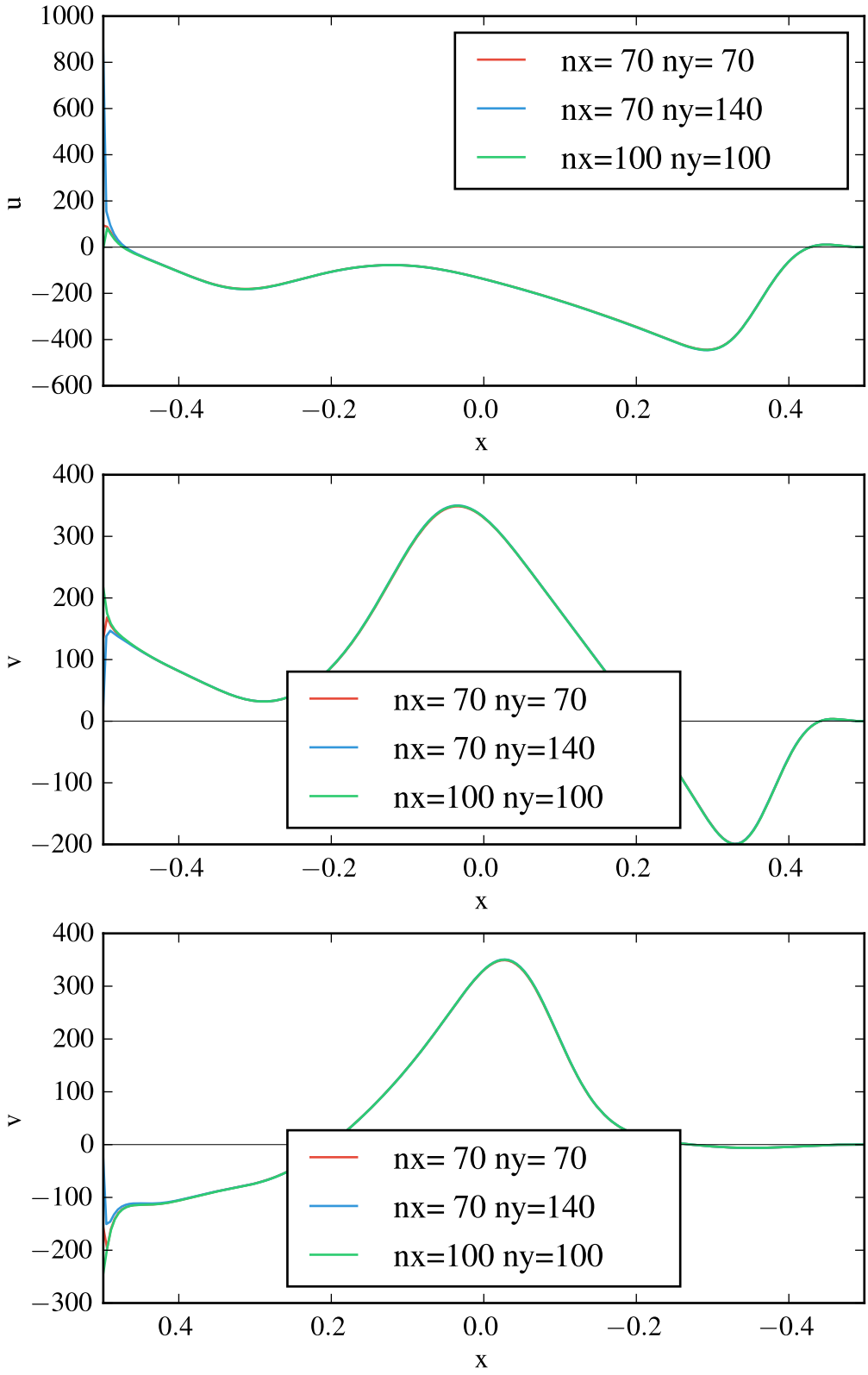


FIGURE B.20: $\Gamma = 3, Re = 1000$. The top panel shows the u component along $\searrow$, in the center v along $\searrow$ is shown and on the bottom v along $\swarrow$ is presented.

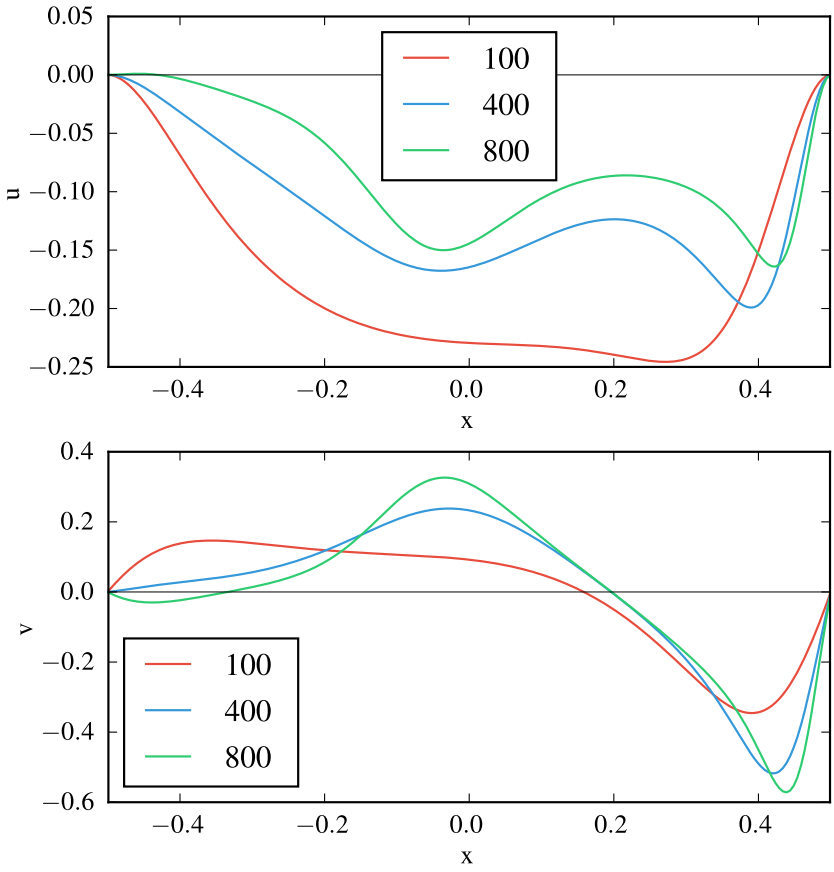


FIGURE B.21: u and v along the central line $\rightarrow$.

B.4.2 *Re* comparison

In Figure B.21 we see that for higher Re a second vortex is arising in the $\Gamma = 0.5$ cavity.

Appendix C

Additional data for Chapter 4

This Appendix holds data and explanations from Chapter 4, which did not fit in the main text.

C.1 Table of Re_c and k_c for the variation of Γ

C.2 Critical mode analysis $\Gamma = 1$, $\alpha = 7°$

Since there is an interesting feature in the localization of the energy production rate for the critical mode for $\Gamma = 1$ and $\alpha = 7°$, we elaborate on the mechanism of energy production for this mode in this section. The production rate was shown in Figure 4.7, where we explained the mechanism for the main energy gain, stemming from the contribution $I_2 = -\frac{1}{D^*}\vec{\tilde{v}}_{\parallel} \cdot (\vec{\tilde{v}}_{\perp} \cdot \vec{\nabla}\vec{v}_0)$. Here, we present the most important contributions for the energy production at the slice with $y = 0.2$, where we find an explanation for arising contributions from the term $I_3 = -\frac{1}{D^*}\vec{\tilde{v}}_{\perp} \cdot (\vec{\tilde{v}}_{\parallel} \cdot \vec{\nabla}\vec{v}_0)$. The Figures C.1, C.2 and C.3 show the basic flow, the total energy production rate and the separate contributions (the plots are not normalized with D^*). The Figures illustrate that a big contribution of I_2 may be seen in this slice with an energy production mechanism that was explained in the main text. However, there are also big regions, where I_2 is negative and these coincide with the observed nose in the production rate, where the mechanism of energy production is different, as the main contribution at this location comes from I_3, which means that there has to be a gradient of the basic flow parallel to the velocity field for components which are orthogonal to the flow: The magnitude of the basic flow at the nose position ($x \approx 0.2$) is approximately given by $(+200, -100, +50)$, the perturbation velocity is $\approx (+3, 0.1, 0)$, resulting in $\vec{\tilde{v}}_{\parallel} = (0.84, 1.29, -0.54)$ and $\vec{\tilde{v}}_{\parallel} = (+3, 0.1, 0)$. The gradients of the basic flow for u and v are shown in Figure C.4, where we see that the gradients have the form $\nabla v_i = (-, +, 0)$ and thus the projection along the parallel perturbation gives a big negative value, which results in a positive production rate, as the contraction

α	Γ	Re_c	k_c	α	Γ	Re_c	k_c
0.000	0.880	801.172	6.538	45.000	0.880	803.613	5.789
0.000	0.900	810.156	16.237	45.000	0.900	795.557	5.789
0.000	0.920	806.641	16.070	45.000	0.920	786.523	5.735
0.000	0.940	802.734	15.819	45.000	0.940	775.781	5.682
0.000	0.960	798.828	15.652	45.000	0.960	764.795	5.682
0.000	0.980	794.531	15.485	45.000	0.980	753.809	5.629
0.000	1.000	790.312	15.266	45.000	1.000	741.875	5.621
0.000	1.020	785.938	15.150	45.000	1.020	730.127	5.576
0.000	1.040	781.250	14.983	45.000	1.040	719.141	5.576
0.000	1.060	775.781	14.816	45.000	1.060	708.887	5.576
0.000	1.080	770.312	14.649	45.000	1.080	699.609	5.576
0.000	1.100	764.453	14.565	45.000	1.100	659.570	3.130
22.500	0.880	600.391	7.350	67.500	0.880	682.031	4.101
22.500	0.900	608.301	7.250	67.500	0.900	676.660	4.101
22.500	0.920	615.918	7.200	67.500	0.920	671.533	4.039
22.500	0.940	622.070	7.150	67.500	0.940	667.139	3.977
22.500	0.960	627.344	7.100	67.500	0.960	663.477	3.915
22.500	0.980	629.980	7.050	67.500	0.980	660.303	3.853
22.500	1.000	630.703	7.035	67.500	1.000	659.688	3.775
22.500	1.020	628.809	6.950	67.500	1.020	656.641	3.728
22.500	1.040	624.121	6.950	67.500	1.040	656.152	3.666
22.500	1.060	616.211	6.850	67.500	1.060	662.744	3.542
22.500	1.080	605.957	6.800	67.500	1.080	637.842	1.930
22.500	1.100	593.359	6.750	67.500	1.100	600.244	1.961

TABLE C.1: **Variation of Γ - calculated values**

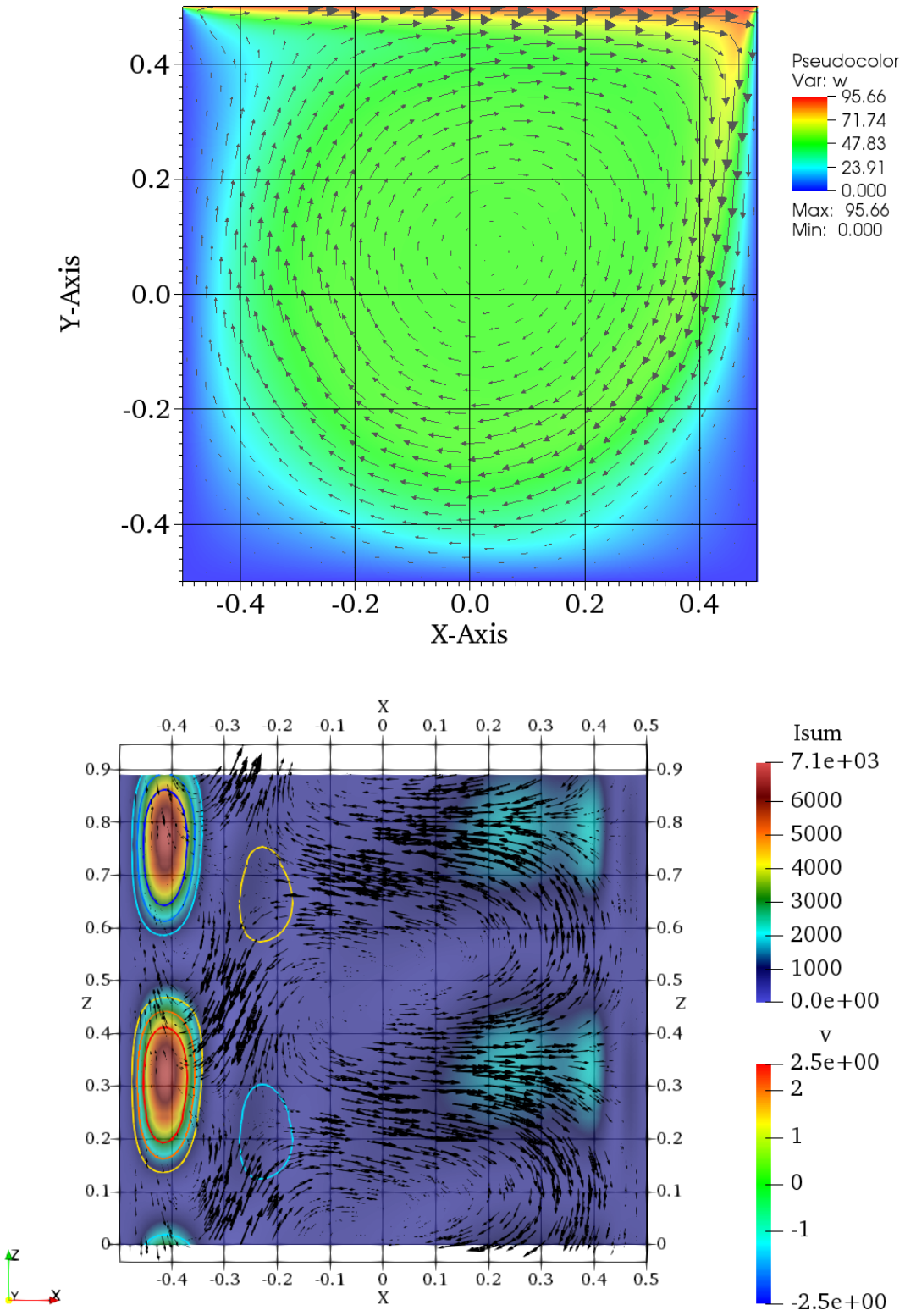


FIGURE C.1: Top: the basic flow with arrows for the u- and v-components of basic flow and the color denotes the magnitude of the w-component. Bottom: $\sum_i I_i$ for the slice at $y = 0.2$ for $\Gamma = 1$ and $\alpha = 7°$ for the critical mode. The color denotes the energy production rate, the arrows the u- and w-components of the velocity of the perturbation and the isolines denote the $v-$component of the perturbation.

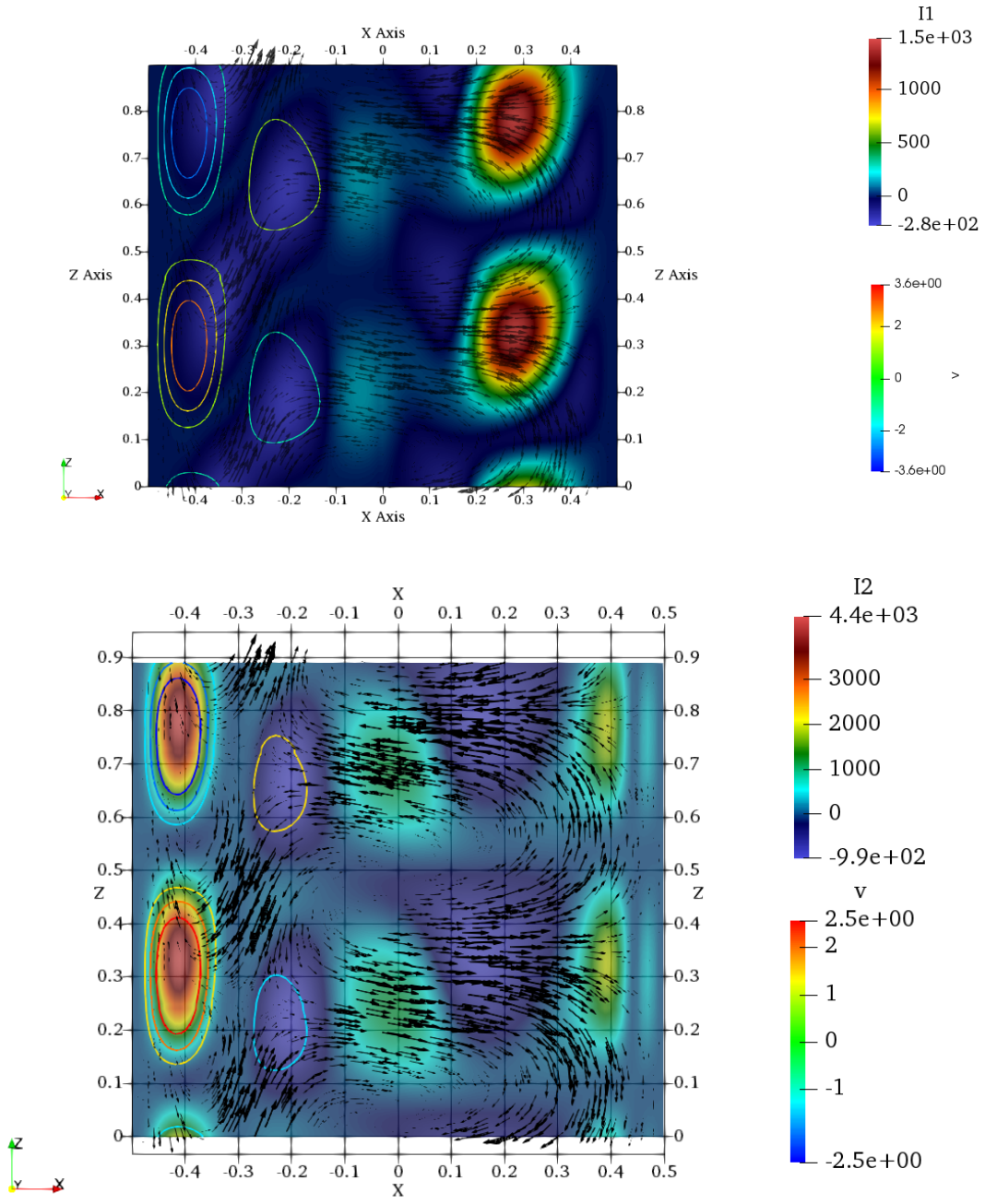


FIGURE C.2: **Energy transfer terms for $y = 0.2$ for the critical mode for $\Gamma = 1$ and $\alpha = 7°$**, the colors denote the energy production rate, the arrows the $u-$ and $w-$components of the velocity of the perturbation and the isolines denote the $v-$component of the perturbation. The shown energy transfer terms are I_1 (top) and I_2 (bottom).

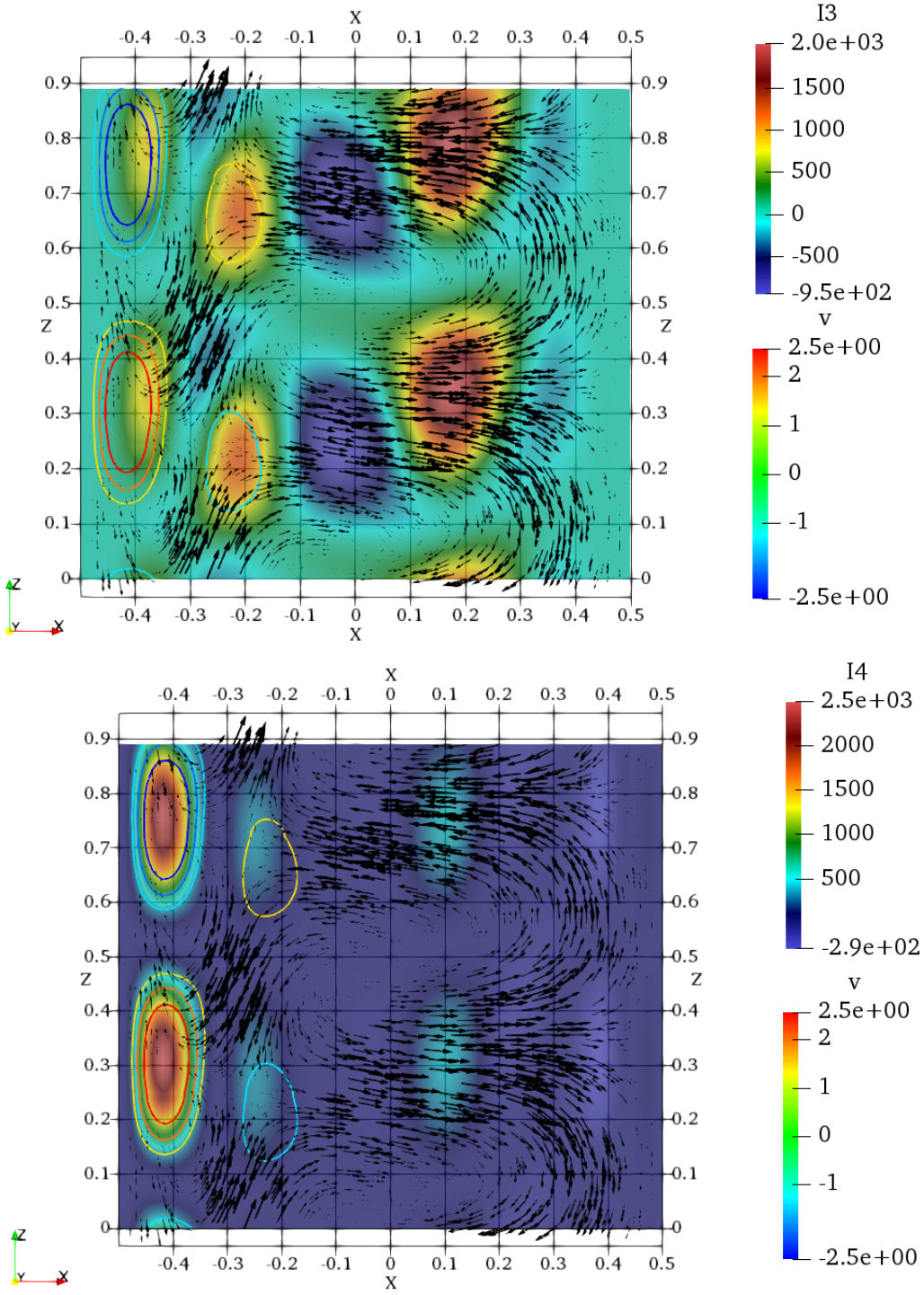


FIGURE C.3: **Energy transfer terms for $y = 0.2$ for the critical mode for $\Gamma = 1$ and $\alpha = 7°$**, the colors denote the energy production rate, the arrows the $u-$ and $w-$components of the velocity of the perturbation and the isolines denote the $v-$component of the perturbation. The shown energy transfer terms are I_3 (top) and I_4 (bottom).

with the orthogonal perturbation does not result in a sign change. The most general thing to extract here is, that the gradients of the basic flow with respect to the $z-$coordinates vanish, which means that for a fixed magnitude of the perturbation vector, the largest possibility for an energy gain is at positions, where $\tilde{w} = 0$, which was the case for the mechanisms described in the main text as well as the mechanism for I_3.

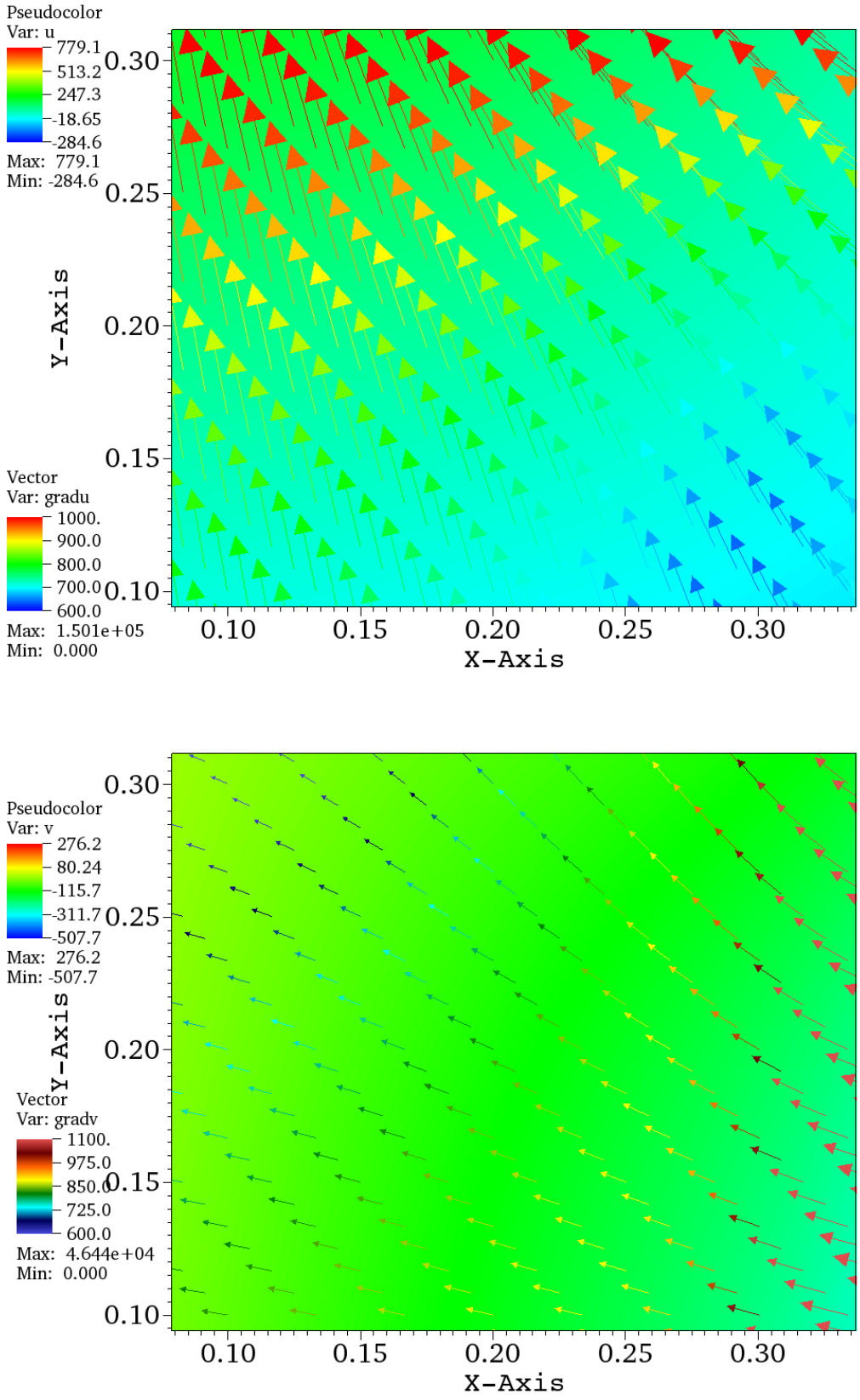


FIGURE C.4: **Basic flow gradients at** $y = 0.2$, $x = 0.2$ ($\Gamma = 1, \alpha = 7°$,top: u, bottom: v. The background is colored with the magnitude of the velocity component and the arrows denote the respective gradients.

Appendix D

Scripts

D.1 Python - FENICS control

D.1.1 Calculation script

```
1  from fenics import *
2  import matplotlib
3  matplotlib.use('pdf')
4  import matplotlib.pyplot as plt
5  import scipy.sparse as sp
6  from scipy.sparse.linalg import eigs
7  import numpy as np
8  import os
9  import sys
10
11 class mycav:
12  def __init__(self,Gamma,alpha,calcdir=None):
13   self.Gamma=Gamma
14   self.alpha=alpha
15   if (calcdir!=None):
16    self.calcdir=calcdir
17    if not os.path.exists(calcdir):
18     os.mkdir(calcdir)
19  def analyze(self,Re,relax_param,lplot=False):
20   self.relax_param=relax_param
21   print('RUNNING ANALYZE WITH REYNOLDS: {}'.format(Re))
22   print('1st RUN --> coarse GRID')
23   filename=self.calcdir+'/'+'Re'+'_{:08.3f}'.format(Re)
24   Refile=open(filename,'w')
25   karr=np.linspace(0.001,30,30)
26   nx=90 #70
27   ny=90 #70
28   self.getbaseflow(Re,nx,ny)
29   sigma,omega=self.linearstab(Re,karr,lplot)
30   for line in range(len(sigma)):
31       Refile.write('{:10g}    {:10g}    {:10g} \n'.format(karr[line],sigma[line
             ],omega[line]))
32   nr,changearr,indices=self.getsignchange(sigma)
33   print('FOUND {} SIGN CHANGES'.format(nr))
34   #1st REFINEMENT OF THE MESH FOR THE CRITICAL k-VALUES
35   if(nr>0):
36    nx=100 #80
37    ny=100 #80
```

```
38      self.getbaseflow(Re,nx,ny)
39      for i in indices:
40       k1=karr[i];k2=karr[i+1]
41       conv=100
42       kdiff=10
43       run=0
44       found=True
45       while(conv>1e-3 and kdiff > 0.1):
46        run+=1
47        print('LOOKING IN THE VICINITY OF k= {}'.format(k1))
48        k_refine=np.linspace(k1,k2,5)
49        sigma2,omega2=self.linearstab(Re,k_refine,lplot)
50        nr2,changearr2,indices2=self.getsignchange(sigma2)
51        if(nr2==0 and run==1):
52         k1=karr[i-1]
53         k2=karr[i+2]
54         continue
55        if(nr==2 and run>1):
56         found=False
57         break
58        else:
59         k1=k_refine[indices2[0]]
60         k2=k_refine[indices2[0]+1]
61         found=True
62        kdiff=abs(k2-k1)
63        conv=abs(self.find_nearest(sigma2,0)[1])
64       for line in range(len(sigma2)):
65           Refile.write('{:10g}     {:10g}     {:10g} \n'.format(k_refine[line],
               sigma2[line],omega2[line]))
66       if found:
67        print('SIGMA(k_crit) before 2nd REFINEMENT: {}'.format(sigma2[indices2
             [0]]))
68        #EXTRACT THE VALUE CLOSEST TO 0
69        kchoose=k_refine[self.find_nearest(sigma2,0)[0]]
70        #2nd REFINEMENT TO CHECK FOR CONVERGENCE
71        nx=110 #90
72        ny=110 #90
73        self.getbaseflow(Re,nx,ny)
74        sigma2,omega2=self.linearstab(Re,np.asarray([kchoose]),lplot)
75        print('SIGMA(k_crit) after 2nd REFINEMENT: {}'.format(sigma2[0]))
76        Refile.write('{:10g}     {:10g}     {:10g} \n'.format(kchoose,sigma2[0],
             omega2[0]))
77       else:
78        print('WARNING: DID NOT FIND ZERO AFTER 1st REFINEMENT FOR k = {}'.format(
             karr[i]))
79     Refile.close()
80     self.plot_k_sigma(Re,filename)
81
82    def findRe_c(self,Re_low,Re_high,lplot=False):
83     resultname=self.calcdir+'/'+'REc_Gamma_{:5.2f}_alpha_{:5.2f}'.format(self.
         Gamma,self.alpha)
84     Resultfile=open(resultname,'w')
85     karr=np.linspace(0.001,30,60)
86     nx=120  #75
87     ny=120  #75
88     Rearr=np.array([Re_low,Re_high])
89     Remid=(Re_low+Re_high)/2.
90     while (np.min(abs(Rearr-Remid))>1.):
```

```
91      print('RUNNING findRe_c WITH REYNOLDS: {}'.format(Remid))
92      filename=self.calcdir+'/'+'Re_crit'+'_{:08.3f}'.format(Remid)
93      Refile=open(filename,'w')
94      if(Remid<500):
95          self.relax_param=0.5
96      elif(Remid>=500 and Remid<1200):
97          self.relax_param=0.3
98      else:
99          self.relax_param=0.1
100     self.getbaseflow(Remid,nx,ny)
101     sigma,omega=self.linearstab(Remid,karr,lplot)
102     for line in range(len(sigma)):
103         Refile.write('{:10g}     {:10g}     {:10g} \n'.format(karr[line],sigma[
                line],omega[line]))
104     nr,changearr,indices=self.getsignchange(sigma)
105     Resultfile.write('{:7.2f}     {:2d} \n'.format(Remid,nr))
106     print('FOUND {} SIGN CHANGES'.format(nr))
107     if(nr>0):
108         Rearr[1]=Remid
109     elif(nr==0):
110         Rearr[0]=Remid
111     Remid=(Rearr[0]+Rearr[1])/2.
112     Refile.close()
113   Resultfile.close()
114 def findRe_c_MORE_EV(self,Re_low,Re_high,nr_EV,kmin,kmax,nr_k,lplot=False):
115   resultname=self.calcdir+'/'+'REc_Gamma_{:5.2f}_alpha_{:5.2f}'.format(self.
        Gamma,self.alpha)
116   Resultfile=open(resultname,'w')
117   karr=np.linspace(kmin,kmax,nr_k)
118   nx=120  #75
119   ny=120  #75
120   Rearr=np.array([Re_low,Re_high])
121   Remid=(Re_low+Re_high)/2.
122   while (np.min(abs(Rearr-Remid))>0.2):
123     print('RUNNING findRe_c WITH REYNOLDS: {}'.format(Remid))
124     filename=self.calcdir+'/'+'Re_crit'+'_{:08.3f}'.format(Remid)
125     Refile=open(filename,'w')
126     if(Remid<500):
127         self.relax_param=0.5
128     elif(Remid>=500 and Remid<1200):
129         self.relax_param=0.3
130     else:
131         self.relax_param=0.1
132     self.getbaseflow(Remid,nx,ny)
133     sigma_low,omega_low,sigma,omega=self.linearstab_MORE_EV(Remid,karr,nr_EV,
          lplot)
134     for line in range(len(sigma)):
135       Refile.write('{:10g}     {:10g}     {:10g} \n'.format(karr[line],sigma_low[
            line],omega_low[line]))
136       Refile.write('OTHER EIGENVALS FOR THIS K-VECTOR: \n')
137       for ij in range(nr_EV):
138         Refile.write('sigma = {}     , omega = {}  \n'.format(sigma[line,ij],omega[
              line,ij]))
139     nr,changearr,indices=self.getsignchange(sigma_low)
140     if(len(indices)==0):
141       Resultfile.write('{:7.2f}     {:2d} \n'.format(Remid,nr))
142     else:
```

```
143         Resultfile.write('{:7.2f}      {:2d}   {:10g} \n'.format(Remid,nr,karr[indices
                [0]]))
144       print('FOUND {} SIGN CHANGES'.format(nr))
145       if(nr>0):
146           Rearr[1]=Remid
147       elif(nr==0):
148           Rearr[0]=Remid
149       Remid=(Rearr[0]+Rearr[1])/2.
150       Refile.close()
151     Resultfile.close()
152
153   def singlescan(self,Gammaarr,alphaarr,Rearr,karr,nx,ny):
154     nx=100  #75
155     ny=100  #75
156     for Gamma in Gammaarr:
157       for alpha in alphaarr:
158         for Re in Rearr:
159           resultname='./'+'SCAN_Gamma_{:5.2f}_alpha_{:5.2f}'.format(Gamma,alpha)
160           print('''
161           PARAMETERS
162           Gamma =  {}
163           alpha =  {}
164           Re    =  {}
165           k     =  {}
166           '''.format(Gamma,alpha,Re))
167           anafile=open(resultname,'a')
168           if(Re<500):
169               self.relax_param=0.5
170           elif(Re>=500 and Remid<1200):
171               self.relax_param=0.3
172           else:
173               self.relax_param=0.1
174           self.getbaseflow(Re,nx,ny)
175           sigma,omega=self.linearstab(Re,karr,lplot)
176           for line in range(len(sigma)):
177               anafile.write('{:10g}    {:10g}    {:10g} \n'.format(karr[line],sigma
                    [line],omega[line]))
178           anafile.close()
179
180   def getbaseflow(self,Re,nx,ny):
181     parameters["mesh_partitioner"]='ParMETIS'
182     #parameters["num_threads"] = 1
183     #CREATE THE MESH
184     self.mesh=RectangleMesh(Point(-1./2.,-self.Gamma/2.),Point(1./2.,self.Gamma
          /2.),nx,ny)
185     #REFINE THE MESH HERE
186     self.mymeshrefine()
187     #DEFINE TRIAL AND TEST FUNCTIONS
188     self.P2 = VectorElement('P',triangle,degree=2,dim=3)
189     self.P1 = FiniteElement('P',triangle,degree=1)
190     self.element = MixedElement([self.P2,self.P1])
191     self.W = FunctionSpace(self.mesh,self.element)
192     self.duvw,self.dp = TestFunctions(self.W)
193     self.du,self.dv,self.dw = split(self.duvw)
194     #THE FUNCTIONS FOR THE FORMULATION
195     uvwp = Function(self.W)
196     uvw,p = split(uvwp)
197     u,v,w = split(uvw)
```

```
198     #THE FUNCTIONS FOR THE STEADY STATE SOLUTION
199     self.uvwp0 = Function(self.W)
200     self.uvw0,p0 = split(self.uvwp0)
201     self.u0,self.v0,self.w0 = split(self.uvw0)
202     Gamma=self.Gamma
203     tol=1e-14
204     #DEFINE THE BOUNDARIES
205     class BoundN(SubDomain):
206      def inside(self,x,on_boundary):
207       return on_boundary and near(x[1],Gamma/2.,tol)
208     class BoundSEW(SubDomain):
209      def inside(self,x,on_boundary):
210       return on_boundary and (near(x[1],-Gamma/2.,tol) or near(x[0],-1./2.,tol)
211                  or near(x[0],1./2.,tol))
212     self.bcN=BoundN()
213     self.bcSEW=BoundSEW()
214     ubcN = DirichletBC(self.W.sub(0),Constant((Re*np.cos(self.alpha/180.*np.pi)
            ,0.,Re*np.sin(self.alpha/180.*np.pi))),self.bcN)
215     ubcSEW= DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcSEW)
216     bcs=[ubcSEW,ubcN]
217
218     ###NS-equation
219     F = self.du*u*u.dx(0)*dx+self.du*v*u.dx(1)*dx+inner(grad(self.du),grad(u))*dx
            +self.du*p.dx(0)*dx +\
220          self.dv*u*v.dx(0)*dx+self.dv*v*v.dx(1)*dx+inner(grad(self.dv),grad(v))*dx
                +self.dv*p.dx(1)*dx +\
221          self.dw*u*w.dx(0)*dx+self.dw*v*w.dx(1)*dx+inner(grad(self.dw),grad(w))*dx
                +\
222          self.dp*u.dx(0)*dx+self.dp*v.dx(1)*dx
223
224     J=derivative(F,uvwp)
225     M=u.dx(0)*dx+v.dx(1)*dx
226     problem = NonlinearVariationalProblem(F, uvwp, bcs, J=J)
227     solver = AdaptiveNonlinearVariationalSolver(problem,M)
228     #solver = NonlinearVariationalSolver(problem)
229     prm = solver.parameters
230     prm_nonlin=prm["nonlinear_variational_solver"]
231     prm_nonlin["newton_solver"]["absolute_tolerance"]= 1E-8
232     prm_nonlin["newton_solver"]["relative_tolerance"]= 1E-8
233     prm_nonlin["newton_solver"]["relaxation_parameter"]=self.relax_param
234     if(self.relax_param < 0.2):
235      prm_nonlin["newton_solver"]["maximum_iterations"] = 400
236     else:
237      prm_nonlin["newton_solver"]["maximum_iterations"] = 200
238     prm_nonlin["newton_solver"]["error_on_nonconvergence"]=True
239     #prm["newton_solver"]["absolute_tolerance"]= 1E-8
240     #prm["newton_solver"]["relative_tolerance"]= 1E-8
241     #prm["newton_solver"]["maximum_iterations"] = 350
242     #prm["newton_solver"]["relaxation_parameter"]=relax_param
243     #prm["newton_solver"]["error_on_nonconvergence"]=False
244     solver_tolerance=1E-8
245     solved=False
246     while not solved:
247      if(prm_nonlin["newton_solver"]["relaxation_parameter"]<0.05):
248          print('''
249          DID NOT CONVERGE !
250          PARAMETERS:
251          Re          : {:08.4f}
```

```
252             Gamma       : {:08.4f}
253             alpha       : {:08.4f}
254             '''.format(Re, self.Gamma, self.alpha))
255             sys.exit()
256         uvwp.assign(self.uvwp0)
257         try:
258           solver.solve(solver_tolerance)
259         except:
260           prm_nonlin["newton_solver"]["relaxation_parameter"]*=0.8
261           self.relax_param*=0.8
262           s = ">>> WARNING: newton relaxation parameter lowered to %g <<<"
263           print(s % prm_nonlin["newton_solver"]["relaxation_parameter"])
264           continue
265         solved=True
266       self.uvwp0.assign(uvwp)
267       return
268
269
270     def linearstab(self,Re,karr,lplot=False):
271       #THE PERTURBATION-FUNCTIONS
272       uvwp_p = TrialFunction(self.W)
273       uvec_p,p_p = split(uvwp_p)
274       u_p,v_p,w_p = split(uvec_p)
275       #THE BOUNDARY CONDITIONS FOR THE PERTURBATION
276       ubcN_p = DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcN)
277       ubcSEW_p= DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcSEW)
278       bcs_p=[ubcSEW_p,ubcN_p]
279       k=Constant(0.1)
280       sigma=np.zeros(np.size(karr))
281       omega=np.zeros(np.size(karr))
282       for i,ck in enumerate(karr):
283         k.assign(ck)
284         Fp_real= \
285             self.du*self.u0*u_p.dx(0)*dx+self.du*self.v0*u_p.dx(1)*dx+self.du*u_p*
                  self.u0.dx(0)*dx+self.du*v_p*self.u0.dx(1)*dx+\
286             self.du*p_p.dx(0)*dx+self.du.dx(0)*u_p.dx(0)*dx+self.du.dx(1)*u_p.dx(1)*
                  dx+self.du*k**2.*u_p*dx+\
287             self.dv*self.u0*v_p.dx(0)*dx+self.dv*self.v0*v_p.dx(1)*dx+self.dv*u_p*
                  self.v0.dx(0)*dx+self.dv*v_p*self.v0.dx(1)*dx+\
288             self.dv*p_p.dx(1)*dx+self.dv.dx(0)*v_p.dx(0)*dx+self.dv.dx(1)*v_p.dx(1)*
                  dx+self.dv*k**2.*v_p*dx+\
289             self.dw*self.u0*w_p.dx(0)*dx+self.dw*self.v0*w_p.dx(1)*dx+self.dw*u_p*
                  self.w0.dx(0)*dx+self.dw*v_p*self.w0.dx(1)*dx+\
290             self.dw.dx(0)*w_p.dx(0)*dx+self.dw.dx(1)*w_p.dx(1)*dx+self.dw*k**2.*w_p*
                  dx+\
291             self.dp*u_p.dx(0)*dx+self.dp*v_p.dx(1)*dx
292         #
293         #
294         #
295         Fp_imag= \
296                 self.du*self.w0*k*u_p*dx+self.dv*self.w0*k*v_p*dx+self.dw*self.w0*k*
                      w_p*dx+self.dw*k*p_p*dx+self.dp*k*w_p*dx
297         B_MAT = \
298                 self.du*u_p*dx+self.dv*v_p*dx+self.dw*w_p*dx
299         #
300         Ar = PETScMatrix()
301         assemble(Fp_real, tensor=Ar)
302         [bc.apply(Ar) for bc in bcs_p]
```

```
303     Ai = PETScMatrix()
304     assemble(Fp_imag, tensor=Ai)
305     [bc.apply(Ai) for bc in bcs_p]
306     M = PETScMatrix()
307     assemble(B_MAT, tensor=M)
308     #[bc.apply(M) for bc in bcs_p]
309     #
310     bcinds = []
311     for bc in bcs_p:
312      bcdict = bc.get_boundary_values()
313      bcinds.extend(bcdict.keys())
314     # This just converts PETSc to CSR
315     Ar = sp.csr_matrix(Ar.mat().getValuesCSR()[::-1])
316     Ai = sp.csr_matrix(Ai.mat().getValuesCSR()[::-1])
317     M = sp.csr_matrix(M.mat().getValuesCSR()[::-1])
318     # Create shift matrix
319     #shift = 1.2345e10*np.ones(len(bcinds))
320     #S = sp.csr_matrix((shift, (bcinds, bcinds)), shape=Ar.shape)
321     v, V = eigs(Ar+1.j*Ai, 10, M, sigma=-1.)
322     sigma[i],omega[i]=np.sort(v)[0].real,np.sort(v)[0].imag
323     print('Re = {}, k = {} lowest_sigma = {}, omega = {}'.format(Re,ck,sigma[i],
            omega[i]))
324     ##SAVE THE RESULTING FLOW FUNCTIONS
325     egv = Function(self.W)
326     egv_dz = Function(self.W)
327
328     ##index of lowest sigma
329     indx = np.argmin(v)
330     #STORE REAL AND IMAGINARY PART OF THE EIGENVECTOR
331     intsteps=100
332     zarr = np.linspace(0.,2.*np.pi/ck,intsteps)
333     delta_z = zarr[1]-zarr[0]
334     D_int=0.
335     I1_int=0.
336     I2_int=0.
337     I3_int=0.
338     I4_int=0.
339     Isum_int=0.
340     Imax = -1.e99
341     iwhere=-10 #index with maximum Isum
342     for i,z in enumerate(zarr):
343      u_vec = np.real(V[:,indx]*np.exp(1j*(ck*z))+np.conjugate(V[:,indx])*np.exp
            (-1j*(ck*z)))
344      egv.vector().set_local(u_vec)
345      euvw,ep = split(egv)
346      eu,ev,ew = split(euvw)
347      u_vec_dz = np.real((1j*ck)*(V[:,indx]*np.exp(1j*(ck*z))-np.conjugate(V[:,
            indx])*np.exp(-1j*(ck*z))))
348      egv_dz.vector().set_local(u_vec_dz)
349      euvw_dz,ep_dz = split(egv_dz)
350      eu_dz,ev_dz,ew_dz = split(euvw_dz)
351      euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
352      eu_par,ev_par,ew_par = euvw_par
353      euvw_orth = euvw - euvw_par
354      eu_orth,ev_orth,ew_orth = euvw_orth
355      D = ((ew.dx(1)-ev_dz)**2 + (eu_dz-ew.dx(0))**2. + (ev.dx(0)-eu.dx(1))**2)*
            dx
356      D = assemble(D)
```

```
357       I1 = - eu_orth * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
358               - ev_orth * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
359               - ew_orth * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
360       I1 = assemble(I1*dx)
361       I2 = - eu_par * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
362               - ev_par * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
363               - ew_par * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
364       I2 = assemble(I2*dx)
365       I3 = - eu_orth * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
366               - ev_orth * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
367               - ew_orth * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
368       I3 = assemble(I3*dx)
369       I4 = - eu_par * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
370               - ev_par * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
371               - ew_par * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
372       I4 = assemble(I4*dx)
373       Isum = I1+I2+I3+I4
374       if(Isum>Imax):
375        Imax = Isum
376        iwhere = i
377       D_int  += D*delta_z
378       I1_int += I1*delta_z
379       I2_int += I2*delta_z
380       I3_int += I3*delta_z
381       I4_int += I4*delta_z
382      egvfolder = self.calcdir+"/EGV_Re_{:08.4f}_k_{:08.4f}".format(Re,ck)
383      if not os.path.exists(egvfolder):
384       os.mkdir(egvfolder)
385      f2=open(egvfolder+"/ENERGIES",'w')
386      f2.write('# D   I1    I2    I3    I4\n')
387      f2.write('{:.6e} {:.6e} {:.6e} {:.6e} {:.6e} '.format(D_int,I1_int,I2_int,
             I3_int,I4_int))
388      f2.close()
389      if(lplot):
390       #PLOT THE FLOW WITH THE MAXIMAL ENERGY GAIN AND THE ENERGY CONTRIBUTIONS
391       z=zarr[iwhere]
392       u_vec = np.real(V[:,indx]*np.exp(1j*(ck*z))+np.conjugate(V[:,indx])*np.exp
              (-1j*(ck*z)))
393       egv.vector().set_local(u_vec)
394       euvw,ep = split(egv)
395       eu,ev,ew = split(euvw)
396       euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
397       eu_par,ev_par,ew_par = euvw_par
398       euvw_orth = euvw - euvw_par
399       eu_orth,ev_orth,ew_orth = euvw_orth
400       u_vec_dz = np.real((1j*ck)*(V[:,indx]*np.exp(1j*(ck*z))-np.conjugate(V[:,
              indx])*np.exp(-1j*(ck*z))))
401       egv_dz.vector().set_local(u_vec_dz)
402       euvw_dz,ep_dz = split(egv_dz)
403       eu_dz,ev_dz,ew_dz = split(euvw_dz)
404       euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
405       eu_par,ev_par,ew_par = euvw_par
406       euvw_orth = euvw - euvw_par
407       eu_orth,ev_orth,ew_orth = euvw_orth
408       D = (ew.dx(1)-ev_dz)**2 + (eu_dz-ew.dx(0))**2. + (ev.dx(0)-eu.dx(1))**2
409       I1 = - eu_orth * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
410             - ev_orth * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
411             - ew_orth * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
```

```
412         I2 = - eu_par * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
413              - ev_par * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
414              - ew_par * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
415         I3 = - eu_orth * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
416              - ev_orth * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
417              - ew_orth * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
418         I4 = - eu_par * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
419              - ev_par * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
420              - ew_par * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
421         Isum = I1+I2+I3+I4
422         IDsum = Isum + D
423         ####Create files for storing solution
424         V2 = FunctionSpace(self.mesh, self.P1)
425         for i,en in zip(['D','I1','I2','I3','I4','Isum','IDsum'],[D,I1,I2,I3,I4,
                Isum,IDsum]):
426          filename=egvfolder+"/"+i+".pvd"
427          Ifile=File(filename)
428          temp = project(en,V2)
429          temp.rename(i,"")
430          Ifile << temp
431         ###RENAME THE FIELD TO BE BE NAMED vel AND p
432         ufilename = egvfolder+"/u.pvd"
433         ufile = File(ufilename)
434         egv.rename("vel","")
435         ufile << egv.sub(0)
436         pfilename=egvfolder+"/p.pvd"
437         pfile = File(pfilename)
438         egv.rename("pressure","")
439         pfile << egv.sub(1)
440      #########SAVING DONE
441      return sigma,omega
442
443     def linearstab_MORE_EV(self,Re,karr,nr_EV,lplot=False,enanalysis=False):
444      #THE PERTURBATION-FUNCTIONS
445      uvwp_p = TrialFunction(self.W)
446      uvec_p,p_p = split(uvwp_p)
447      u_p,v_p,w_p = split(uvec_p)
448      #THE BOUNDARY CONDITIONS FOR THE PERTURBATION
449      ubcN_p = DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcN)
450      ubcSEW_p= DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcSEW)
451      bcs_p=[ubcSEW_p,ubcN_p]
452      k=Constant(0.1)
453      sigma_low=np.zeros(np.size(karr))
454      omega_low=np.zeros(np.size(karr))
455      sigma=np.zeros([np.size(karr),nr_EV])
456      omega=np.zeros([np.size(karr),nr_EV])
457      for i,ck in enumerate(karr):
458       k.assign(ck)
459       Fp_real= \
460           self.du*self.u0*u_p.dx(0)*dx+self.du*self.v0*u_p.dx(1)*dx+self.du*u_p*
                self.u0.dx(0)*dx+self.du*v_p*self.u0.dx(1)*dx+\
461           self.du*p_p.dx(0)*dx+self.du.dx(0)*u_p.dx(0)*dx+self.du.dx(1)*u_p.dx(1)*
                dx+self.du*k**2.*u_p*dx+\
462           self.dv*self.u0*v_p.dx(0)*dx+self.dv*self.v0*v_p.dx(1)*dx+self.dv*u_p*
                self.v0.dx(0)*dx+self.dv*v_p*self.v0.dx(1)*dx+\
463           self.dv*p_p.dx(1)*dx+self.dv.dx(0)*v_p.dx(0)*dx+self.dv.dx(1)*v_p.dx(1)*
                dx+self.dv*k**2.*v_p*dx+\
```

```
464         self.dw*self.u0*w_p.dx(0)*dx+self.dw*self.v0*w_p.dx(1)*dx+self.dw*u_p*
                self.w0.dx(0)*dx+self.dw*v_p*self.w0.dx(1)*dx+\
465         self.dw.dx(0)*w_p.dx(0)*dx+self.dw.dx(1)*w_p.dx(1)*dx+self.dw*k**2.*w_p*
                dx+\
466         self.dp*u_p.dx(0)*dx+self.dp*v_p.dx(1)*dx
467     #
468     #
469     #
470     Fp_imag= \
471             self.du*self.w0*k*u_p*dx+self.dv*self.w0*k*v_p*dx+self.dw*self.w0*k*
                w_p*dx+self.dw*k*p_p*dx+self.dp*k*w_p*dx
472     B_MAT = \
473             self.du*u_p*dx+self.dv*v_p*dx+self.dw*w_p*dx
474     #
475     Ar = PETScMatrix()
476     assemble(Fp_real, tensor=Ar)
477     [bc.apply(Ar) for bc in bcs_p]
478     Ai = PETScMatrix()
479     assemble(Fp_imag, tensor=Ai)
480     [bc.apply(Ai) for bc in bcs_p]
481     M = PETScMatrix()
482     assemble(B_MAT, tensor=M)
483     #[bc.apply(M) for bc in bcs_p]
484     #
485     bcinds = []
486     for bc in bcs_p:
487      bcdict = bc.get_boundary_values()
488      bcinds.extend(bcdict.keys())
489     # This just converts PETSc to CSR
490     Ar = sp.csr_matrix(Ar.mat().getValuesCSR()[::-1])
491     Ai = sp.csr_matrix(Ai.mat().getValuesCSR()[::-1])
492     M = sp.csr_matrix(M.mat().getValuesCSR()[::-1])
493     # Create shift matrix
494     #shift = 1.2345e10*np.ones(len(bcinds))
495     #S = sp.csr_matrix((shift, (bcinds, bcinds)), shape=Ar.shape)
496     if(nr_EV<10):
497      v, V = eigs(Ar+1.j*Ai, 10, M, sigma=-10)
498     else:
499      v, V = eigs(Ar+1.j*Ai, 10, M, sigma=-10)
500     sigma_low[i],omega_low[i]=np.sort(v)[0].real,np.sort(v)[0].imag
501     print('Re = {}, k = {} lowest_sigma = {}, omega = {}'.format(Re,ck,sigma_low
            [i],omega_low[i]))
502     print('OTHER EIGENVALS : \n')
503     #ARRAY FOR A MAPPING OF SIGMA
504     maparr=np.argsort(v)
505     for ij in range(nr_EV):
506      sigma[i,ij],omega[i,ij]=np.sort(v)[ij].real,np.sort(v)[ij].imag
507      print('sigma = {}      , omega = {}   \n'.format(sigma[i,ij],omega[i,ij]))
508     if(enanalysis):
509      ##SAVE THE RESULTING FLOW FUNCTIONS
510      egv = Function(self.W)
511      egv_dz = Function(self.W)
512
513      ##index of lowest sigma
514      for isigma in range(nr_EV):
515       indx = maparr[isigma]
516       #STORE REAL AND IMAGINARY PART OF THE EIGENVECTOR
517       intsteps=100
```

```
518         zarr = np.linspace(0.,2.*np.pi/ck,intsteps)
519         delta_z = zarr[1]-zarr[0]
520         D_int=0.
521         I1_int=0.
522         I2_int=0.
523         I3_int=0.
524         I4_int=0.
525         Isum_int=0.
526         Imax = -1.e99
527         iwhere=-10 #index with maximum Isum
528         for i,z in enumerate(zarr):
529          u_vec = np.real(V[:,indx]*np.exp(1j*(ck*z))+np.conjugate(V[:,indx])*np.
                 exp(-1j*(ck*z)))
530          egv.vector().set_local(u_vec)
531          euvw,ep = split(egv)
532          eu,ev,ew = split(euvw)
533          u_vec_dz = np.real((1j*ck)*(V[:,indx]*np.exp(1j*(ck*z))-np.conjugate(V[:,
                 indx])*np.exp(-1j*(ck*z))))
534          egv_dz.vector().set_local(u_vec_dz)
535          euvw_dz,ep_dz = split(egv_dz)
536          eu_dz,ev_dz,ew_dz = split(euvw_dz)
537          euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
538          eu_par,ev_par,ew_par = euvw_par
539          euvw_orth = euvw - euvw_par
540          eu_orth,ev_orth,ew_orth = euvw_orth
541          D = ((ew.dx(1)-ev_dz)**2 + (eu_dz-ew.dx(0))**2. + (ev.dx(0)-eu.dx(1))**2)
                 *dx
542          D = assemble(D)
543          I1 = - eu_orth * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
544                  - ev_orth * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
545                  - ew_orth * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
546          I1 = assemble(I1*dx)
547          I2 = - eu_par * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
548                  - ev_par * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
549                  - ew_par * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
550          I2 = assemble(I2*dx)
551          I3 = - eu_orth * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
552                  - ev_orth * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
553                  - ew_orth * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
554          I3 = assemble(I3*dx)
555          I4 = - eu_par * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
556                  - ev_par * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
557                  - ew_par * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
558          I4 = assemble(I4*dx)
559          Isum = I1+I2+I3+I4
560          if(Isum>Imax):
561           Imax = Isum
562           iwhere = i
563          D_int  += D*delta_z
564          I1_int += I1*delta_z
565          I2_int += I2*delta_z
566          I3_int += I3*delta_z
567          I4_int += I4*delta_z
568         egvfolder = self.calcdir+"/EGV_Re_{:08.4f}_k_{:08.4f}_EV_{:03d}".format(Re
                 ,ck,isigma)
569         if not os.path.exists(egvfolder):
570          os.mkdir(egvfolder)
571         f2=open(egvfolder+"/ENERGIES".format(isigma),'w')
```

```
572         f2.write('# D   I1    I2    I3    I4\n')
573         f2.write('{:.6e} {:.6e} {:.6e} {:.6e} {:.6e} '.format(D_int,I1_int,I2_int,
                I3_int,I4_int))
574         f2.close()
575         if(lplot):
576          #PLOT THE FLOW WITH THE MAXIMAL ENERGY GAIN AND THE ENERGY CONTRIBUTIONS
577          z=zarr[iwhere]
578          u_vec = np.real(V[:,indx]*np.exp(1j*(ck*z))+np.conjugate(V[:,indx])*np.
                 exp(-1j*(ck*z)))
579          egv.vector().set_local(u_vec)
580          euvw,ep = split(egv)
581          eu,ev,ew = split(euvw)
582          euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
583          eu_par,ev_par,ew_par = euvw_par
584          euvw_orth = euvw - euvw_par
585          eu_orth,ev_orth,ew_orth = euvw_orth
586          u_vec_dz = np.real((1j*ck)*(V[:,indx]*np.exp(1j*(ck*z))-np.conjugate(V[:,
                 indx])*np.exp(-1j*(ck*z))))
587          egv_dz.vector().set_local(u_vec_dz)
588          euvw_dz,ep_dz = split(egv_dz)
589          eu_dz,ev_dz,ew_dz = split(euvw_dz)
590          euvw_par = dot(euvw,self.uvw0)*self.uvw0 /(dot(self.uvw0,self.uvw0))
591          eu_par,ev_par,ew_par = euvw_par
592          euvw_orth = euvw - euvw_par
593          eu_orth,ev_orth,ew_orth = euvw_orth
594          D = (ew.dx(1)-ev_dz)**2 + (eu_dz-ew.dx(0))**2. + (ev.dx(0)-eu.dx(1))**2
595          I1 = - eu_orth * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
596               - ev_orth * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
597               - ew_orth * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
598          I2 = - eu_par * (eu_orth * self.u0.dx(0) + ev_orth * self.u0.dx(1)) \
599               - ev_par * (eu_orth * self.v0.dx(0) + ev_orth * self.v0.dx(1)) \
600               - ew_par * (eu_orth * self.w0.dx(0) + ev_orth * self.w0.dx(1))
601          I3 = - eu_orth * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
602               - ev_orth * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
603               - ew_orth * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
604          I4 = - eu_par * (eu_par * self.u0.dx(0) + ev_par * self.u0.dx(1)) \
605               - ev_par * (eu_par * self.v0.dx(0) + ev_par * self.v0.dx(1)) \
606               - ew_par * (eu_par * self.w0.dx(0) + ev_par * self.w0.dx(1))
607          Isum = I1+I2+I3+I4
608          IDsum = Isum + D
609          ####Create files for storing solution
610          V2 = FunctionSpace(self.mesh,self.P1)
611          for i,en in zip(['D','I1','I2','I3','I4','Isum','IDsum'],[D,I1,I2,I3,I4,
                 Isum,IDsum]):
612           filename=egvfolder+"/"+i+".pvd"
613           Ifile=File(filename)
614           temp = project(en,V2)
615           temp.rename(i,"")
616           Ifile << temp
617          ###RENAME THE FIELD TO BE BE NAMED vel AND p
618          ufilename = egvfolder+"/u.pvd"
619          ufile = File(ufilename)
620          egv.rename("vel","")
621          ufile << egv.sub(0)
622          pfilename=egvfolder+"/p.pvd"
623          pfile = File(pfilename)
624          egv.rename("pressure","")
625          pfile << egv.sub(1)
```

```
626     #########SAVING DONE
627     return sigma_low,omega_low,sigma,omega
628
629
630
631   def mymeshrefine(self):
632     #parameters["num_threads"] = 1
633     parameters["mesh_partitioner"]='ParMETIS'
634     #REFINE CLOSE TO THE BORDER
635     #TRY 3 REFINEMENTS IN THE BEGINNING
636     #@30% 15% 5% of the box
637     dist_pc=np.array([0.3,0.15,0.05])
638     distx_left = -1./2.+dist_pc*1.
639     distx_right = 1./2.-dist_pc*1.
640     disty_bottom = -self.Gamma/2.+dist_pc*self.Gamma
641     disty_top = self.Gamma/2.-dist_pc*self.Gamma
642     for i in range(len(dist_pc)):
643       cell_markers = CellFunction("bool",self.mesh)
644       cell_markers.set_all(False)
645       for cell in cells(self.mesh):
646         p = cell.midpoint()
647           if ((p.x()<distx_left[i])    or
648               (p.x()>distx_right[i])   or
649               (p.y()<disty_bottom[i]) or
650               (p.y()>disty_top[i])):
651               cell_markers[cell]=True
652     self.mesh=refine(self.mesh,cell_markers)
653   def getsignchange(self,inarr):
654     nr=0
655     change=[]
656     indices=[]
657     for i in range(np.size(inarr)-1):
658       if(inarr[i]>=0 and inarr[i+1]<0):
659         nr+=1
660         change.append(1)
661         indices.append(i)
662       elif(inarr[i]<=0 and inarr[i+1]>0):
663         nr+=1
664         change.append(-1)
665         indices.append(i)
666     return nr,np.array(change),np.array(indices)
667   def find_nearest(self,array, value):
668     array = np.asarray(array)
669     idx = (np.abs(array - value)).argmin()
670     return idx,array[idx]
671   def plot_k_sigma(self,Re,filename):
672     plotfile=open(filename,'r')
673     data=plotfile.readlines()
674     data_arr=np.zeros([len(data),2])
675     for i,line in enumerate(data):
676       data_arr[i,0],data_arr[i,1]=float(line.split()[0]),float(line.split()[1])
677     #sort the array
678     temp = np.argsort(data_arr,0)[:,0]
679     sorted_arr=data_arr[temp]
680     #SET MATPLOTLIB TeX
681     plt.rc('font',**{'family':'serif','serif':['Times']})
682     plt.rc('text',usetex=True)
683     def cm2inch(value):
```

```
684       return value/2.54
685     fig,ax = plt.subplots(1,figsize=(cm2inch(13.8),cm2inch(7.0)))
686     ax.plot(sorted_arr[:,0],sorted_arr[:,1],c='r',linestyle='-')
687     ax.grid(True,which='both',ls="--",lw=0.15)
688     ax.set_title('Stability for Re = {}'.format(Re))
689     ax.title.set_weight('bold')
690     ax.set_xlim([0,10.])
691     ax.set_xlabel(r"k",labelpad=1.)
692     ax.set_ylabel(r"$\sigma$",labelpad=1.)
693     savename=self.calcdir+'/'+'Re_{:07.1f}.pdf'.format(Re)
694     plt.savefig(savename,format='pdf',dpi=fig.dpi)
695
696   def readdataset(self,Gamma,angle,Re):
697     folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
698     filename=folder+'/'+'Re'+'_{:08.3f}'.format(Re)
699     Refile=open(filename,'r')
700     lines=Refile.readlines()
701     karr=np.zeros(len(lines))
702     sigma_arr=np.zeros(len(lines))
703     omega_arr=np.zeros(len(lines))
704     for i,line in enumerate(lines):
705       karr[i] = float(line.split()[0])
706       sigma_arr[i] = float(line.split()[1])
707       omega_arr[i] = float(line.split()[2])
708     return karr,sigma_arr,omega_arr
709
710   def get_kneutral(self,Gamma,alpha,Re):
711     karr,sigma_arr,omega_arr=self.readdataset(Gamma,alpha,Re)
712     nr,changearr,indices=self.getsignchange(sigma_arr)
713     if(nr == 0):
714       return 0
715     else:
716       return karr[indices],sigma_arr[indices],omega_arr[indices]
717 #A FUNCTION WHERE WE COMPARE DIFFERENT CURVES
718 #plotarr has to be a dictionairy that provides the Gamma,Re,angle values
719   def compareplot(self,plotarr,show=True):
720     #SET MATPLOTLIB TeX
721     plt.rc('font',**{'family':'serif','serif':['Times']})
722     plt.rc('text',usetex=True)
723     def cm2inch(value):
724       return value/2.54
725     fig,ax = plt.subplots(1,figsize=(cm2inch(13.8),cm2inch(7.0)))
726     ax.grid(True,which='both',ls="--",lw=0.15)
727     colors = ['#0000ff','#ff0000','#009933','#cc0099','#ff9900','#0099cc',\
728               '#009999']
729     for i in range(len(plotarr)):
730       Gamma=plotarr[i]['Gamma']
731       Re =  plotarr[i]['Re']
732       angle = plotarr[i]['angle']
733       label = r'$\alpha={:5.2f} \Gamma={:5.2f} Re={:8.2f}$'.format(angle,Gamma,Re)
734       karr,sigma_arr,omega_arr=self.readdataset(Gamma,angle,Re)
735       #SORT THE ARRAYS
736       sigma_arr=sigma_arr[np.argsort(karr)]
737       omega_arr=omega_arr[np.argsort(karr)]
738       karr = np.sort(karr)
739       ax.plot(karr,sigma_arr,c=colors[i],linestyle='-',label=label)
740     ax.set_xlabel(r"k",labelpad=1.)
741     ax.set_ylabel(r"$\sigma$",labelpad=1.)
```

```
    ax.legend()
    if(show):
     plt.show()
    else:
     plt.savefig('TEMPPLOT.pdf',format='pdf',dpi=fig.dpi)
  def analyze_k_n(self,Re,relax_param,k1,k2,ngrid):
    print('RUNNING ANALYZE_K WITH REYNOLDS: {} \n k1 = {}   k2 = {}'.format(Re,k1,
        k2))
    filename=self.calcdir+'/'+'Re_k'+'_{:08.3f}'.format(Re)
    Refile=open(filename,'w')
    karr=np.linspace(k1,k2,3)
    nx=ngrid
    ny=ngrid
    self.getbaseflow(Re,nx,ny)
    sigma,omega=self.linearstab(Re,karr)
    for line in range(len(sigma)):
     Refile.write('{:10g}     {:10g}     {:10g} \n'.format(karr[line],sigma[line],
         omega[line]))
    nr,changearr,indices=self.getsignchange(sigma)
    Refile.close()

def gamma_angle_Re(gammaarr,anglearr,Rearr,relax_param_arr):
 for Gamma in gammaarr:
  for angle in anglearr:
   print('''
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   CALCULATING GAMMA = {:07.3f} AND ANGLE = {:05.1f}
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'''.format(Gamma,angle))
   folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
   cav=mycav(Gamma,angle,calcdir=folder)
   for Re,relax_param in zip(Rearr,relax_param_arr):
    cav.analyze(Re,relax_param)

def RcAnalysis(gammaarr,anglearr,Reminarr,Remaxarr):
 for Gamma in gammaarr:
  for angle in anglearr:
   print('''
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ReC analysis: GAMMA = {:07.3f} AND ANGLE = {:05.1f}
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'''.format(Gamma,angle))
   folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
   cav=mycav(Gamma,angle,calcdir=folder)
   Re_low=Reminarr[0]
   Re_high=Remaxarr[0]
   cav.findRe_c(Re_low,Re_high)
def RcAnalysis_MORE_EV(gammaarr,anglearr,Reminarr,Remaxarr,nr_EV_arr,kmin,kmax,
     nr_k):
 for Gamma in gammaarr:
  for angle in anglearr:
   print('''
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ReC analysis: GAMMA = {:07.3f} AND ANGLE = {:05.1f}
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'''.format(Gamma,angle))
   folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
   cav=mycav(Gamma,angle,calcdir=folder)
   Re_low=Reminarr[0]
   Re_high=Remaxarr[0]
   nr_EV=nr_EV_arr[0]
```

```
797     cav.findRe_c_MORE_EV(Re_low,Re_high,nr_EV,kmin,kmax,nr_k)
798
799 if __name__=='__main__':
800  print('HELLO')
```

D.1.2 Plotting script

```
1  from fenics import *
2  import matplotlib.pyplot as plt
3  import scipy.sparse as sp
4  from scipy.sparse.linalg import eigs
5  import numpy as np
6  import os
7  import sys
8
9  class mycav:
10     def __init__(self,Gamma,alpha,calcdir=None):
11         self.Gamma=Gamma
12         self.alpha=alpha
13         if (calcdir!=None):
14             self.calcdir=calcdir
15             if not os.path.exists(calcdir):
16                 os.mkdir(calcdir)
17     def analyze(self,Re,relax_param):
18         print('RUNNING ANALYZE WITH REYNOLDS: {}'.format(Re))
19         print('1st RUN --> coarse GRID')
20         filename=self.calcdir+'/'+'Re'+'_{:08.3f}'.format(Re)
21         Refile=open(filename,'w')
22         karr=np.linspace(0.001,30,30)
23         nx=70
24         ny=70
25         self.getbaseflow(Re,nx,ny,relax_param)
26         sigma,omega=self.linearstab(Re,karr)
27         for line in range(len(sigma)):
28             Refile.write('{:10g}     {:10g}     {:10g} \n'.format(karr[line],
                   sigma[line],omega[line]))
29         nr,changearr,indices=self.getsignchange(sigma)
30         print('FOUND {} SIGN CHANGES'.format(nr))
31         #1st REFINEMENT OF THE MESH FOR THE CRITICAL k-VALUES
32         if(nr>0):
33             nx=75
34             ny=75
35             self.getbaseflow(Re,nx,ny,relax_param)
36             for i in indices:
37                 k1=karr[i];k2=karr[i+1]
38                 conv=100
39                 kdiff=10
40                 run=0
41                 while(conv>1e-3 and kdiff > 0.1):
42                     run+=1
43                     print('LOOKING IN THE VICINITY OF k= {}'.format(k1))
44                     k_refine=np.linspace(k1,k2,5)
45                     sigma2,omega2=self.linearstab(Re,k_refine)
46                     nr2,changearr2,indices2=self.getsignchange(sigma2)
47                     if(nr2==0 and run>1):
48                         k1=karr[i-1]
49                         k2=karr[i+2]
50                     elif(nr2==0 and run==1):
```

```
51                          break
52                      else:
53                          k1=k_refine[indices2[0]]
54                          k2=k_refine[indices2[0]+1]
55                      kdiff=abs(k2-k1)
56                      conv=abs(self.find_nearest(sigma2,0)[1])
57                      for line in range(len(sigma2)):
58                          Refile.write('{:10g}    {:10g}    {:10g} \n'.format(
                                k_refine[line],sigma2[line],omega2[line]))
59                  print('SIGMA(k_crit) before 2nd REFINEMENT: {}'.format(sigma2[
                        indices2[0]]))
60                  #EXTRACT THE VALUE CLOSEST TO 0
61                  kchoose=k_refine[self.find_nearest(sigma2,0)[0]]
62                  #2nd REFINEMENT TO CHECK FOR CONVERGENCE
63                  nx=85
64                  ny=85
65                  self.getbaseflow(Re,nx,ny,relax_param)
66                  sigma2,omega2=self.linearstab(Re,np.asarray([kchoose]))
67                  print('SIGMA(k_crit) after 2nd REFINEMENT: {}'.format(sigma2
                        [0]))
68                  Refile.write('{:10g}    {:10g}    {:10g} \n'.format(kchoose,
                        sigma2[0],omega2[0]))
69          Refile.close()
70          self.plot_k_sigma(Re,filename)
71
72      def findRe_c(self,Re_low,Re_high):
73          resultname=self.calcdir+'/'+'REc_Gamma_{:5.2f}_alpha_{:5.2f}'.format(
                self.Gamma,self.alpha)
74          Resultfile=open(resultname,'w')
75          karr=np.linspace(0.001,30,30)
76          nx=75
77          ny=75
78          Rearr=np.array([Re_low,Re_high])
79          Remid=(Re_low+Re_high)/2.
80          while (np.min(abs(Rearr-Remid))>1.):
81              print('RUNNING findRe_c WITH REYNOLDS: {}'.format(Remid))
82              filename=self.calcdir+'/'+'Re_crit'+'_{:08.3f}'.format(Remid)
83              Refile=open(filename,'w')
84              if(Remid<500):
85                  relax_param=0.5
86              elif(Remid>=500 and Remid<1200):
87                  relax_param=0.3
88              else:
89                  relax_param=0.1
90              self.getbaseflow(Remid,nx,ny,relax_param)
91              sigma,omega=self.linearstab(Remid,karr)
92              for line in range(len(sigma)):
93                  Refile.write('{:10g}    {:10g}    {:10g} \n'.format(karr[line],
                        sigma[line],omega[line]))
94              nr,changearr,indices=self.getsignchange(sigma)
95              Resultfile.write('{:7.2f}    {:2d} \n'.format(Remid,nr))
96              print('FOUND {} SIGN CHANGES'.format(nr))
97              if(nr>0):
98                  Rearr[1]=Remid
99              elif(nr==0):
100                 Rearr[0]=Remid
101             Remid=(Rearr[0]+Rearr[1])/2.
102             Refile.close()
```

```
103             Resultfile.close()
104         def getbaseflow(self,Re,nx,ny,relax_param):
105             #CREATE THE MESH
106             self.mesh=RectangleMesh(Point(-self.Gamma/2.,-0.5),Point(self.Gamma
                    /2.,0.5),nx,ny)
107             #REFINE THE MESH HERE
108             self.mymeshrefine()
109             #DEFINE TRIAL AND TEST FUNCTIONS
110             self.P2 = VectorElement('P',triangle,degree=2,dim=3)
111             self.P1 = FiniteElement('P',triangle,degree=1)
112             self.element = MixedElement([self.P2,self.P1])
113             self.W = FunctionSpace(self.mesh,self.element)
114             self.duvw,self.dp = TestFunctions(self.W)
115             self.du,self.dv,self.dw = split(self.duvw)
116             #THE FUNCTIONS FOR THE FORMULATION
117             uvwp = Function(self.W)
118             uvw,p = split(uvwp)
119             u,v,w = split(uvw)
120             #THE FUNCTIONS FOR THE STEADY STATE SOLUTION
121             self.uvwp0 = Function(self.W)
122             self.uvw0,p0 = split(self.uvwp0)
123             self.u0,self.v0,self.w0 = split(self.uvw0)
124             Gamma=self.Gamma
125             tol=1e-14
126             #DEFINE THE BOUNDARIES
127             class BoundN(SubDomain):
128                 def inside(self,x,on_boundary):
129                     return on_boundary and near(x[1],0.5,tol)
130             class BoundSEW(SubDomain):
131                 def inside(self,x,on_boundary):
132                     return on_boundary and (near(x[1],-0.5,tol) or
133                         near(x[0],-Gamma/2.,tol)
134                         or near(x[0],Gamma/2.,tol))
135             self.bcN=BoundN()
136             self.bcSEW=BoundSEW()
137             ubcN = DirichletBC(self.W.sub(0),Constant((Re*np.cos(self.alpha/180.*np
                    .pi),0.,Re*np.sin(self.alpha/180.*np.pi))),self.bcN)
138             ubcSEW= DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcSEW)
139             bcs=[ubcSEW,ubcN]
140 
141             ###NS-equation
142             F = self.du*u*u.dx(0)*dx+self.du*v*u.dx(1)*dx+inner(grad(self.du),grad(
                    u))*dx+self.du*p.dx(0)*dx +\
143                 self.dv*u*v.dx(0)*dx+self.dv*v*v.dx(1)*dx+inner(grad(self.dv),grad(
                        v))*dx+self.dv*p.dx(1)*dx +\
144                 self.dw*u*w.dx(0)*dx+self.dw*v*w.dx(1)*dx+inner(grad(self.dw),grad(
                        w))*dx +\
145                 self.dp*u.dx(0)*dx+self.dp*v.dx(1)*dx
146 
147 
148             J=derivative(F,uvwp)
149             M=u.dx(0)*dx+v.dx(1)*dx
150 
151             problem = NonlinearVariationalProblem(F, uvwp, bcs, J=J)
152             solver = AdaptiveNonlinearVariationalSolver(problem,M)
153             #solver = NonlinearVariationalSolver(problem)
154             prm = solver.parameters
```

```
155         prm["nonlinear_variational_solver"]["newton_solver"]["
                absolute_tolerance"]= 1E-8
156         prm["nonlinear_variational_solver"]["newton_solver"]["
                relative_tolerance"]= 1E-8
157         prm["nonlinear_variational_solver"]["newton_solver"]["
                maximum_iterations"] = 350
158         prm["nonlinear_variational_solver"]["newton_solver"]["
                relaxation_parameter"]=relax_param
159         solver_tolerance=1E-8
160         solver.solve(solver_tolerance)
161         self.uvwp0.assign(uvwp)
162
163
164
165     def linearstab(self,Re,karr):
166         #THE PERTURBATION-FUNCTIONS
167         uvwp_p = TrialFunction(self.W)
168         uvec_p,p_p = split(uvwp_p)
169         u_p,v_p,w_p = split(uvec_p)
170         #THE BOUNDARY CONDITIONS FOR THE PERTURBATION
171         ubcN_p = DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcN)
172         ubcSEW_p= DirichletBC(self.W.sub(0),Constant((0.,0.,0.)),self.bcSEW)
173         bcs_p=[ubcSEW_p,ubcN_p]
174         k=Constant(0.1)
175         sigma=np.zeros(np.size(karr))
176         omega=np.zeros(np.size(karr))
177         for i,ck in enumerate(karr):
178             k.assign(ck)
179             Fp_real= \
180                 self.du*self.u0*u_p.dx(0)*dx+self.du*self.v0*u_p.dx(1)*dx+self.
                        du*u_p*self.u0.dx(0)*dx+self.du*v_p*self.u0.dx(1)*dx+\
181                 self.du*p_p.dx(0)*dx+self.du.dx(0)*u_p.dx(0)*dx+self.du.dx(1)*
                        u_p.dx(1)*dx+self.du*k**2.*u_p*dx+\
182                 self.dv*self.u0*v_p.dx(0)*dx+self.dv*self.v0*v_p.dx(1)*dx+self.
                        dv*u_p*self.v0.dx(0)*dx+self.dv*v_p*self.v0.dx(1)*dx+\
183                 self.dv*p_p.dx(1)*dx+self.dv.dx(0)*v_p.dx(0)*dx+self.dv.dx(1)*
                        v_p.dx(1)*dx+self.dv*k**2.*v_p*dx+\
184                 self.dw*self.u0*w_p.dx(0)*dx+self.dw*self.v0*w_p.dx(1)*dx+self.
                        dw*u_p*self.w0.dx(0)*dx+self.dw*v_p*self.w0.dx(1)*dx+\
185                 self.dw.dx(0)*w_p.dx(0)*dx+self.dw.dx(1)*w_p.dx(1)*dx+self.dw*k
                        **2.*w_p*dx+\
186                 self.dp*u_p.dx(0)*dx+self.dp*v_p.dx(1)*dx
187             #
188             #
189             #
190             Fp_imag= \
191                     self.du*self.w0*k*u_p*dx+self.dv*self.w0*k*v_p*dx+self.dw*
                            self.w0*k*w_p*dx+self.dw*k*p_p*dx+self.dp*k*w_p*dx
192             B_MAT = \
193                     self.du*u_p*dx+self.dv*v_p*dx+self.dw*w_p*dx
194             #
195             Ar = PETScMatrix()
196             assemble(Fp_real, tensor=Ar)
197             [bc.apply(Ar) for bc in bcs_p]
198             Ai = PETScMatrix()
199             assemble(Fp_imag, tensor=Ai)
200             [bc.apply(Ai) for bc in bcs_p]
201             M = PETScMatrix()
```

```
202             assemble(B_MAT, tensor=M)
203             #[bc.apply(M) for bc in bcs_p]
204             #
205             bcinds = []
206             for bc in bcs_p:
207                 bcdict = bc.get_boundary_values()
208                 bcinds.extend(bcdict.keys())
209             # This just converts PETSc to CSR
210             Ar = sp.csr_matrix(Ar.mat().getValuesCSR()[::-1])
211             Ai = sp.csr_matrix(Ai.mat().getValuesCSR()[::-1])
212             M = sp.csr_matrix(M.mat().getValuesCSR()[::-1])
213             # Create shift matrix
214             #shift = 1.2345e10*np.ones(len(bcinds))
215             #S = sp.csr_matrix((shift, (bcinds, bcinds)), shape=Ar.shape)
216             v, V = eigs(Ar+1.j*Ai, 10, M, sigma=-1.)
217             sigma[i],omega[i]=np.sort(v)[0].real,np.sort(v)[0].imag
218             print('Re = {}, k = {} lowest_sigma = {}, omega = {}'.format(Re,ck,
                    sigma[i],omega[i]))
219         return sigma,omega
220     def mymeshrefine(self):
221         #REFINE CLOSE TO THE BORDER
222         #TRY 3 REFINEMENTS IN THE BEGINNING
223         #@30% 15% 5% of the box
224         dist_pc=np.array([0.1,0.01])
225         distx_left = -self.Gamma/2.+dist_pc*self.Gamma
226         distx_right = self.Gamma/2.-dist_pc*self.Gamma
227         disty_bottom = -1./2.+dist_pc*1.
228         disty_top = 1./2.-dist_pc*1
229         for i in range(len(dist_pc)):
230             cell_markers = CellFunction("bool",self.mesh)
231             cell_markers.set_all(False)
232             for cell in cells(self.mesh):
233                 p = cell.midpoint()
234                 if ((p.x()<distx_left[i])    or
235                     (p.x()>distx_right[i])   or
236                     (p.y()<disty_bottom[i]) or
237                     (p.y()>disty_top[i])):
238                     cell_markers[cell]=True
239             self.mesh=refine(self.mesh,cell_markers)
240     def getsignchange(self,inarr):
241         nr=0
242         change=[]
243         indices=[]
244         for i in range(np.size(inarr)-1):
245             if(inarr[i]>=0 and inarr[i+1]<0):
246                 nr+=1
247                 change.append(1)
248                 indices.append(i)
249             elif(inarr[i]<=0 and inarr[i+1]>0):
250                 nr+=1
251                 change.append(-1)
252                 indices.append(i)
253         return nr,np.array(change),np.array(indices)
254     def find_nearest(self,array, value):
255         array = np.asarray(array)
256         idx = (np.abs(array - value)).argmin()
257         return idx,array[idx]
258     def plot_k_sigma(self,Re,filename):
```

```
259             plotfile=open(filename,'r')
260             data=plotfile.readlines()
261             data_arr=np.zeros([len(data),2])
262             for i,line in enumerate(data):
263                 data_arr[i,0],data_arr[i,1]=float(line.split()[0]),float(line.split
                        ()[1])
264             #sort the array
265             temp = np.argsort(data_arr,0)[:,0]
266             sorted_arr=data_arr[temp]
267             #SET MATPLOTLIB TeX
268             plt.rc('font',**{'family':'serif','serif':['Times']})
269             plt.rc('text',usetex=True)
270             def cm2inch(value):
271                 return value/2.54
272             fig,ax = plt.subplots(1,figsize=(cm2inch(13.8),cm2inch(7.0)))
273             ax.plot(sorted_arr[:,0],sorted_arr[:,1],c='r',linestyle='-')
274             ax.grid(True,which='both',ls="--",lw=0.15)
275             ax.set_title('Stability for Re = {}'.format(Re))
276             ax.title.set_weight('bold')
277             ax.set_xlim([0,10.])
278             ax.set_xlabel(r"k",labelpad=1.)
279             ax.set_ylabel(r"$\sigma$",labelpad=1.)
280             savename=self.calcdir+'/'+'Re_{:07.1f}.pdf'.format(Re)
281             plt.savefig(savename,format='pdf',dpi=fig.dpi)
282
283         def readdataset(self,Gamma,angle,Re):
284             folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
285             filename=folder+'/'+'Re'+'_{:08.3f}'.format(Re)
286             Refile=open(filename,'r')
287             lines=Refile.readlines()
288             karr=np.zeros(len(lines))
289             sigma_arr=np.zeros(len(lines))
290             omega_arr=np.zeros(len(lines))
291             for i,line in enumerate(lines):
292                 karr[i] = float(line.split()[0])
293                 sigma_arr[i] = float(line.split()[1])
294                 omega_arr[i] = float(line.split()[2])
295             return karr,sigma_arr,omega_arr
296
297         def get_kneutral(self,Gamma,alpha,Re):
298             karr,sigma_arr,omega_arr=self.readdataset(Gamma,alpha,Re)
299             nr,changearr,indices=self.getsignchange(sigma_arr)
300             if(nr == 0):
301                 return 0
302             else:
303                 return karr[indices],sigma_arr[indices],omega_arr[indices]
304     #A FUNCTION WHERE WE COMPARE DIFFERENT CURVES
305     #plotarr has to be a dictionairy that provides the Gamma,Re,angle values
306         def compareplot(self,plotarr,show=True):
307             #SET MATPLOTLIB TeX
308             plt.rc('font',**{'family':'serif','serif':['Times']})
309             plt.rc('text',usetex=True)
310             def cm2inch(value):
311                 return value/2.54
312             fig,ax = plt.subplots(1,figsize=(cm2inch(13.8),cm2inch(7.0)))
313             ax.grid(True,which='both',ls="--",lw=0.15)
314             colors = ['#0000ff','#ff0000','#009933','#cc0099','#ff9900','#0099cc',\
315                       '#009999']
```

```
316             for i in range(len(plotarr)):
317                 Gamma=plotarr[i]['Gamma']
318                 Re =  plotarr[i]['Re']
319                 angle = plotarr[i]['angle']
320                 label = r'$\alpha={:5.2f} \Gamma={:5.2f} Re={:8.2f}$'.format(angle,
                        Gamma,Re)
321                 karr,sigma_arr,omega_arr=self.readdataset(Gamma,angle,Re)
322                 #SORT THE ARRAYS
323                 sigma_arr=sigma_arr[np.argsort(karr)]
324                 omega_arr=omega_arr[np.argsort(karr)]
325                 karr = np.sort(karr)
326                 ax.plot(karr,sigma_arr,c=colors[i],linestyle='-',label=label)
327             ax.set_xlabel(r"k",labelpad=1.)
328             ax.set_ylabel(r"$\sigma$",labelpad=1.)
329             ax.legend()
330             if(show):
331                 plt.show()
332             else:
333                 plt.savefig('TEMPPLOT.pdf',format='pdf',dpi=fig.dpi)
334         def analyze_k_n(self,Re,relax_param,k1,k2,ngrid):
335             print('RUNNING ANALYZE_K WITH REYNOLDS: {} \n k1 = {}  k2 = {}'.format(
                    Re,k1,k2))
336             filename=self.calcdir+'/'+'Re_k'+'_{:08.3f}'.format(Re)
337             Refile=open(filename,'w')
338             karr=np.linspace(k1,k2,3)
339             nx=ngrid
340             ny=ngrid
341             self.getbaseflow(Re,nx,ny,relax_param)
342             sigma,omega=self.linearstab(Re,karr)
343             for line in range(len(sigma)):
344                 Refile.write('{:10g}     {:10g}     {:10g} \n'.format(karr[line],
                        sigma[line],omega[line]))
345             nr,changearr,indices=self.getsignchange(sigma)
346             Refile.close()
347
348     def gamma_angle_Re(gammaarr,anglearr,Rearr,relax_param_arr):
349         for Gamma in gammaarr:
350             for angle in anglearr:
351                 print('''
352                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
353                 CALCULATING GAMMA = {:07.3f} AND ANGLE = {:05.1f}
354                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'''.format(Gamma,angle)
                        )
355                 folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
356                 cav=mycav(Gamma,angle,calcdir=folder)
357                 for Re,relax_param in zip(Rearr,relax_param_arr):
358                     cav.analyze(Re,relax_param)
359
360     def RcAnalysis(gammaarr,anglearr):
361         for Gamma in gammaarr:
362             for angle in anglearr:
363                 print('''
364                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
365                 ReC analysis: GAMMA = {:07.3f} AND ANGLE = {:05.1f}
366                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'''.format(Gamma,angle)
                        )
367                 folder='Gamma_{:07.3f}_alpha_{:05.1f}'.format(Gamma,angle)
368                 cav=mycav(Gamma,angle,calcdir=folder)
```

```
369                 Re_low=300
370                 Re_high=900
371                 cav.findRe_c(Re_low,Re_high)
372
373 #TODO: WRITE A FUNCTION TO SAVE THE VELOCITY-FIELD
374
375 if __name__=='__main__':
376     #Gammaarr=np.array([1.,0.5,2.,3.])
377     #anglearr=np.array([0.,22.5,45,67.5])
378     #Rearr = [700,750,800,850,900,950,1000]
379     #Rearr2 = [710,720,730,740,760,770,780,790,810,820,830,840]
380     ##gamma_angle_Re(Gammaarr, anglearr, Rearr2)
381     RcAnalysis(Gammaarr,anglearr)
```

Bibliography

Albensoeder, S., H. C. Kuhlmann, and H. J. Rath (2001). "Three-dimensional centrifugal-flow instabilities in the lid-driven-cavity problem". In: *Physics of Fluids* 13.1, pp. 121–135.

Alnæs, Martin et al. (2015). "The FEniCS Project Version 1.5". eng. In:

Botella, O. and R. Peyret (1998). "Benchmark spectral results on the lid-driven cavity flow". In: *Comp. Fluids*, pp. 421–433.

Braun, S. (2001). *Strömungslehre für TPh.*

Childs, Hank et al. (2012). "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data". In: *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pp. 357–372.

Deng, G.B et al. (1994). "Incompressible flow calculations with a consistent physical interpolation finite volume approach". In: *Computers & Fluids* 23.8, pp. 1029–1047.

Ding, Yan and Mutsuto Kawahara (1999). "Three-dimensional linear stability analysis of incompressible viscous flows using the finite element method". In: *International Journal for Numerical Methods in Fluids* 31.2, pp. 451–479.

Feynman R. Leighton R. B., Sands M. (1964). *The Feynman lectures on physics.*

Ghia, U, K.N Ghia, and C.T Shin (1982). "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method". In: *Journal of Computational Physics* 48.3, pp. 387–411.

Gupta, Murli M., Ram P. Manohar, and Ben Noble (1981). "Nature of viscous flows near sharp corners". In: *Computers & Fluids* 9.4, pp. 379–388.

Hancock, C., E. Lewis, and H. K. Moffatt (1981). "Effects of inertia in forced corner flows". In: *Journal of Fluid Mechanics* 112.-1, p. 315.

Kuhlmann, H. (2010). *Grundlagen der numerischen Methoden der Strömungs- und Wärmetechnik.*

— (2012). *Hydrodynamische Stabilität - Skriptum.*

Moffatt, H. K. (1964). "Viscous and resistive eddies near a sharp corner". In: *Journal of Fluid Mechanics* 18.01, p. 1.

Paul F. Fischer, James W. Lottes and Stefan G. Kerkemeier (2008). *nek5000 Web page.* http://nek5000.mcs.anl.gov.

Theofilis, V., P. W. Duck, and J. Owen (2004). "Viscous linear stability analysis of rectangular duct and cavity flows". In: *Journal of Fluid Mechanics* 505, pp. 249–286.