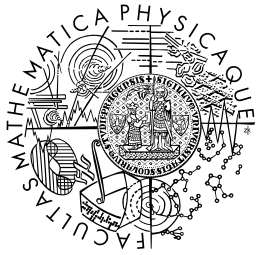

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Martin Jurček

Numerical modeling of compressible flow using spectral element method

Mathematical Institute of Charles University

Supervisor of the master thesis:	prof. RNDr. Vít Dolejší, Ph.D., DSc.
Study programme:	Physics
Study branch:	Mathematical and Computational Modelling in Physics

Prague 2019

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague date ..... signature of the author

I would like to express my thank to RNDr. Jan Pech, Ph.D. who has a principal role in the leading of the thesis, namely for the topic suggestion and assistance with the software. I am also very grateful to my supervisor prof. RNDr. Vít Dolejší, Ph.D., DSc. for consulting the mathematical background.

Title: Numerical modeling of compressible flow using spectral element method

Author: Martin Jurček

Institute: Mathematical Institute of Charles University

Supervisor: prof. RNDr. Vít Dolejší, Ph.D., DSc., Department of Numerical Mathematics

Abstract: The development of computational fluid dynamics has given us a very powerful tool for investigation of fluid dynamics. However, in order to maintain the progress, it is necessary to improve the numerical algorithms. Nowadays, the high-order methods based on the discontinuous projection seem to have the largest potential for the future. In the work, we used open-source framework Nektar++, which provides the high-order discretization method. We tested the abilities of the framework for computing the compressible sonic and transonic flow. We successfully obtained simulations of the viscous and inviscid flow. We computed the lift and the drag coefficients and showed that for a higher polynomial order we can obtain the same accuracy with less degrees of freedom and lower computational time. Also, we tested the shock capturing method for the computation of the inviscid transonic flow and confirmed the potential of the high order methods.

Keywords: Compressible Navier Stokes equations, Discontinuous Galerkin, Nektar++

Contents

1. Introduction

Fluid dynamics is a physical discipline that studies the flow of fluids. Fluid dynamics is important mainly because a wide range of application including aerodynamics, geodynamics, meteorology, medicine, hydrology as well as car industry, aviation and astronautics. This wide range includes both academic and industrial applications.

Before the progress of computer technology, more than sixty years ago, the image of the flow was obtained mainly by physics experiments. Nowadays, these experiments are conducted in wind tubes. Although these experiments have still a significant role in the research of fluid dynamics, the computer technology provides us an another approach to obtain an image of the flow. This technological progress led to a development of numerical mathematics. The field that uses numerical mathematics to simulate the flow of the fluid is called computational fluid dynamics (CFD). The development of CFD had a significant impact on the wide range of applications as mentioned above and the CFD has become widespread.

However, the real world problems are complex and simulating can be a very challenging task. A mathematical simulation consists of a couple of parts. At first a physical model that describes the phenomenon must be derived (in the case of fluid dynamics the Euler and Navier-Stokes equations) and described mathematically usually using partial differential equations. Then the numerics is applied, firstly by transformating the problem into a system of algebraic equations. This process is called the numerical discretization. The resulting system of algebraic equations can be solved by an appropriate numerical method, for example, Newton(-like) method for nonlinear problems conjugate gradient method for symmetric linear problems or GMRES method for non-symmetric linear problems.

In the thesis we deal with the simulation of the compressible flow. The nature of the equations has a fundamental impact on the method used for the discretization of the problem.

The first investigated discretization methods for the compressible flow were the finite difference method (FDM) and the finite volume method (FVM), for more information see Feistauer et al. [2003]. While the finite difference method has rather a historical importance and is used only in special cases today, the finite volume method is still the most popular in the industry research. However, it has several drawbacks. The basic scheme has only the first order of accuracy which requires the use of high number of degrees of freedom in order to achieve the required accuracy. Additional drawback of the low order schemes is the occurrence of a high amount the numerical viscosity which can dominate the physical one. There exist also high-order finite volume schemes but their theoretical justification is questionable.

Another approach represents the finite element method (FEM). However, this scheme is not suitable for equations with a dominating convection term because of the Gibbs phenomenon, which leads to spurious oscillations in the solution. This holds for the compressible Navier-Stokes and Euler equations and the simple use of the finite element method is not suitable. The attempt to solve this problem

leads to the streamline diffusion method by adding an artificial diffusion in the equations.

However, a more natural approach is the discontinuous Galerkin method (DG), which was firstly introduced by Reed and R. Hill [1973]. It is based on the discontinuous projection and weak formulation, and combines both the finite volume method and finite element method. Unlike the finite volume method, the discontinuous Galerkin method is suitable for using a higher order of the polynomial expansion, see [Dolejší and Feistauer, 2015]. There are several options how to define the discontinuous Galerkin scheme. In the work we use the local discontinuous Galerkin method, for details see Cockburn and Shu [1998]. The methods that use higher order polynomial expansions on multi element grids are called the high-order spectral/hp element methods.

The high-order spectral element methods have many advantages. The usage of a high polynomial degree reduces the numerical diffusion. Using higher polynomial order, we can achieve the same accuracy with a coarser mesh and less degrees of freedom. We can also use high order meshes with curved elements for better description of more complex geometries.

In September 2006, 22 university organizationals and industry research organizations from more than 10 European countries joined forces in ADIGMA project. The main goal of the project was the development and improvement of high-order methods for the compressible flow as well as deepen the knowledge of the whole field. The project has finished in 2009 and shown a big potential of high-order methods both in academia and industry. More information and result can be found in Norbert et al. [2010].

In the work, we use open-source framework Nektar++, which is designed to provide the spectral/hp element discretization. The compressible flow solver applies the discontinuous Galerkin discretization using a high polynomial expansion.

The goal of the work is to summarize the numerical discretization implemented in Nektar++, create suitable high order meshes for NACA0012 airfoil and test the compressible flow solver in the two-dimensional case using Nektar++ for various types of flow regimes around NACA0012 airfoil, which is often used in simulations. This may be done by simulating both the viscous and inviscid flow for different Mach and Reynolds numbers and computing the lift and the drag coefficients for different meshes and polynomial expansions or studying the isolines of the solution. In the case of the transonic flow we would like to investigate the creation of the shock wave, which is typical for this type of flow.

2. Governing equations of the compressible flow

In computational fluid dynamics, we distinguish two key concepts of describing a flow of the fluid: the compressible and the incompressible flow. The incompressible flow refers to the case where the density in an infinitesimally small domain moving with the flow is constant. If the change of the density is no more negligible it is necessary to work with the concept of the compressible flow, where the incompressibility is no more assumed.

In this chapter, the fundamental concepts and equations for the description of the compressible flow are summarized. We present the basic concepts of the inviscid and viscous flow and formulate the Euler and the Navier-Stokes equations. Although, we do not cover the whole process of a derivation we introduce the key concepts. A detail description can be found in Feistauer et al. [2003].

2.1 Euler equations

In fluid dynamics, the Euler equations are hyperbolic partial differential equations describing the inviscid flow. The equations can be derived using the balance of mass, linear momentum and energy, assuming that the we do not take into account the effects of viscosity and heat conductivity. The Euler equations represent conservation law and in two dimensions they can be expressed in the following form:

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot f(\boldsymbol{w}) = 0, \tag{2.1}$$

where $f(\boldsymbol{w}) = (\boldsymbol{f_1}, \boldsymbol{f_2})$ represents the inviscid fluxes and $\boldsymbol{w}$ represents the state vector of conservative variables: density ρ, linear momentum in x and y direction $\rho v_1, \rho v_2$, respectively, and total energy E:

$$\boldsymbol{w} = \begin{bmatrix} \rho \\ \rho v_1 \\ \rho v_2 \\ E \end{bmatrix}, \quad \boldsymbol{f_1} = \begin{bmatrix} \rho v_1 \\ p + \rho v_1^2 \\ \rho v_1 v_2 \\ v_1(E+p) \end{bmatrix}, \quad \boldsymbol{f_2} = \begin{bmatrix} \rho v_2 \\ \rho v_1 v_2 \\ p + \rho v_2^2 \\ v_2(E+p) \end{bmatrix}, \tag{2.2}$$

where p stands for the pressure. In the work we are concerned with the flow of a perfect gas. The corresponding equation of state is given by:

$$p = R\rho T, \tag{2.3}$$

where R is the gas constant and T is the temperature. The total energy is equal to the sum of the internal and the kinetic part:

$$E = \rho \left(c_v T + \frac{|\boldsymbol{v}|^2}{2} \right), \tag{2.4}$$

where $\boldsymbol{v}$, $\boldsymbol{v} = (v_1, v_2)$ is the vector of the velocity and c_v denotes the specific heat at constant volume. Using relations $\gamma = c_p/c_v$ and $R = c_p - c_v$, where γ is the

Poisson adiabatic constant and c_p the specific heat at a constant pressure, we can rewrite (2.4) in the following form:

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \left|\boldsymbol{v}\right|^2 . \tag{2.5}$$

2.1.1 Mach number

An important quantity which characterize the flow is the Mach number. It is defined as a ratio of the absolute value of flow velocity $\boldsymbol{v}$ to local speed of sound a:

$$M = \frac{|\boldsymbol{v}|}{a}, \tag{2.6}$$

where local speed of sound a is equal to:

$$a = \sqrt{\gamma \frac{p}{\rho}}. \tag{2.7}$$

We distinguish various types of the flow depending on the value of the Mach number. The flow at point $\boldsymbol{x}$ and time t is said to be:

- subsonic if $M(\boldsymbol{x}, t) < 1$,
- sonic if $M(\boldsymbol{x}, t) = 1$,
- supersonic if $M(\boldsymbol{x}, t) > 1$.

If there is a subset of our computational domain where the flow is subsonic and subset where the flow is supersonic, we call this flow transonic. For details, see Feistauer et al. [2003].

In the work, we aim to simulate a subsonic flow and transonic flow. The compressibility can not be neglected for these flows and we can not use the incompressible flow approach.

2.2 Navier-Stokes equations

The Navier-Stokes equations describe a viscid flow. Unlike the Euler equations, the Navier-Stokes equations contain spatial derivative of viscid fluxes R

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot f(\boldsymbol{w}) = \nabla \cdot R(\boldsymbol{w}, \nabla \boldsymbol{w}). \tag{2.8}$$

In this case $R(\boldsymbol{w}, \nabla \boldsymbol{w}) = (\boldsymbol{R_1}, \boldsymbol{R_2})$

$$\boldsymbol{R_1} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ u\tau_{xx} + v\tau_{yx} + k\frac{\partial T}{\partial x} \end{bmatrix}, \boldsymbol{R_2} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + k\frac{\partial T}{\partial y} \end{bmatrix},$$

where T and k denote temperature and thermal conductivity, respectively. The elements of Cauchy stress tensor τ are equal to:

$$\tau_{xx} = 2\mu\left(\frac{\partial v_1}{\partial x} - \frac{1}{3}(\frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y})\right), \quad \tau_{yy} = 2\mu\left(\frac{\partial v_2}{\partial y} - \frac{1}{3}(\frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y})\right),$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial v_1}{\partial y} + \frac{\partial v_2}{\partial x}\right),$$

where μ stands for dynamic viscosity.

In the work we use the dimensionless form of the equations which we obtain by rescaling (2.8) by several characteristic quantities, namely the reference density ρ^*, velocity v^*, and length L^*, for details see Feistauer et al. [2003]. Then the important quantities characterizing the flow are the Reynolds and Prandtl numbers given by

$$Re = \frac{\rho^* v^* L^*}{\mu}, \tag{2.9}$$

and

$$Pr = \frac{c_p \mu}{k}, \tag{2.10}$$

respectively, where μ denotes the dynamic viscosity, and k denotes reference thermal conductivity.

3. Discontinuous Galerkin

The goal of a spatial discretization is to transform the physical problem into algebraic one. Having formulated the governing equations, we proceed to formulate the numerical discretization for both the Euler and the Navier-Stokes equations. In this chapter we describe the discontinuous Galerkin method. The discontinuous Galerkin method is based on the discontinuous projection and the weak formulation and can be seen, in a sense, as a combination of the finite volume and finite element method.

At first, we divide the mesh into K nonoverlaping elements. This is explained in Section 3.1. As we mentioned above, the discontinuous Galerkin uses the discontinuous projection. This means that we use a finite dimensional space of piecewise discontinuous functions to approximate the test and solution functions. The appropriate finite elements spaces are defined in Section 3.2.

To obtain the spatial discretization of the Euler equations, we begin with writing the weak formulation of the Euler equations, see Section 3.3. Unlike the finite element method, the functions are generally discontinuous and the information between elements is propagated via the numerical flux. We define the appropriate advection numerical fluxes and discuss the connection with the Riemann problem in Section 3.4 and we present one exact and three approximative Riemann solvers used in the work.

Further, in Section 3.5, we focus on the discretization of the Navier-Stokes equations, which contain the viscid fluxes. We introduce the mixed formulation and the local discontinuous Galerkin method and discuss the choice of the numerical fluxes. Finally, the final spatial discretization for the Navier-Stokes equations is formulated.

The implementation of the boundary conditions is discussed in Section 3.7. We summarize the implementation of far-field, full slip and no slip boundary conditions.

In Section 3.8, we define our approximative solution. Finally, we discuss the shock capturing method. We note, that we use the Einstein summation convention in the work.

3.1 Mesh partition

In the work, the computational domain is $\Omega \times (0, T)$, where $\Omega \in \mathbb{R}^2$ is a nonempty subset of the two dimensional space and $T > 0$. Our spatial computational domain Ω is divided into a finite number of nonoverlaping closed elements Ω_i, $i = 1, \cdots, K$, such that:

$$\Omega \subseteq \bigcup_{i=1}^{K} \Omega_i, \quad \Omega_i \bigcap_{i \neq j} \Omega_j = \emptyset. \tag{3.1}$$

We denote the boundary of Ω_i as $\partial\Omega_i$, $i = 1, \cdots, K$. We denote the set of all one dimensional edges of all elements Ω_i as Γ. We denote the set of all interface edges as Γ^I, the set of all edges where we set the wall (slip or on-slip) boundary condition as Γ^W and the set of edges where we set the far-field boundary condition as Γ^F.

3.2 Discontinuous finite element spaces

Having divided our space computational domain Ω into nonoverlaping elements Ω_j we can define the finite dimensional space. The natural choice is to use a polynomial function on each element. As mentioned above, there is no restriction on the continuity of the functions between elements. Our discontinuous finite dimensional space S_p is defined as follows:

$$S_p \equiv \{v; \quad v \in L^2(\Omega), \quad v|_{\Omega_i} \in P_p(\Omega_i) \quad \forall i = 1, \ldots, K\}, \tag{3.2}$$

where $P_p(\Omega_i)$ denotes the space of polynomials defined on Ω_i of degree at most p. Since we work with system of equations, we define the spaces for vector functions as:

$$\mathbb{S}_p \equiv S_p \times S_p \times S_p \times S_p, \tag{3.3}$$

$$\mathbb{T}_p \equiv S_p \times S_p \times S_p, \tag{3.4}$$

and

$$\mathbb{N}_p \equiv S_p \times S_p. \tag{3.5}$$

3.3 Discretizaton of the Euler equations

Let us formulate the spatial discretization of the Euler equations. At first, we formulate the weak form by multiplying equation (2.1) by test function $\boldsymbol{\phi}$ from a suitable space of test functions and integrating the equation over volume Ω_j, $j = 1, \ldots, K$. Applying Gauss's divergence theorem and using the Einstein summation convention, we obtain

$$\int_{\Omega_j} \frac{\partial \boldsymbol{w}}{\partial t} \cdot \boldsymbol{\phi} \, \mathrm{dx} + \int_{\partial\Omega_j} \boldsymbol{f_i}(\boldsymbol{w}) n_i \cdot \boldsymbol{\phi} \, \mathrm{dS} - \int_{\Omega_j} \boldsymbol{f_i}(\boldsymbol{w}) \cdot \frac{\partial \boldsymbol{\phi}}{\partial x_i} = 0. \tag{3.6}$$

At this point, we use discontinuous finite element space $\mathbb{S}_p$ to find functions to approximate $\boldsymbol{w}$ and $\boldsymbol{\phi}$. Let $\boldsymbol{w}^\delta(t)$ be the approximate solution function on Ω satisfying $\boldsymbol{w}^\delta(t)(x) = \boldsymbol{w}^\delta(x,t)$ and $\boldsymbol{w}^\delta(t) \in \mathbb{S}_p$ for $\forall t \in (0,T)$ and $\boldsymbol{\phi}^\delta \in \mathbb{S}_p$ the test function.

Summing equation (3.6) over all elements, we obtain:

$$\begin{aligned} &\sum_{j=1}^{K} \int_{\Omega_j} \frac{\partial \boldsymbol{w}^\delta(t)}{\partial t} \cdot \boldsymbol{\phi}^\delta \, \mathrm{dx} + \sum_{j=1}^{K} \int_{\partial\Omega_j} \boldsymbol{f_i}(\boldsymbol{w}^\delta(t)) n_i \cdot \boldsymbol{\phi}^\delta \, \mathrm{dS} \\ &- \sum_{j=1}^{K} \int_{\Omega_j} \boldsymbol{f_i}(\boldsymbol{w}^\delta(t)) \cdot \frac{\partial \boldsymbol{\phi}^\delta}{\partial x_i} = 0. \end{aligned} \tag{3.7}$$

Further, we have to give a meaning to the boundary integrals in (3.7) since functions $\boldsymbol{w}^\delta$ and $\boldsymbol{\phi}^\delta$ are discontinuous on $\partial\Omega_j$. In order to allow the information to propagate between the elements, we approximate the values of the fluxes by:

$$\boldsymbol{f_i}(\boldsymbol{w}^\delta) n_i \cdot \boldsymbol{\phi}^\delta \Big|_{\partial\Omega_j} \approx \boldsymbol{H}(\boldsymbol{w}_L^\delta, \boldsymbol{w}_R^\delta, \boldsymbol{n}) \cdot \boldsymbol{\phi}^\delta|_{\partial\Omega_j}, \tag{3.8}$$

where $\boldsymbol{H}(\boldsymbol{w}_L^\delta, \boldsymbol{w}_R^\delta, \boldsymbol{n})$ represents the *numerical flux*. Subscript L stands for the left state and subscript R stands for the right state. In the work, we consider the left state to be the interior state and the right state to be the exterior to Ω_j.

In the next chapter we summarize the properties of the numerical flux and explain the computation of the numerical flux using the Riemann solver.

3.4 Advection numerical flux

There are several possibilities how to define the numerical flux. Usually, we assume that numerical flux $\boldsymbol{H}$ satisfies the following properties:

- $\boldsymbol{H}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{n})$ is defined and continuous on $D \times D \times S$, where D represents the domain, where the fluxes are defined and S the unit sphere in $\mathbb{R}^2$
- $\boldsymbol{H}$ is consistent: $\boldsymbol{H}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{n}) = \boldsymbol{f_i}(\boldsymbol{u})n_i, \quad \forall \boldsymbol{u} \in D, \ \forall \boldsymbol{n} \in S$
- $\boldsymbol{H}$ is conservative:

$$\boldsymbol{H}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{n}) = -\boldsymbol{H}(\boldsymbol{v}, \boldsymbol{u}, -\boldsymbol{n}), \quad \forall \boldsymbol{v}, \boldsymbol{u} \in D, \ \forall \boldsymbol{n} \in S$$

For more information see Feistauer et al. [2003]

In the work, the advective numerical flux is computed via the Riemann solver. In the case of 2D domain, the system is locally rotated in the normal direction with respect to the interface and the following 1D Riemann problem is solved for a given point

$$\frac{\partial \tilde{\boldsymbol{w}}}{\partial t} + \frac{\partial \boldsymbol{f}_1(\tilde{\boldsymbol{w}})}{\partial \tilde{x}_1} = 0, \quad (\tilde{x}_1, t) \in \mathbb{R} \times (0, \infty), \tag{3.9}$$

$$\tilde{\boldsymbol{w}}(\tilde{x}_1, 0) = \tilde{\boldsymbol{w}}^0(\tilde{x}_1) = \begin{cases} \mathbb{Q}(\boldsymbol{n})\boldsymbol{w}_L, & \text{if } \tilde{x}_1 < 0, \\ \mathbb{Q}(\boldsymbol{n})\boldsymbol{w}_R, & \text{if } \tilde{x}_1 > 0, \end{cases} \tag{3.10}$$

where the coordinate $\tilde{x}_1$ is oriented in the direction of the normal vector $\boldsymbol{n}$ and

$$\tilde{\boldsymbol{w}} = \mathbb{Q}(\boldsymbol{n})\boldsymbol{w}, \tag{3.11}$$

and $\mathbb{Q}(\boldsymbol{n})$ is the rotation matrix through angle α ($\boldsymbol{n} = (\cos\alpha, \sin\alpha)$).

Having solved the Riemann problem, we rotate the system back to the Cartesian system:

$$\boldsymbol{H}(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}) = \mathbb{Q}^{-1}(\boldsymbol{n})\boldsymbol{g}_R(\mathbb{Q}(\boldsymbol{n})\boldsymbol{w_R}, \mathbb{Q}(\boldsymbol{n})\boldsymbol{w_L}), \tag{3.12}$$

where $\boldsymbol{g}_R$ is the solution of the 1D Riemann problem.

In order to solve the 1D Riemann problem we make use of an exact or approximate Riemann solver. In the next chapter, we briefly summarize the Riemann solvers used in the work. For more information, see Toro [2009].

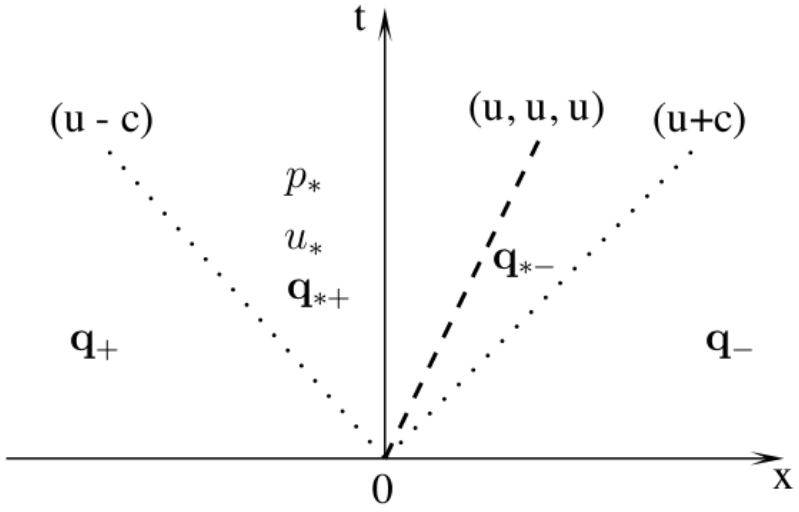

Figure 3.1: A scheme of the Riemann problem for the Euler equations (see Mengaldo et al. [2014]).

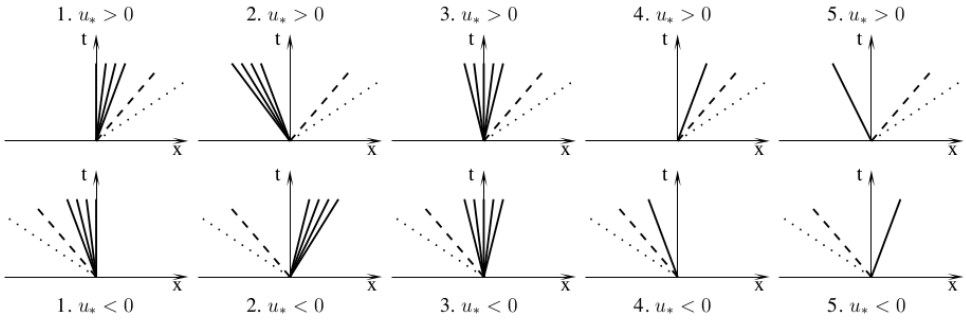

Figure 3.2: Possible wave patterns for the Riemann problem for the Euler equations (see Mengaldo et al. [2014]).

3.4.1 Exact Toro

Based on the analysis of the spectrum of the Euler problem, the solution of the Riemann problem consists of three waves. The left and right are either the rarefaction wave or the shock wave and the middle wave is the contact discontinuity. The general solution consists of four domains denoted as $\boldsymbol{q}_+$, $\boldsymbol{q}_{*+}$, $\boldsymbol{q}_{*-}$, $\boldsymbol{q}_-$, see Figure 3.1. The first step in order to obtain the solution at a given point, is to evaluate the velocity and pressure between the first and third wave u_* and p_* respectively. These quantities are constant across the contact discontinuity (only density changes). The values u_* and p_* can be obtained by solving a nonlinear algebraic equation using an iterative process. The second step consists of determining the wave pattern, see Figure 3.2. The final relations are given in Toro [2009].

3.4.2 Roe flux

The exact solver provides the most precise solution of the Riemann problem. However, the iterative process for finding the solution can be numerically costly. Therefore, a couple of approximate solver were introduced. One of the popular approximate Riemann solver is the Roe scheme. It is based on replacing a given nonlinear Jacobian matrix in the Riemann problem by a constant matrix with suitable properties. The approximation matrix is chosen that its eigenvalues $\hat{\lambda}$ and eigenvectors $\hat{\boldsymbol{r}}_n$ correspond are equal to eigenvalues λ and eigenvectors $\boldsymbol{r}_n$ of the original matrix with the Roe averaged values defined as:

$$\begin{aligned}
\sqrt{\hat{\rho}} &= \frac{1}{2}(\sqrt{\rho_L} + \sqrt{\rho_R}), \\
\hat{v_1} &= \frac{\sqrt{\rho_L}v_{1L} + \sqrt{\rho_R}v_{1R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\hat{v_2} &= \frac{\sqrt{\rho_L}v_{2L} + \sqrt{\rho_R}v_{2R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\hat{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}},
\end{aligned} \tag{3.13}$$

Finally, the flux is given by:

$$\boldsymbol{g}_{Roe} = \frac{1}{2}(f(\boldsymbol{w}_L) + f(\boldsymbol{w}_R)) - \frac{1}{2}\sum_{n=1}^{m} \gamma_n \left|\hat{\lambda}_n\right| \hat{\boldsymbol{r}}_n, \tag{3.14}$$

where coefficient γ_n can be easily derived, see Toro [2009]. Subscripts L and R stand for the left and right state, respectively.

3.4.3 HLL

Another approach was presented by Harten, Lax and van Lear. The key idea is to use a two wave instead of three wave configuration. These waves are responsible for three possible states, which are chosen based on wave speeds S_L and S_R defined as:

$$\begin{aligned}
S_L &= \min\{v_{1L} - a_L, v_{1R} - a_R\}, \\
S_R &= \max\{v_{1L} - a_L, v_{1R} - a_R\}.
\end{aligned} \tag{3.15}$$

Finally, the flux is given by:

$$\boldsymbol{g}_{HLL} = \begin{cases} \boldsymbol{f}(\boldsymbol{w_L}) & \text{if } 0 \leq S_L, \\ \dfrac{S_R\boldsymbol{f}(\boldsymbol{w_L}) - S_L\boldsymbol{f}(\boldsymbol{w_R}) + S_RS_L(\boldsymbol{w_R} - \boldsymbol{w_L})}{S_R - S_L} & \text{if } S_L \leq 0 \leq S_R, \\ \boldsymbol{f}(\boldsymbol{w_R}) & \text{if } 0 \geq S_R. \end{cases} \tag{3.16}$$

3.4.4 HLLC

HLLC Riemman solver uses a three wave configuration. There are four states chosen by wave speeds S_L, S_R defined as above and S_* defined as:

$$S_* = \frac{p_R - p_L + \rho_L v_{1L}(S_L - v_{1L}) - \rho_R v_{1R}(S_R - v_{1R})}{\rho_L(S_L - v_{1L}) - \rho_R(S_R - v_{1R})}, \tag{3.17}$$

and

$$\boldsymbol{w_{*K}} = \rho_K \left(\frac{S_K - v_{1K}}{S_K - S_*} \right) \begin{bmatrix} 1 \\ S* \\ v_{2K} \\ \frac{E_K}{\rho_K} + (S_* - v_{1K}) \left[S_* + \frac{p_K}{\rho_K (S_K - v_{1K})} \right] \end{bmatrix}, \tag{3.18}$$

Where $K = \{L, R\}$. The final flux is then given by:

$$\boldsymbol{g}_{HLL} = \begin{cases} \boldsymbol{f}(\boldsymbol{w_L}) & \text{if } 0 \leq S_L \\ \boldsymbol{f}(\boldsymbol{w_L}) - S_L(\boldsymbol{w_{*L}} - \boldsymbol{w_L}) & \text{if } S_L \leq 0 \leq S_* \\ \boldsymbol{f}(\boldsymbol{w_R}) - S_R(\boldsymbol{w_{*R}} - \boldsymbol{w_R}) & \text{if } S_* \leq 0 \leq S_R \\ \boldsymbol{f}(\boldsymbol{w_R}) & \text{if } 0 \geq S_R \end{cases} \tag{3.19}$$

3.5 Discretization of the Navier Stokes eqautions

The Navier-Stokes equations take into account the effects of the fluid viscosity and the transfer of the heat. Apart from inviscid fluxes $\boldsymbol{f}_i(\boldsymbol{w})$, $i = 1, 2$, which are treated the same as in the case of the Euler equations (see Section 3.3) with the given Riemann solver for the computation of the advective numerical flux, we have to deal with the discretization of viscous fluxes $\boldsymbol{R}_i(\boldsymbol{w}, \nabla \boldsymbol{w})$, $i = 1, 2$.

We can see that the viscous fluxes depend on the values of the derivatives of the velocity and the temperature, see (2.8). In order to obtain these values, we use the mixed formulation which introduces auxiliary vector $\boldsymbol{W} = [W_1, W_2, W_3] = [v_1, v_2, T]$. We can write $\boldsymbol{W} = P(\boldsymbol{w})$, where $P: \mathbb{R}^4 \to \mathbb{R}^3$ is defined as:

$$\begin{bmatrix} v_1 \\ v_2 \\ T \end{bmatrix} = \begin{bmatrix} \rho v_1 / \rho \\ \rho v_2 / \rho \\ \frac{1}{c_v \rho} \left[E - \frac{(\rho v_1)^2 + (\rho v_1)^2}{2\rho} \right]. \end{bmatrix} \tag{3.20}$$

We solve the following problem:

$$g = \nabla \boldsymbol{W}, \tag{3.21}$$

where $g = (\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3)$, $\boldsymbol{g}_i \in \mathbb{N}_p \ \forall i = 1, 2, 3$ and $\boldsymbol{W} \in \mathbb{T}_p$. For the definition of spaces $\mathbb{N}_p$ and $\mathbb{T}_p$ see Section 3.2. Then we should write: $\boldsymbol{R}_i(\boldsymbol{w}, \nabla \boldsymbol{w}) = \tilde{\boldsymbol{R}}_i(\boldsymbol{w}, g)$, $i = 1, 2$, but for, simplicity, we keep the same notation for $\boldsymbol{R}_i$. Then (2.8) takes the form:

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla \cdot f(\boldsymbol{w}) = \nabla \cdot R(\boldsymbol{w}, g), \tag{3.22}$$

We proceed as above, we multiply both equations by test functions Θ^δ and $\boldsymbol{\phi}^\delta$, integrate both equations over Ω_j, apply Gauss's divergence theorem and sum over the partition. Then, we obtain the following formulas:

$$\sum_{j=1}^{K} \int_{\Omega_j} \boldsymbol{g_i^\delta} \cdot \Theta^\delta \, \mathrm{dx} = \sum_{j=1}^{K} \int_{\partial\Omega_j} W_i^\delta \boldsymbol{n} \cdot \Theta^\delta \, \mathrm{dS} - \sum_{j=1}^{K} \int_{\Omega_j} W_i^\delta \frac{\partial \Theta_l^\delta}{\partial x_l} \, \mathrm{dx}, \quad \forall i = 1, 2, 3, \tag{3.23}$$

$$\sum_{j=1}^{K}\int_{\Omega_j}\frac{\partial \boldsymbol{w}^\delta}{\partial t}\cdot\boldsymbol{\phi}^\delta\ \mathrm{dx}+\sum_{j=1}^{K}\int_{\partial\Omega_j}\boldsymbol{f_i}(\boldsymbol{w}^\delta)n_i\cdot\boldsymbol{\phi}^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}\boldsymbol{f_i}(\boldsymbol{w}^\delta)\cdot\frac{\partial\boldsymbol{\phi}^\delta}{\partial x_i}\ \mathrm{dx}=$$
$$\sum_{j=1}^{K}\int_{\partial\Omega_j}\boldsymbol{R_i}(\boldsymbol{w}^\delta,g^\delta)n_i\cdot\boldsymbol{\phi}^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}\boldsymbol{R_i}(\boldsymbol{w}^\delta,g^\delta)\cdot\frac{\partial\boldsymbol{\phi}^\delta}{\partial x_i}\ \mathrm{dx}. \tag{3.24}$$

At this point, in order to give a meaning of the boundary integrals, we approximate the following terms by a diffusive numerical flux:

$$W_i^\delta\boldsymbol{n}\cdot\boldsymbol{\phi}^\delta\Big|_{\partial\Omega_j}\approx\hat{W}_i^\delta\boldsymbol{n}\cdot\boldsymbol{\phi}^\delta|_{\partial\Omega_j},\quad\forall i=1,2,3, \tag{3.25}$$

$$\boldsymbol{R}_i(\boldsymbol{w}^\delta,g^\delta)\cdot\boldsymbol{\phi}^\delta\Big|_{\partial\Omega_j}\approx\hat{\boldsymbol{R}}_i(\boldsymbol{w}^\delta,g^\delta)\cdot\boldsymbol{\phi}^\delta|_{\partial\Omega_j}\quad\forall i=1,2,3, \tag{3.26}$$

where $\hat{W}_i^\delta$ and $\hat{\boldsymbol{R}}_i$ represent the diffusion numerical flux specified below.

3.6 Diffusion numerical flux

As mentioned earlier, we used the local discontinuous Galerkin method (LDG) in the work. We begin with a little more general formula:

$$\hat{W}_i^\delta=\frac{1}{2}(W_i^{\delta L}+W_i^{\delta R})-\beta(W_i^{\delta L}-W_i^{\delta R}),\quad\forall i=1,2,3, \tag{3.27}$$

$$\begin{aligned}\hat{\boldsymbol{R}}_i(\boldsymbol{w}^\delta,g^\delta)=\frac{1}{2}[\boldsymbol{R}_i(\boldsymbol{w}^{\delta L},g^{\delta L})+\boldsymbol{R}_i(\boldsymbol{w}^{\delta R},g^{\delta R})]+\beta[\boldsymbol{R}_i(\boldsymbol{w}^{\delta L},g^{\delta L})-\\ \boldsymbol{R}_i(\boldsymbol{w}^{\delta R},g^{\delta R})]-(\eta/h)(W_i^{\delta L}-W_i^{\delta R})\boldsymbol{n},\quad\forall i=1,2,3,\end{aligned} \tag{3.28}$$

where subscripts L and R denote the internal and external values, respectively, η is the stabilization parameter and h is the typical scale of the problem.

At this moment we can recover various schemes depending on the choice of β. In our case (LDG) we set $\beta=\frac{1}{2}$ and $\eta=0$. We note, that we could obtain the central flux (CF) or the second Bassi-Rebay method (BR2) for the different choice of β.

Having set parameters β and η, we obtain the exterior value for the numerical flux of auxiliary variable W_i.

$$\hat{W}_i^\delta=W_i^{\delta R},\quad\forall i=1,2,3, \tag{3.29}$$

the internal values for viscous flux $\boldsymbol{R_i}$:

$$\hat{\boldsymbol{R}}_i(\boldsymbol{w}^\delta,g^\delta)=\boldsymbol{R_i}(\boldsymbol{w}^{\delta L},g^{\delta L}),\quad\forall i=1,2. \tag{3.30}$$

3.7 Boundary conditions

One of the factors that influences the convergence of the solution is the method how the boundary conditions are implemented. It seems that it is more advantageous to impose the boundary condition via modifying the numerical flux, instead of modifying the boundary solution directly at the points. We call this approach the weak boundary conditions. It is shown that the usage of the weak boundary conditions may improve the convergence of the solution.

In following section, we summarize the boundary conditions used in the work and briefly present the implementation. For additional information, see Mengaldo et al. [2014]

3.7.1 Far-field boundary conditions

In mathematical simulations, we are limited to the finite computational domain. For this reason, we must define the far-field boundary conditions on the boundary of computational domain Ω.

At first we consider advection terms. In this case we have to give a meaning to state $\boldsymbol{w}_R$ for $\forall \mathscr{T} \in \Gamma^F$. We call this state a ghost state, since it has to be prescribed. We write

$$\boldsymbol{w}_R = \begin{bmatrix} \rho_\infty \\ \rho_\infty v_{1\infty} \\ \rho_\infty v_{2\infty} \\ E_\infty \end{bmatrix}, \tag{3.31}$$

where $\rho_\infty, v_{1\infty}, v_{2\infty}, E_\infty$ represent the free-stream density, velocity in x and y direction and total energy, respectively. This state is used to compute advection numerical flux $\boldsymbol{H}(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n})$ using the exact or approximate Riemann solver for $\forall \mathscr{T} \in \Gamma^F$.

Further we consider the diffusion term in the Navier-Stokes equations. The diffusion numerical flux $\hat{W}_i^\delta$ takes the external values and $\hat{\boldsymbol{R}}_i$ the internal values. For this reason we have to set only $\hat{W}_i^\delta$, $i = 1, 2, 3$. We set:

$$\boldsymbol{W}^{\delta W} = \begin{bmatrix} \hat{W}_1^\delta \\ \hat{W}_2^\delta \\ \hat{W}_3^\delta \end{bmatrix} = \begin{bmatrix} v_{1L} \\ v_{2L} \\ \frac{p_L}{\rho_L R} \end{bmatrix} \quad \forall \mathscr{T} \in \Gamma^F. \tag{3.32}$$

3.7.2 Full slip boundary conditions

The boundary conditions for the Euler equations prevent the fluid to get through obstacle (e.q. NACA profile). Without any assumptions on the viscosity of the fluid, we require only the normal component of the velocity on the boundary to be zero:

$$\boldsymbol{v} \cdot \boldsymbol{n} = 0 \qquad \text{on } \partial\Omega^W \tag{3.33}$$

where $\partial\Omega^W$ is the boundary of the obstacle and $\boldsymbol{n}$ is the normal vector, with respect to the boundary.

The right (ghost) state for computing the numerical flux for $\forall \mathscr{T} \in \Gamma^W$ is equal to:

$$\boldsymbol{w}_R = \begin{bmatrix} \rho_L \\ \rho_L v_{1L} - 2(\boldsymbol{v}_L \cdot \boldsymbol{n}) n_1 \\ \rho_L v_{2L} - 2(\boldsymbol{v}_L \cdot \boldsymbol{n}) n_2 \\ E_L \end{bmatrix} \tag{3.34}$$

where subscript L denotes the inner state.

3.7.3 No slip boundary conditions

Because of the viscosity, the Navier-Stokes equations require the velocity to be zero on the boundary

$$\boldsymbol{v} = 0 \qquad \text{on } \partial\Omega^W \tag{3.35}$$

Further, we require the boundary condition for the temperature. We distinguish two cases:

- Isothermal wall
- Adiabatic wall

In case of the isothermal wall, we prescribe the temperature on the boundary

$$T = T_B, \tag{3.36}$$

and in case of the adiabatic wall, we set the thermal flux to be zero:

$$k \nabla T \cdot \boldsymbol{n} = 0. \tag{3.37}$$

The right (ghost) state for the advection numerical flux for $\forall \mathscr{T} \in \Gamma^W$ is the same for both the isothermal and adiabatic wall and is equal to:

$$\boldsymbol{w}_R = \begin{bmatrix} \rho_L \\ -\rho_L v_{1L} \\ -\rho_L v_{2L} \\ E_L \end{bmatrix}. \tag{3.38}$$

Further, we consider diffusion terms. In case of the isothermal wall, we set:

$$\boldsymbol{W}^{\delta N} = \begin{bmatrix} 0 \\ 0 \\ T_B \end{bmatrix} \quad \forall \mathscr{T} \in \Gamma^W. \tag{3.39}$$

Finally, the boundary condition for the adiabatic wall is of the form:

$$\boldsymbol{W}^{\delta W} = \begin{bmatrix} 0 \\ 0 \\ T_L \end{bmatrix} \quad \forall \mathscr{T} \in \Gamma^W, \tag{3.40}$$

where T_L is the inner temperature. These values are directly used as the numerical flux for auxiliary variable $\boldsymbol{W}$, see Section 3.5.

3.8 Formulation of the approximative solution

In this section we summarize the approximative solutions for both the Euler and the Navier-Stokes equations. Firstly we introduce the following notation. We introduce two inner products: $(\cdot,\cdot)_{\delta S}$: $\mathbb{S}_p \times \mathbb{S}_p \to \mathbb{R}$ as:

$$(\boldsymbol{u},\boldsymbol{w})_{\delta S} = \sum_{j=1}^{K} \int_{\Omega_j} \boldsymbol{u}\cdot\boldsymbol{w}\ \mathrm{dx} \quad \forall \boldsymbol{u},\boldsymbol{w} \in \mathbb{S}_p, \tag{3.41}$$

and $(\cdot,\cdot)_{\delta N}$: $\mathbb{N}_p \times \mathbb{N}_p \to \mathbb{R}$ as:

$$(\boldsymbol{u},\boldsymbol{w})_{\delta N} = \sum_{j=1}^{K} \int_{\Omega_j} \boldsymbol{u}\cdot\boldsymbol{w}\ \mathrm{dx} \quad \forall \boldsymbol{u},\boldsymbol{w} \in \mathbb{N}_p. \tag{3.42}$$

Further we define the following forms: $b_\delta :\ \mathbb{S}_p \times \mathbb{S}_p \to \mathbb{R}$ as:

$$b_\delta(\boldsymbol{w}^\delta,\boldsymbol{\phi}^\delta) = \sum_{\mathscr{T}\in\Gamma} \int_{\mathscr{T}} \boldsymbol{H}(\boldsymbol{w}_L^\delta,\boldsymbol{w}_R^\delta,\boldsymbol{n})\cdot\boldsymbol{\phi}^\delta\ \mathrm{dS} - \sum_{j=1}^{K}\int_{\Omega_j} \boldsymbol{f_i}(\boldsymbol{w}^\delta(t))\cdot\frac{\partial\boldsymbol{\phi}^\delta}{\partial x_i}\ \mathrm{dx}, \tag{3.43}$$

$a_\delta :\ \mathbb{S}_p \times (\mathbb{N}_p \times \mathbb{N}_p \times \mathbb{N}_p) \times \mathbb{S}_p \to \mathbb{R}$ as:

$$a_\delta(\boldsymbol{w}^\delta,g^\delta,\boldsymbol{\phi}^\delta) = \sum_{j=1}^{K}\int_{\partial\Omega_j} \boldsymbol{R}_i(\boldsymbol{w}^{\delta L},g^{\delta L})n_i\cdot\boldsymbol{\phi}^\delta\ \mathrm{dS} - \sum_{j=1}^{K}\int_{\Omega_j}\boldsymbol{R}_i(\boldsymbol{w}^\delta,g^\delta)\cdot\frac{\partial\boldsymbol{\phi}^\delta}{\partial x_i}\ \mathrm{dx}, \tag{3.44}$$

and $c_\delta^i : \mathbb{T}_p \times \mathbb{N}_p \to \mathbb{R}$ as:

$$\begin{aligned} c_\delta^i(\boldsymbol{W}^\delta,\Theta^\delta) = &\sum_{\mathscr{T}\in\Gamma^I}\int_{\mathscr{T}} \hat{W}_i^{\delta R}\boldsymbol{n}\cdot\Theta^\delta\ \mathrm{dS} + \sum_{\mathscr{T}\in\Gamma^F}\int_{\mathscr{T}}\hat{W}_i^{\delta F}\boldsymbol{n}\cdot\Theta^\delta\ \mathrm{dS} + \\ &\sum_{\mathscr{T}\in\Gamma^W}\int_{\mathscr{T}}\hat{W}_i^{\delta W}\boldsymbol{n}\cdot\Theta^\delta\ \mathrm{dS} - \sum_{j=1}^{K}\int_{\Omega_j} W_i^\delta\frac{\partial\Theta_l^\delta}{\partial x_l}\ \mathrm{dx} \quad \forall i=1,2,3. \end{aligned} \tag{3.45}$$

Definition 1. *We call $\boldsymbol{w}^\delta \in C^1(0,T;\mathbb{S}_p)$ a semi-discrete solution of the Euler equations if:*

$$\begin{aligned} &\left(\frac{\partial\boldsymbol{w}^\delta(t)}{\partial t},\boldsymbol{\phi}^\delta\right)_{\delta S} + b_\delta(\boldsymbol{w}^\delta,\boldsymbol{\phi}^\delta) = 0 \quad \forall\boldsymbol{\phi}^\delta\in\mathbb{S}_p,\ \forall t\in(0,T), \\ &\boldsymbol{w}^\delta(0) = \boldsymbol{w}^{\delta 0} \end{aligned} \tag{3.46}$$

where $\boldsymbol{w}^{\delta 0}$ is the $\mathbb{S}_p$ approximation of $\boldsymbol{w}^0$, which is an initial condition, and the meaning of $\boldsymbol{w}_R$ for $\forall\mathscr{T}\in\Gamma^W$, $\forall\mathscr{T}\in\Gamma^F$ was explained in Section 3.7.

Definition 2. *We call $(g^\delta,\boldsymbol{w}^\delta)$, $g^\delta = (\boldsymbol{g}_1^\delta,\boldsymbol{g}_2^\delta,\boldsymbol{g}_3^\delta)$ where $\boldsymbol{g}_i^\delta \in C^1(0,T;\mathbb{N}_p), \forall i = 1,2,3$ and $\boldsymbol{w}^\delta \in C^1(0,T;\mathbb{S}_p)$ a semi-discrete solution of the Navier-Stokes equations if:*

$$\left(\boldsymbol{g}_i^\delta(t),\Theta^\delta\right)_{\delta N} = c_\delta^i(\boldsymbol{W}^\delta,\Theta^\delta), \quad \forall i=1,2,3,\ \forall\Theta^\delta\in\mathbb{N}_p\ \forall t\in(0,T), \tag{3.47}$$

$$\begin{aligned} &\left(\frac{\partial\boldsymbol{w}^\delta(t)}{\partial t},\boldsymbol{\phi}^\delta\right)_{\delta S} + b_\delta(\boldsymbol{w}^\delta,\boldsymbol{\phi}^\delta) = a_\delta(\boldsymbol{w}^\delta,g^\delta,\boldsymbol{\phi}^\delta) \quad \forall\boldsymbol{\phi}^\delta\in\mathbb{S}_p,\ \forall t\in(0,T), \\ &\boldsymbol{w}^\delta(0) = \boldsymbol{w}^{\delta 0} \end{aligned} \tag{3.48}$$

where $\boldsymbol{w}^{\delta 0}$ is the $\mathbb{S}_p$ approximation of $\boldsymbol{w}^0$, which is an initial condition, and the meaning of $\boldsymbol{w}_R$ for $\forall\mathscr{T}\in\Gamma^W$, $\forall\mathscr{T}\in\Gamma^F$ was explained in Section 3.7. The relation between $\boldsymbol{W}^\delta$ and $\boldsymbol{w}^\delta$ is given by (3.20):

$$\boldsymbol{W}^\delta = P(\boldsymbol{w}^\delta). \tag{3.49}$$

3.9 Shock capturing

One of the difficulties while solving hyperbolic partial differential equations may be the presence of shock waves, which are characterized by abrupt change of the state variables. This shock waves are typical a for transonic flow, as we will see in Section 8.3. This fact leads to numerical instabilities. The basic idea how to prevent this fact is to add a suitable diffusion term to the governing equations. However, adding the diffusion term makes sense only at the elements where the shock waves are present. There are several possibilities how to detect and capture the elements, where the shock waves occurs. For this purpose, we use a sensor variable s_e defined as:

$$s_e = \log_{10}\left(\frac{\|\rho_e^p - \rho_e^{p-1}\|_{L_2}}{\|\rho_e^p\|_{L_2}}\right), \tag{3.50}$$

where ρ_e^p is the actual average solution at the element e and ρ_e^{p-1} is the average solution on the element e reduced to the polynomial of degree $p-1$. Having the method to capture the discontinuities we can define the term that actually influences the solution (for more information see Persson and Peraire [2006]). For this purpose, we add term $\nabla\cdot(\varepsilon\nabla\boldsymbol{w})$ to the right side of the governing equations, where ε is defined as:

$$\epsilon = \epsilon_0 \begin{cases} 0 & \text{if } s_e < s_\kappa - \kappa, \\ 0.5\left(1+\sin\dfrac{\pi(s_e - s_\kappa)}{2\kappa}\right) & \text{if } s_\kappa - \kappa < s_e < s_\kappa + \kappa, \\ 1 & \text{if } s_e > s_\kappa + \kappa, \end{cases} \tag{3.51}$$

where κ is chosen empirically to obtain a sharp but smooth function and

$$\epsilon_0 = c_1 h/p,\ \ s_\kappa = c_2/p^4, \tag{3.52}$$

where h is the element size, p is the polynomial expansion degree and c_1 and c_2 are constants. In our case, we set $c_1 = 10$ and $c_2 = 1$.

This definition also ensures that the diffusion is added locally. We note, that we used the shock capturing only for the Euler equations. The Navier-Stokes equations contain a diffusive term and adding another was not necessary. The Euler equations then take the following form:

$$\frac{\partial \boldsymbol{w}}{\partial t} + \nabla\cdot f(\boldsymbol{w}) = \nabla\cdot(\varepsilon\nabla\boldsymbol{w}). \tag{3.53}$$

We use a similar discretization method based on the mixed formulation

$$g = \nabla\boldsymbol{w}, \tag{3.54}$$

where $g = (\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_4)$, $\boldsymbol{g}_i \in \mathbb{N}_p\ \forall i = 1,2,3,4$ and the local discontinuous Galerkin approach. Proceeding the same as above we obtain:

$$\sum_{j=1}^{K}\int_{\Omega_j} \boldsymbol{g_i^\delta}\cdot\Theta^\delta \,\mathrm{dx} = \sum_{j=1}^{K}\int_{\partial\Omega_j} \hat{w}_i^\delta \boldsymbol{n}\cdot\Theta^\delta \,\mathrm{dS} - \sum_{j=1}^{K}\int_{\Omega_j} w_i^\delta \frac{\partial\Theta_l^\delta}{\partial x_l}\,\mathrm{dx}, \quad \forall i = 1,2,3,4 \tag{3.55}$$

$$\sum_{j=1}^{K}\int_{\Omega_j}\frac{\partial \boldsymbol{w}^\delta}{\partial t}\cdot\boldsymbol{\phi}^\delta\ \mathrm{dx}+\sum_{\mathscr{T}\in\Gamma}\int_{\mathscr{T}}\boldsymbol{H}(\boldsymbol{w}_L^\delta,\boldsymbol{w}_R^\delta,\boldsymbol{n})\cdot\boldsymbol{\phi}^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}\boldsymbol{f_i}(\boldsymbol{w}^\delta)\cdot\frac{\partial\boldsymbol{\phi}^\delta}{\partial x_i}\ \mathrm{dx}=$$
$$\sum_{j=1}^{K}\int_{\partial\Omega_j}\epsilon(\hat{\boldsymbol{g}}_i^\delta\cdot n)\boldsymbol{\phi}_i^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}\epsilon[\boldsymbol{g_i^\delta}]_j\frac{\partial\boldsymbol{\phi}_i^\delta}{\partial x_j}\ \mathrm{dx}. \tag{3.56}$$

We use the same numerical flux, presented in Section 3.6. Then we obtain the exterior values for the variable:

$$\hat{w}_i^\delta=w_i^{\delta R},\quad \forall i=1,2,3,4, \tag{3.57}$$

and the internal values for:

$$\hat{g}^\delta=g^{\delta L}. \tag{3.58}$$

We use internal values for the boundary conditions for variable $\hat{w}_i^\delta$ for $\forall i=1,2,3,4,$. We define the following forms: $d_\delta^i:\ \mathbb{S}_p\times\mathbb{N}_p\to\mathbb{R}\quad\forall i=1,2,3,4,$

$$d_\delta^i(\boldsymbol{w}^\delta,\Theta^\delta)=\sum_{j=1}^{K}\int_{\partial\Omega_j}w_i^{\delta R}\boldsymbol{n}\cdot\Theta^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}w_i^\delta\frac{\partial\Theta_l^\delta}{\partial x_l}\ \mathrm{dx},\quad\forall i=1,2,3,4, \tag{3.59}$$

$s_\delta:\ (\mathbb{N}_p\times\mathbb{N}_p\times\mathbb{N}_p\times\mathbb{N}_p)\times\mathbb{S}_p\to\mathbb{R},$

$$s_\delta(g^\delta,\boldsymbol{\phi}_\delta)=\sum_{j=1}^{K}\int_{\partial\Omega_j}\epsilon(\boldsymbol{g}_i^{\delta L}\cdot n)\boldsymbol{\phi}_i^\delta\ \mathrm{dS}-\sum_{j=1}^{K}\int_{\Omega_j}\epsilon[\boldsymbol{g}_i^\delta]_j\frac{\partial\boldsymbol{\phi}_i^\delta}{\partial x_j}\ \mathrm{dx}. \tag{3.60}$$

Definition 3. *We call (g^δ,$\boldsymbol{w}^\delta$), $g^\delta=(\boldsymbol{g}_1^\delta,\boldsymbol{g}_2^\delta,\boldsymbol{g}_3^\delta,\boldsymbol{g}_4^\delta)$, where $\boldsymbol{g}_i^\delta\in C^1(0,T;\mathbb{N}_p)\ \forall i=1,2,3,4$ and $\boldsymbol{w}^\delta\in C^1(0,T;\mathbb{S}_p)$ a semi-discrete solution of the Euler equations with the shock capturing if:*

$$\left(\boldsymbol{g}_i^\delta(t),\Theta^\delta\right)_{\delta N}=d_\delta^i(\boldsymbol{w}^\delta,\Theta^\delta),\quad\forall i=1,2,3,4,\ \forall\Theta^\delta\in\mathbb{N}_p\ \forall t\in(0,T),$$

$$\left(\frac{\partial\boldsymbol{w}^\delta(t)}{\partial t},\boldsymbol{\phi}^\delta\right)_{\delta S}+b_\delta(\boldsymbol{w}^\delta,\boldsymbol{\phi}^\delta)=s_\delta(g^\delta(t),\boldsymbol{\phi}^\delta)\quad\forall\boldsymbol{\phi}^\delta\in\mathbb{S}_p,\ \forall t\in(0,T),$$
$$\boldsymbol{w}^\delta(0)=\boldsymbol{w}^{\delta 0}, \tag{3.61}$$

where $\boldsymbol{w}^{\delta 0}$ is the $\mathbb{S}_p$ approximation of $\boldsymbol{w}^0$, which is an initial condition.

4. Expansion bases

In the previous chapter, we formulated the discontinuous Galerkin method. We summarized the discretization of the Euler and the Navier-Stokes equations and introduced the numerical flux for the advection and diffusion terms to propagate the information between elements of the mesh. So far, we defined finite dimensional spaces S_p for scalar and $\mathbb{S}_p$ for vector functions (see Section 3.2) but we did not actually construct the basis of the spaces. In this chapter, we focus on a decomposition of the solution and test function into an expansion basis. There are several possibilities how to choose the appropriate basis. The main factor is a computational efficiency.

In Section 4.1, the expansion basis in 1D using the Jacobi polynomial is defined and is expanded into higher dimension in Section 4.2. Further, we define the integration and differentiation both on a reference and general element.

4.1 1D expansion basis

Firstly, we consider 1D expansion basis, since it can be easily extended on quadrilateral elements using the tensor product. Let $P > 0$ be the given polynomial approximation degree. Let $-1 \leq \xi \leq 1$, than we define the following expansion basis functions $\psi_p,\ p = 0, \ldots, P$:

$$\psi_p(\xi) = \begin{cases} \dfrac{1-\xi}{2} & \text{if } p = 0, \\ \left(\dfrac{1-\xi}{2}\right)\left(\dfrac{1+\xi}{2}\right) P_{p-1}^{1,1}(\xi) & \text{if } 0 < p < P, \\ \dfrac{1+\xi}{2} & \text{if } p = P, \end{cases} \tag{4.1}$$

where $P_{p-1}^{1,1}(\xi)$ denotes the Jacobi polynomial. For more information about the

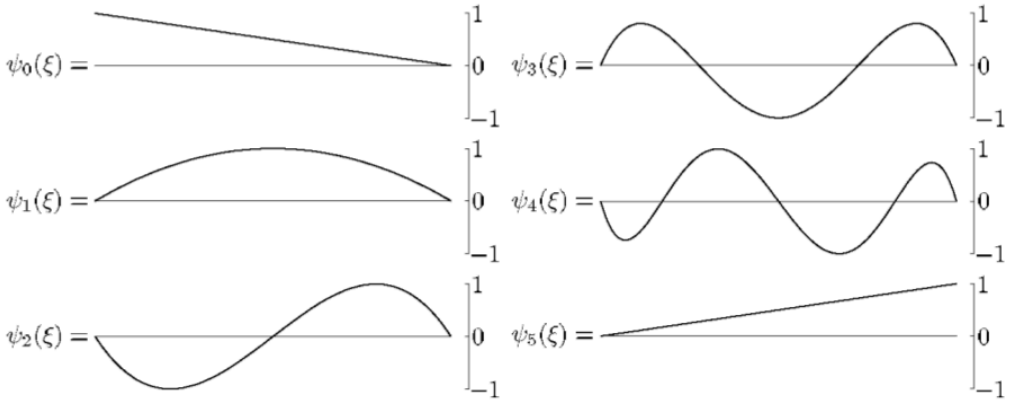

Figure 4.1: Example of expansion basis in 1D for $P = 5$ (see Karniadakis and Sherwin [2005])

Jacobi polynomials see Appendix A. The first six nodes of the basis is shown in Figure 4.1.

Since only the first and the last modes have values on the boundary we can easily distinguish between interior and boundary modes. This property seems to be very efficient when computing with the boundary terms. On the other hand, we lose the complete orthogonality of the basis. However, the computation is still efficient, because the majority of the basis elements are orthogonal.

Term $\left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right)$ corresponds to weight function for the orthogonal relationship of $P^{1,1}_{p-1}(\xi)$.

4.2 Multidimensional expansion basis

Having defined the 1D expansion basis, we proceed to formulate the basis for two dimensions. At first, we introduce two reference elements in the two dimensional space: a quadrilateral and triangular.

Reference quadrilateral element Ω_q and reference triangular elements Ω_t are defined by the following relations:

$$\Omega_q = \{(\xi_1, \xi_2) : -1 \leq \xi_1, \xi_2 \leq 1\}, \tag{4.2}$$

$$\Omega_t = \{(\xi_1, \xi_2) : -1 \leq \xi_1, \xi_2, \xi_1 + \xi_2 \leq 1\}. \tag{4.3}$$

A suitable coordinate system should have independent constant limits. This is the reason for defining the collapsed coordinate system as followings:

$$\begin{aligned} \eta_1 &= 2\frac{1+\xi_1}{1-\xi_2} - 1, \\ \eta_2 &= \xi_2. \end{aligned} \tag{4.4}$$

Then Ω_t can be written in more appropriate form:

$$\Omega_t = \{(\eta_1, \eta_2) : -1 \leq \eta_1, \eta_2 \leq 1\}. \tag{4.5}$$

Having defined the reference elements in the two dimensional space, we can define the expansion bases which is, for quadrilateral elements , nothing that the tensor product of one dimensional bases in x and y direction. In Figure 4.3, we can see the example of a basis

$$\phi_{pq}(\xi_1, \xi_2) = \phi_p(\xi_1)\phi_q(\xi_2) \quad 0 \leq p, q \leq P. \tag{4.6}$$

The collapsed coordinate system is used for the triangular elements, for details see Karniadakis and Sherwin [2005].

4.3 Local element operations

Having defined the polynomial expansions and the reference elements for the two dimensional space, we formulate basic operations such as the integration and differentiation in order to evaluate the terms in (3.46), (3.47), (3.61). The problematics is the simplest in the case of the reference quadrilateral element. The collapsed coordinate system, defined by (4.4), is used for the reference triangular element. This transfers the problem to the previous case. Finally, we use transformation relations to get on the general element.

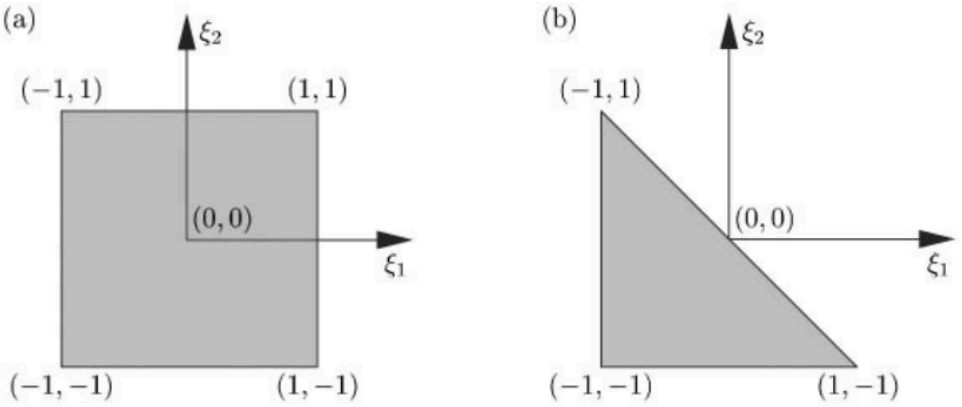

Figure 4.2: The reference elements: (a) quadrilateral, (b) triangular, see Karniadakis and Sherwin [2005].

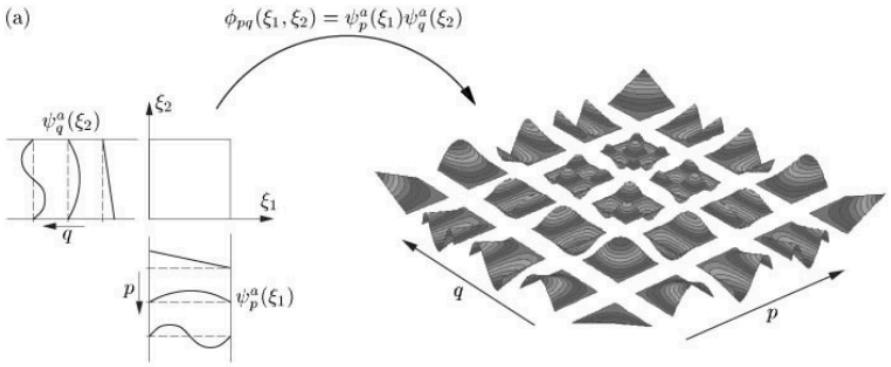

Figure 4.3: Example of an expansion basis on the reference quadrilateral element, see Karniadakis and Sherwin [2005]

4.3.1 Integration over a reference element

Let us have continuous function $\hat{u}(\xi_1, \xi_2)$ defined on reference element Ω_q. We use the Fubini theorem and a numerical quadrature of a given order to obtain a relation for the numerical integration:

$$\int_{\Omega_q} \hat{u}(\xi_1,\xi_2)\ \mathrm{d}\xi_1\ \mathrm{d}\xi_2 = \int_{-1}^{1}\int_{-1}^{1} \hat{u}(\xi_1,\xi_2)\ \mathrm{d}\xi_1\ \mathrm{d}\xi_2 \approx \sum_{i=0}^{Q-1} w_i \sum_{j=0}^{Q-1} w_j \hat{u}(\xi_{ij}), \tag{4.7}$$

where w_i and w_j are the weights of the numerical quadrature, ξ_{ij} are the discrete quadrature points $\xi_{ij} = (\xi_{ij}^{(1)}, \xi_{ij}^{(2)})$ and Q is the numbers of the quadrature points. Inverting relation (4.4) we can transform the triangular system into quadrilateral.

$$\xi_1 = \frac{(1+\eta_1)(1-\eta_2)}{2} - 1, \tag{4.8}$$

$$\xi_2 = \eta_2. \tag{4.9}$$

Then we proceed the same as in the previous case. The only difference is the occurrence of the Jacobian of the transformation, which is easy to evaluate.

$$\begin{aligned}\int_{\Omega_t} \hat{u}(\xi_1,\xi_2)\ \mathrm{d}\xi_1\ \mathrm{d}\xi_2 &= \int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\left|\frac{\partial(\xi_1,\xi_2)}{\partial(\eta_1,\eta_2)}\right|\ \mathrm{d}\eta_1\ \mathrm{d}\eta_2 \\ = \int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\frac{1-\eta_2}{2}\ \mathrm{d}\eta_1\ \mathrm{d}\eta_2 &\approx \sum_{i=0}^{Q-1} w_i \sum_{j=0}^{Q-1} w_j u(\eta_{ij})\frac{1-\eta_{ij}^{(2)}}{2},\end{aligned} \tag{4.10}$$

where $\eta_{ij} = (\eta_{ij}^{(1)}, \eta_{ij}^{(2)})$ represent the quadrature points.

4.3.2 Integration over a general element

Let us assume the mapping from the reference element into general element Ω_k: $\boldsymbol{\chi}^k = (\chi_1^k, \chi_2^k)$: $\Omega_q \to \Omega_k$ for any k. This mapping is given by the following formula:

$$x_i = \chi_i^k(\xi_1,\xi_2) \quad i = 1,2. \tag{4.11}$$

We denote the Jacobi matrix of the mapping as:

$$D_{\Omega_k} = \left\{\frac{\partial \chi_i^k}{\partial \xi_j}\right\}_{i,j=1}^{2}, \tag{4.12}$$

and the determinant:

$$J_{\Omega_k} = \mathrm{det} D_{\Omega_k}. \tag{4.13}$$

Using the substitution theorem, we can easily transform from the general into reference element. The integral over the element Ω_k is given by:

$$\int_{\Omega_k} u(x_1,x_2)\ \mathrm{d}x_1\ \mathrm{d}x_2 = \int_{\Omega_q} \hat{u}(\xi_1,\xi_2) J_{\Omega_k}(\xi_1,\xi_2)\ \mathrm{d}\xi_1\ \mathrm{d}\xi_2. \tag{4.14}$$

The same method is used for the triangular elements.

4.3.3 Differentiation in the reference element

In order to derive a discrete matrix form of the discretization of the Euler and the Navier-Stokes equations, one has to deal with the derivative in the advective and the diffusive terms.

Having the finite dimensional basis, we assume the restriction of $\hat{u}^\delta$ on element to be a linear combination of ϕ_{pq}, $0 \le p,q \le P$ using the coefficients $\hat{u}_{pq} \in \mathbb{R}$. We can make equivalent decomposition using Lagrangian polynomials $h_p(\xi_1), h_q(\xi_2)$ where the vales $\hat{u}_{pq}^\delta$ equal to values of $\hat{u}^\delta$ at the quadrature points.

$$\hat{u}^\delta(\xi_1,\xi_2) = \sum_{p=0}^{P}\sum_{q=0}^{P} \hat{u}_{pq}\phi_{pq}(\xi_1,\xi_2) = \sum_{p=0}^{Q-1}\sum_{q=0}^{Q-1} \hat{u}_{pq}^\delta h_p(\xi_1) h_q(\xi_2), \tag{4.15}$$

where

$$\hat{u}_{pq}^\delta = \hat{u}^\delta(\xi_{pq}) \quad \forall p,q:\ 0 \le p,q \le Q-1, \quad Q > P. \tag{4.16}$$

The value of Q depends on the numerical quadrature. Differentiating this relation with respect to ξ_1, we obtain:

$$\frac{\partial \hat{u}^\delta}{\partial \xi_1}(\xi_1, \xi_2) = \sum_{p=0}^{P} \sum_{q=0}^{P} \hat{u}^\delta_{pq} \frac{\mathrm{d}h_p(\xi_1)}{\mathrm{d}\xi_1} h_q(\xi_2). \tag{4.17}$$

The previous relation holds at every point ξ_1, ξ_2. Using discrete quadrature set of points, we can use properties of the Lagrangian polynomial to obtain the reduced form:

$$\frac{\partial \hat{u}^\delta}{\partial \xi_1}(\xi_{ij}) = \sum_{p=0}^{P} \hat{u}^\delta_{pj} \frac{\mathrm{d}h_p(\xi_1)}{\mathrm{d}\xi_1}\bigg|_{\xi^{(1)}_{ij}}. \tag{4.18}$$

This holds for the quadrilateral element. Again, the formula (4.4) is used to obtain the relation for the triangular elements. The final relation follows from application of the chain rule. For more information see Karniadakis and Sherwin [2005].

4.3.4 Differentiation in a general element

Finally, we have to transform from the general into the reference element. For this purpose, we use the chain rule:

$$\begin{aligned} \frac{\partial u^\delta}{\partial x_1}(x_1, x_2) &= \frac{\partial \xi_1}{\partial x_1}\frac{\partial \hat{u}^\delta}{\partial \xi_1} + \frac{\partial \xi_2}{\partial x_1}\frac{\partial \hat{u}^\delta}{\partial \xi_2}, \\ \frac{\partial u^\delta}{\partial x_2}(x_1, x_2) &= \frac{\partial \xi_1}{\partial x_2}\frac{\partial \hat{u}^\delta}{\partial \xi_1} + \frac{\partial \xi_2}{\partial x_2}\frac{\partial \hat{u}^\delta}{\partial \xi_2}. \end{aligned} \tag{4.19}$$

The derivation with the respect of ξ_1 and ξ_2 was introduced above. The geometric factors $\frac{\partial \xi_1}{\partial x_1}$, $\frac{\partial \xi_1}{\partial x_2}$, $\frac{\partial \xi_2}{\partial x_1}$ and $\frac{\partial \xi_2}{\partial x_2}$ can be easily evaluated, for more information see Karniadakis and Sherwin [2005]. We note that we can use an equivalent formulation:

$$\nabla u^\delta = D^{\mathrm{T}}_{\Omega_k} \hat{\nabla} \hat{u}^\delta, \tag{4.20}$$

where $\hat{\nabla}$ denotes the derivatives with respect to ξ_1 and ξ_2.

The operations presented above are used to evaluate integrals of the following form $\int_{\Omega_k} \frac{\partial u}{\partial x_i} w_i \,\mathrm{dx}$, $\int_{\Omega_k} \frac{\partial u}{\partial x_i}\frac{\partial w}{\partial x_i} \,\mathrm{dx}$, used in the discretizations. Then, we can write:

$$\int_{\Omega_k} \frac{\partial u}{\partial x_i} w_i \,\mathrm{dx} = \int_{\Omega_q} (D^{\mathrm{T}}_{\Omega_k} \hat{\nabla} \hat{u}^\delta)_i \hat{w}^\delta_i J_{\Omega_k}(\xi_1, \xi_2) \,\mathrm{d}\xi_1 \,\mathrm{d}\xi_2, \tag{4.21}$$

$$\int_{\Omega_k} \frac{\partial u}{\partial x_i}\frac{\partial w}{\partial x_i} \,\mathrm{dx} = \int_{\Omega_q} (D^{\mathrm{T}}_{\Omega_k} \hat{\nabla} \hat{u}^\delta)_i (D^{\mathrm{T}}_{\Omega_k} \hat{\nabla} \hat{w}^\delta)_i J_{\Omega_k}(\xi_1, \xi_2) \,\mathrm{d}\xi_1 \,\mathrm{d}\xi_2. \tag{4.22}$$

5. Matrix formulation

In this chapter, the matrix form of the discretization of the Euler and Navier-Stokes equations is briefly summarized.

5.1 Basic notation and backward transformation

In the previous chapter, we defined basis ϕ_{pq}. Then, the restriction of the solution on element Ω_k is of the following form:

$$\hat{u}^\delta(\xi_1, \xi_2) = \sum_{p=0}^{P} \sum_{q=0}^{P} \hat{u}_{pq} \phi_{pq}(\xi_1, \xi_2). \tag{5.1}$$

Further, instead of working with two indices p and q, we will use global index n, $(n = 0, \cdots, (P+1)^2 - 1)$ defined as:

$$n(p, q) = q + (P+1)p. \tag{5.2}$$

Using this notation we obtain:

$$\hat{u}^\delta(\xi_1, \xi_2) = \sum_{n=0}^{N_n - 1} \hat{u}_n \phi_n(\xi_1, \xi_2), \tag{5.3}$$

where $N_n = (P+1)^2$. At this point, we define one more global set of indices m $(m = 0, \cdots, Q^2 - 1)$ as:

$$m(i, j) = i + j \cdot Q. \tag{5.4}$$

Let us introduce two vectors: $\hat{\boldsymbol{u}}^\delta$ and $\hat{\boldsymbol{u}}$. Let $\hat{\boldsymbol{u}}^\delta$ be defined as the values of $\hat{u}^\delta$ evaluated at quadrature of points $\boldsymbol{\xi}_m = (\xi_{ij})$, $m = 0, \cdots, Q^2 - 1$:

$$u_m = u^\delta(\boldsymbol{\xi}_m) = u^\delta(\xi_{ij}), \quad m = 0, \cdots, Q^2 - 1 \tag{5.5}$$

We represent the components of the expansion coefficient vector $\hat{\boldsymbol{u}}$:

$$\hat{u}_n = \hat{u}_{pq}, \quad n = 0, \cdots, (P+1)^2 - 1, \tag{5.6}$$

using the formula (5.3). We define matrix $\mathbb{B} = \{\mathbb{B}_{mn}\}_{m,n=0}^{Q^2-1,(P+1)^2-1}$, which provides the backward transformation from the coefficient space to physical space:

$$\begin{aligned} \mathbb{B}_{mn} = \phi_n(\xi_m) = \phi_{pq}(\xi_{ij}), \quad & m = 0, \cdots, Q^2 - 1, \\ & n = 0, \cdots, (P+1)^2 - 1, \end{aligned} \tag{5.7}$$

then:

$$\hat{\boldsymbol{u}}^\delta = \mathbb{B}\hat{\boldsymbol{u}}. \tag{5.8}$$

Finally, we define weight matrix $\mathbb{W} = \{\mathbb{W}_{mn}\}_{m,n=0}^{Q^2-1,(P+1)^2-1}$, containing the quadrature weights multiplied by the Jacobian at the quadrature points.

$$\begin{aligned} \mathbb{W}_{mn} = J_{ij} w_i w_j \delta_{mn}, \quad & m = 0, \cdots, Q^2 - 1, \\ & n = 0, \cdots, (P+1)^2 - 1, \end{aligned} \tag{5.9}$$

5.2 Differentiation matrix

We recall:

$$\frac{\partial \hat{u}^{\delta}}{\partial \xi_1}(\xi_1, \xi_2) = \sum_{p=0}^{Q_1} \sum_{q=0}^{Q_2} \hat{u}_{pq}^{\delta} \frac{\mathrm{d}h_p(\xi_1)}{\mathrm{d}\xi_1} h_q(\xi_2). \tag{5.10}$$

Then we define differentiation matrix $\mathbb{D}_{\xi_i} = \{[\mathbb{D}_{\xi_i}]_{mm'}\}_{m,m'=0}^{Q^2-1}$, $i = 1, 2$ as:

$$\frac{\partial \hat{\boldsymbol{u}}^{\delta}}{\partial \xi_i} = \mathbb{D}_{\xi_i} \hat{\boldsymbol{u}}^{\delta} \quad i = 1, 2. \tag{5.11}$$

where the elements satisfy:

$$[\mathbb{D}_{\xi_1}]_{mm'} = \left. \frac{\mathrm{d}h_r(\xi_1)}{\mathrm{d}\xi_1} \right|_{\xi_{mm'}^{(1)}} h_s(\xi_{mm'}^{(2)}), \tag{5.12}$$

where

$$m'(r, s) = r + s \cdot Q, \quad r, s = 0, \cdots, Q. \tag{5.13}$$

We introduce auxiliary matrix function $\boldsymbol{\Lambda}(f)$ which evaluates arbitrary function f at quadrature points:

$$[\boldsymbol{\Lambda}(f(\xi_1, \xi_2))]_{mn} = f(\xi_{ij}^{(1)}, \xi_{ij}^{(2)}) \delta_{mn}. \tag{5.14}$$

Using the chain rule, we can write the differentiation in the discrete matrix form:

$$\frac{\partial \boldsymbol{u}^{\delta}}{\partial x_i} = \left[\Lambda \left(\frac{\partial \xi_1}{\partial x_i} \right) \mathbb{D}_{\xi_1} + \Lambda \left(\frac{\partial \xi_2}{\partial x_i} \right) \mathbb{D}_{\xi_2} \right] \hat{\boldsymbol{u}}^{\delta} := \mathbb{D}_{x_i} \hat{\boldsymbol{u}}^{\delta} \quad i = 1, 2. \tag{5.15}$$

5.3 Matrix form of the approximative solution

In this section, we formulate the discontinuous Galerkin matrix form of the Euler and Navier-Stokes equations. Elemental mass matrix $\mathbb{M}$ consists of scalar products of test functions. Using a numerical quadrature, the mass matrix can be written as:

$$\mathbb{M} = \mathbb{B}^T \mathbb{W} \mathbb{B}. \tag{5.16}$$

Having defined all operations in discrete a matrix formalism, we can formulate the discrete form of the discretization of the governing equations. Firstly, we consider the Euler equations. The inner product of the test functions and the basis functions is responsible for the presence of mass matrix $\mathbb{M}$. This matrix needs to be inverted in oder to obtain the demanded initial value problem. The derivatives of the test functions are performed using (5.15). Terms $\boldsymbol{b}, \hat{\boldsymbol{b}}$ and $\tilde{\boldsymbol{b}}$ are responsible for the evaluation of the surface integral using corresponding numerical flux, for more information see Karniadakis and Sherwin [2005].

$$\frac{\mathrm{d}\hat{\boldsymbol{w}}}{\mathrm{d}t} = \mathbb{M}^{-1}[(\mathbb{D}_{x_1}\mathbb{B})^T \mathbb{W} \Lambda(\boldsymbol{f}_1(\hat{\boldsymbol{w}}^{\delta})) + (\mathbb{D}_{x_2}\mathbb{B})^T \mathbb{W} \Lambda(\boldsymbol{f}_2(\hat{\boldsymbol{w}}^{\delta}))] - \mathbb{M}^{-1}\boldsymbol{b}. \tag{5.17}$$

Further, we formulate the matrix formulation of the discretization of the Navier-Stokes equations.

$$\frac{\mathrm{d}\hat{\boldsymbol{g}}}{\mathrm{d}t} = \mathbb{M}^{-1}[(\mathbb{D}_{x_1}\mathbb{B})^T \mathbb{W} \Lambda(\hat{\boldsymbol{W}}^{\delta})) + (\mathbb{D}_{x_2}\mathbb{B})^T \mathbb{W} \Lambda(\hat{\boldsymbol{W}}^{\delta}))] - \mathbb{M}^{-1}\hat{\boldsymbol{b}}, \tag{5.18}$$

$$\frac{\mathrm{d}\hat{\boldsymbol{w}}}{\mathrm{d}t} = \mathbb{M}^{-1}[(\mathbb{D}_{x_1}\mathbb{B})^T\mathbb{W}\Lambda(\boldsymbol{f}_1(\hat{\boldsymbol{w}}^\delta))) + (\mathbb{D}_{x_2}\mathbb{B})^T\mathbb{W}\Lambda(\boldsymbol{f}_2(\hat{\boldsymbol{w}}^\delta)))] - \mathbb{M}^{-1}\boldsymbol{b} + \\ \mathbb{M}^{-1}[(\mathbb{D}_{x_1}\mathbb{B})^T\mathbb{W}\Lambda(\boldsymbol{R}_1(\hat{\boldsymbol{w}}^\delta, \hat{\boldsymbol{g}}^\delta))) + (\mathbb{D}_{x_2}\mathbb{B})^T\mathbb{W}\Lambda(\boldsymbol{R}_2(\hat{\boldsymbol{w}}^\delta, \hat{\boldsymbol{g}}^\delta)))] - \mathbb{M}^{-1}\tilde{\boldsymbol{b}}. \tag{5.19}$$

6. Time integration

In this chapter we deal with the solution of a time integration problem. As we have shown (see (3.46), (3.47)), the discretization of the Euler and Navier-Stokes equation leads to a problem, which can be formally written as:

$$\begin{aligned} \frac{d\boldsymbol{y}}{dt} &= \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{y}(t_0) &= \boldsymbol{y_0}. \end{aligned} \tag{6.1}$$

There are two approaches how to solve the problem: an explicit and implicit method. The advantage of explicit methods is the ability of a parallelization and the absence of solving an implicit problem. However, unlike implicit methods, explicit methods require very small time step. In our case the time integration is performed by the explicit Runge-Kutta method.

6.1 Runge-Kutta method

Let us have a partition of time interval $[0, T]$:

$$0 = t_0 < t_1 < t_2 < \cdots < t_R = T, \tag{6.2}$$

satisfying:

$$t_k = t_{k-1} + \Delta t. \tag{6.3}$$

where Δt is the time step. We denote $\boldsymbol{y}_n$ as an approximation of the solution in time t_n:

$$\boldsymbol{y}_n \approx \boldsymbol{y}(t_n). \tag{6.4}$$

Let s be the stage of the method with coefficients a_{ij}, b_i, c_i, $i, j = 1, \cdots, s$, then:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t \sum_{i=i}^{s} b_i \boldsymbol{\kappa}_i, \tag{6.5}$$

where

$$\boldsymbol{\kappa}_i = \boldsymbol{f}(t_n + c_i \Delta t, \boldsymbol{y}_n + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{\kappa}_j), \quad i = 1, \cdots, s. \tag{6.6}$$

In the work we used the 4th-order Runge Kutta method given by the following Butcher table:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} . \tag{6.7}$$

6.2 Application in the discretization

Having defined approximative discrete solutions, we show the application of the time integration. The situation is straightforward for the Euler equations. We obtain the solution in the next time step using see (3.46). In the case of the Navier-Stokes equations the process is as follows:

- Having solution $\boldsymbol{w}_k^\delta$ in t_k we evaluate $\boldsymbol{W}_k^\delta$ using (3.49).
- Using $\boldsymbol{W}_k^\delta$ and (3.47) we compute $\boldsymbol{g}_{i,k}^\delta, \quad \forall i = 1, 2, 3.$
- Using $\boldsymbol{g}_{i,k}^\delta$ and (3.48) we obtain $\boldsymbol{w}_{k+1}^\delta$.

In the work we used a fixed time step short enough not to let a solution diverge. We had to use a smaller time step for parallel computations.

7. Nektar++

In the work, we used open-source cross platform framework Nektar++. Nektar++ is designed to provide the spectral/hp element discretization. The software is mainly developed by members of the SherwinLab at Imperial College London and Kirby's group at the University of Utah.

Nektar++ contains various solvers. Except for the compressible flow solver, it provides, for example, an incompressible solver or linear elasticity solver. It supports only the discontinuous projection for the compressible flow solver, but supports both the continuous and discontinuous for other (e.q. incompressible flow solver). One, two and three dimensional problems are supported.

The computation requires two source files: a mesh file and a session file, both in XML file format. The mesh file specifies vertices, edges, elements, curved edges, a computational domain and a polynomial expansion. In the session file, we specify all other parameters, including solver informations, boundary and initial conditions.

Apart from the final solution file in fld format, Nektar++ can generate chk files to check the current solution. These files can be converted in vtu files, using Nektar++ utility FieldConvert, in order to display the results. For this purpose, we used open source software Paraview.

Nektar++ framework is developed in C++ language, which combines the high performance speed and the objected oriented programming concepts. The architecture is build on six libraries, four of them are responsible for the numerical discretization and function representation:

- StdRegions - Represents the reference element and basic operations on it.
- SpatialDomains - Represents the mapping between the element of the mesh and the reference element.
- LocalRegions - Local representation and operations on each element.
- GlobalRegions - Global representation.

LibUtils library contains the basic building blocks and SolverUtils library contains a solver representation.

For more information, see Cantwell et al. [2015] and for the documentation, see Nek [2017].

Nektar++ allows parallel computing, which is very useful when carrying out the computation on a cluster and we have multiple cores available. We performed parallel computations on Karlin computational cluster Snehurka.

Figure 7.1: Nektar++, see Nek [2017]

8. Numerical experiments

One of the standard test case is the simulation of the flow around NACA airfoil. In the work, we used NACA0012 airfoil, because it is one of the most common and there are many results available, for example see Norbert et al. [2010].

This chapter is organized as follows. In Section 8.1, we cover the process of the mesh generation. Further, we present the results of the numerical simulations. At first, we study the inviscid subsonic flow (see Section 8.2), the values of the lift and drag coefficients, and the quality of the solution, depending on the mesh and polynomial expansion. The results of the simulations of the inviscid transonic flow are presented in Section 8.3. We discuss the shock capturing method and show the impact on the solution. Further, in Sections 8.4 and 8.5, the viscous subsonic flow is studied. We discuss the simulations of the boundary layer with respect to the polynomial degree expansion and the size of boundary layer elements. Finally, we present the results for an unsteady flow for both the subsonic and transonic velocity, see Section 8.6.

8.1 High order mesh generation

A coarseness of the mesh plays a crucial role in the finite volume schemes, because we approximate the solution function by a constant function on each element. If we aim to obtain an accurate solution we need to use a sufficiently fine mesh. In the case of higher order methods, the situation is slightly different. Even with a coarse mesh we can obtain a good solution providing that we use a sufficiently high polynomial order expansion. However, if we use high order methods it is convenient to use a high order curvilinear mesh with curved boundaries. This may be challenging for more complex geometries, especially in industry, where geometries can be very complex.

The creation of the mesh was done as follows: Firstly, we generated mesh points for 2D domain with NACA0012 profile, using software E2D2, see Feistauer et al. [1992], and created a simple linear mesh. In the second step, we enhanced the mesh by creating curved boundaries. This was done by finding boundary points that divide the length of the corresponding curved side in the ratio that corresponds to the zero points of GLL quadrature, see Figure 8.1. The Gibbs phenomenon is eliminated for this choice of points. This is done by dividing the

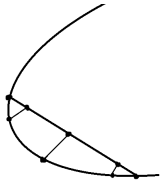

Figure 8.1: The creation of a high order mesh

linear mesh in a corresponding ratio and then finding the closest point on the profile, for each point. Then we had to transform the points into the XML file, which is suitable for Nektar++.

The last step was to make a boundary layer. For this purpose, Nektar++ provides utility NekMesh, which allows us to create the boundary layer. We had also split the elements behind the profile, in order to obtain a structured mesh, see Figure 8.2. For this purpose, we had to slightly modify the source code of utility NekMesh in Nektar++. This utility was also used for additional local refinements of the mesh (e.g. in the area of the shock wave, see Figure 8.8).

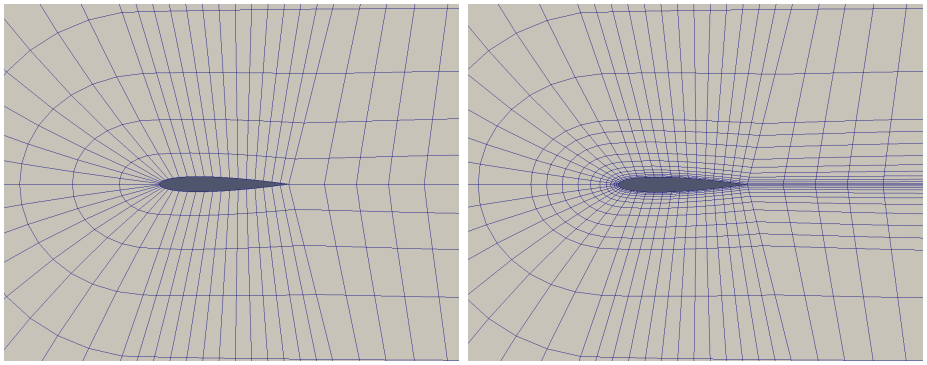

Figure 8.2: A basic mesh (left), a mesh with a boundary layer (right)

8.2 Inviscid flow, M = 0.5 $\alpha = 2°$

The first test case was the inviscid subsonic flow with Mach number $M = 0.5$ and angle of attack $\alpha = 2°$. We tested various meshes denoted as M1, M2, M3, M4 with the number of elements equals $N = 420, N = 918, N = 1540, N = 2915$, respectively. Since no boundary layer is occurring for the inviscid flow, we did not have to create a thin boundary layer in the mesh and we used thin elements only in the top of the airfoil. For this reason, the time step could be larger than for the viscous flow and we were able to compute the steady solution for high polynomial order expansion. The aforementioned meshes are displayed in Figure 8.3. We used the polynomial expansion up to the sixth degree and computed the lift and the drag coefficients (c_l and c_d, respectively) according to the following relation:

$$\begin{pmatrix} c_l \\ c_d \end{pmatrix} = \frac{1}{\frac{1}{2}\rho_\infty \left|\boldsymbol{v}_\infty\right|^2 L} \int_{\Gamma^W} (p\mathbb{I} - \mathbb{T})\boldsymbol{n} \, \mathrm{d}S, \tag{8.1}$$

where ρ_∞ is the far-field density, $\boldsymbol{v}_\infty$ is the far-field velocity and L is the reference length, Γ^W is the boundary of the profile, p is the pressure, $\mathbb{I}$ is the identity matrix, $\mathbb{T}$ is the Cauchy stress tensor and $\boldsymbol{n}$ is the normal vector with respect to the boundary. We implemented this relation in Nektar++ as a part of the postprocessig. We used the exact Riemann solver for the computation of the advective numerical flux (for details see Section 3.4.1)

We carried out the computations and compared the convergence of the solution with respect to the mesh refinement and polynomial order expansion. The results, summarized in Table 8.1 and Graph 8.4, show that the values of the lift and drag coefficients converge to values:

$$c_d \approx 0.0018,$$

$$c_l \approx 0.247.$$

We can see that the value of coefficient c_d is very small. This is what we expected, since we work with the inviscid flow.

In Figure 8.5, we can see the isolines of the Mach number of the solution. On the left, we used mesh M1 and the third polynomial degree and on the right we used the first polynomial degree and mesh M3. From the figure, we can see that even though the number of degrees of freedom is similar, the result for the third degree expansion is better and much smoother. This fact illustrates the benefit of using the higher order polynomial expansion.

In Figure 8.6, we compared expansion coefficients $\hat{u}_{pq}$ of the solution functions in various elements. We used mesh M1 and the sixth polynomial order expansion. In the top picture, we can see the expansion coefficients of the solution function at the top of the airfoil. The second picture is related to the second layer at the top of the profile. The last figure shows the solution on the side of the profile. The elements, where the expansion coefficients are studied are displayed in Figure 8.7. We observe, that coefficients for the element at the top of the airfoil have the largest values, because of the largest gradients in the solution. On the other hand, the coefficients for the element on the side of the profile are relatively small.

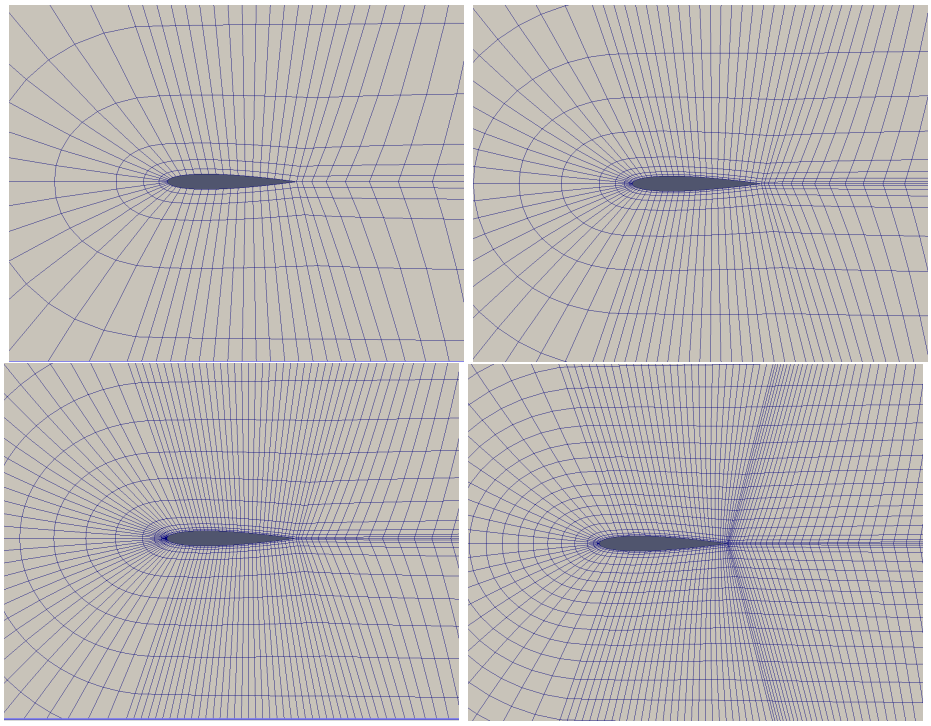

Figure 8.3: Meshes used for inviscid flow: M1 (top left), M2 (top right), M3 (lower left), M4 (lower right)

8.3 Inviscid flow, M = 0.85

The second test case was the transonic inviscid flow with the Mach number $M = 0.85$ and no angle of attack. The occurrence of a shock wave and is typical for the transonic flow. The shock waves are characterized by an abrupt change of the conservative variables. Therefore, we refined the elements where the shock wave

p	mesh	DOF	c_d	c_l
1	M1	1680	-0.0004769	0.209294
1	M2	3672	-0.0051603	0.236006
1	M3	6160	0.03424510	0.238760
1	M4	11664	-0.0080630	0.240999
2	M1	3780	0.00092069	0.235886
2	M2	8262	0.00193734	0.245964
2	M3	13860	0.00200521	0.246749
2	M4	26244	0.00190760	0.247049
3	M1	6720	0.00179973	0.244322
3	M2	14688	0.00190379	0.247680
3	M3	24640	0.00191845	0.247503
3	M4	46656	0.00185410	0.247624
4	M1	10500	0.00195637	0.246686
4	M2	22950	0.00186030	0.247539
4	M3	38500	0.00185072	0.247406
4	M4	72900	0.00180892	0.247647
5	M1	15120	0.00192342	0.247357
5	M2	33048	0.00183330	0.247428
5	M3	55440	0.00184208	0.247337
5	M4	104976	0.00180748	0.247611
6	M1	20580	0.00189227	0.247484
6	M2	44982	0.00185158	0.247346
6	M3	75460	0.00178361	0.247278
6	M4	142884	0.00180395	0.247677

Table 8.1: c_d and c_l for different meshes and polynomial orders for inviscid flow $M = 0.5$, $\alpha = 2^\circ$

occurs. The mesh (see Figure 8.8) we used was compounded of 1378 elements and we used the second degree polynomial expansion. We tried using a higher degree, but the computational time was too high to obtain a steady solution. Because the mesh at the area of the shock wave is very fine, we had to use a small time step.

Further, we also used the shock capturing method (for details see Section 3.9) and compared the solution with and without the shock capturing technique. In formula (3.51), we introduced the sensor variable, which is useful to indicate the occurrence of the shock wave. The values of the sensor variable are depicted in Figure 8.11. The values of the artificial diffusion, also introduced in Section 3.9, are displayed in Figure 8.12. We observe that the artificial diffusion occurs only in the area of the shock wave. We studied the values of the Mach number and the pressure coefficient on the boundary of the airfoil. The pressure coefficient is defined as:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty \left|\boldsymbol{v}_\infty\right|^2}, \tag{8.2}$$

where p_∞ is the far-field pressure, ρ_∞ is the far-field density and $\boldsymbol{v}_\infty$ is the far-field velocity. We implemented the computation of C_p in Nekar++ as a part of the postprocessing. The results are depicted in Figure 8.10. In the top graph, the

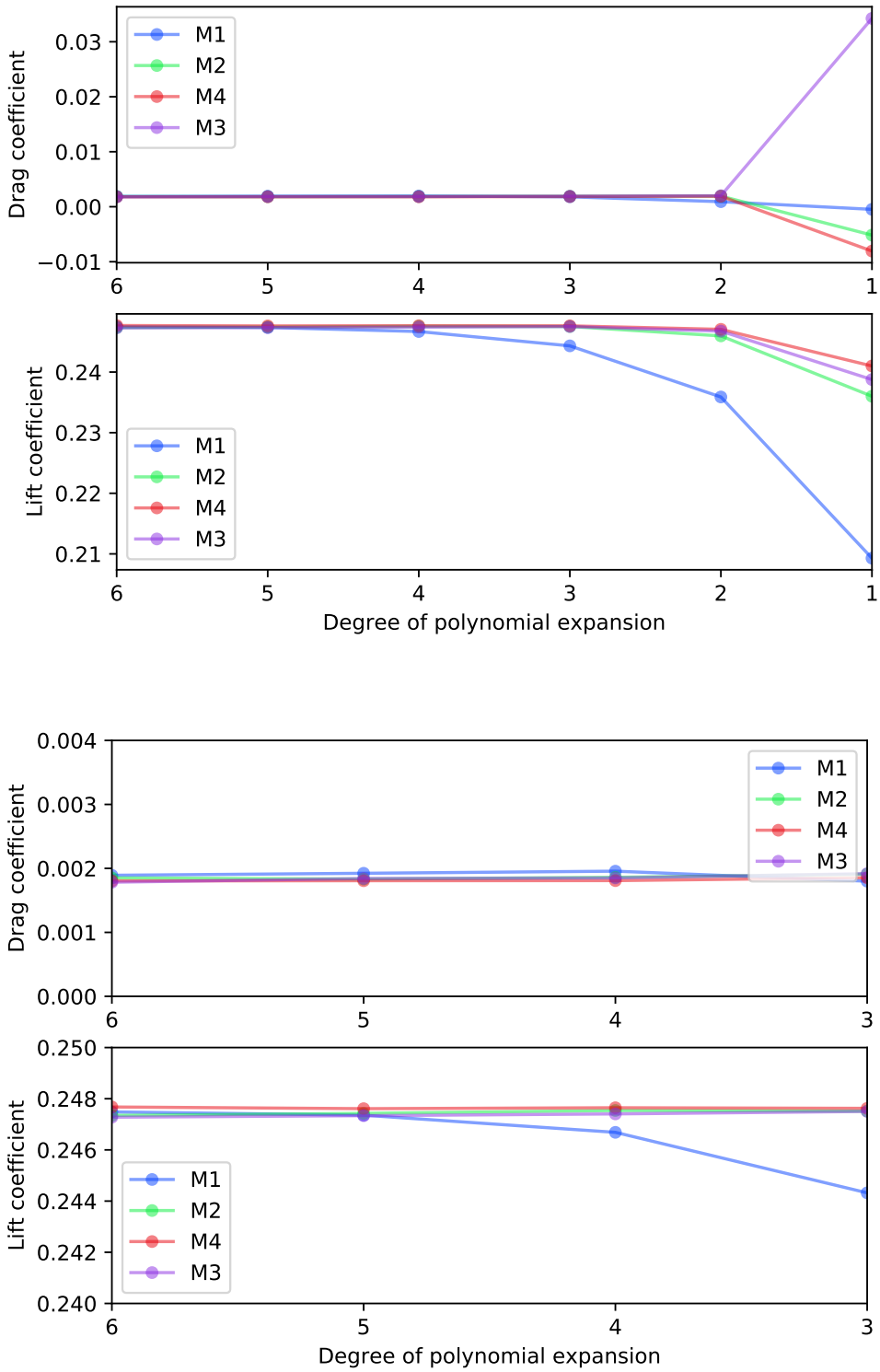

Figure 8.4: Dependence of the lift and drag coefficients on the polynomial degree for various meshes for inviscid flow $M = 0.5$, $\alpha = 2°$

results without the shock capturing technique are shown. We can see a substantial oscillation of the Mach number and the pressure coefficient at the point of the shock. This oscillation doesn't represent a physically correct solution. In the

Figure 8.5: Isolines of the Mach number for mesh M1 and third polynomial order (left), and M3 and first polynomial order (right) - inviscid flow $M = 0.5$, $\alpha = 2°$

lower graph, the results with the shock capturing technique are displayed and we can see that the oscillations at the point of the shock are suppressed. The isolines of the Mach number can be seen in Figure 8.9.

8.4 Viscous flow, M = 0.5, Re = 500, $\alpha = 2°$

The next test case was the viscous flow with Mach number $M = 0.5$, Reynolds number $Re = 500$ and angle of attack $\alpha = 2°$. The viscous flow is characterized by a presence of the boundary layer in order to satisfy the no slip boundary condition. Therefore, we had to refine the elements on the boundary of the profile. Considering $Re = 500$, the boundary layer is relatively thick, compared to higher Reynolds numbers. We used three meshes, denoted as M5, M6, M7 with the number of elements equals $N = 754$, $N = 1178$, $N = 1368$ and the Roe advection numerical flux (for details see Section 3.4.2).

Firstly, we studied the computation of the lift and drag coefficients. The results are summarized in Table 8.2. The steady state was achieved using up to the fourth degree of the polynomial expansion for mesh M5 and up to the third degree for mesh M6 and M7. From the table, we observe that the coefficients converge to the values:

$$c_d \approx 0.18,$$

$$c_l \approx 0.11.$$

In Figures 8.13, 8.15 we can see the isolines of the Mach number for mesh M5 and in Figures 8.14, 8.16 we can see the isolines of the Mach number for mesh M6. We observe that the boundary layer is developed for both meshes and the solution of the fourth order is very smooth.

Further we studied the simulation of the boundary layer. In Figure 8.17 we can see the isolines of the Mach number corresponding to the boundary layer and the mesh. The solution was obtained using first, second, third and fourth polynomial degree (top left, top right, lower left, lower right, respectively). We observe, that for the first degree, the boundary layer is not fully developed and is the same size as the the first boundary element. We can see, that the result

p	mesh	DOF	c_d	c_l
1	M5	3016	0.1712074	0.11092685
1	M6	4712	0.1614234	0.10908513
1	M7	5472	0.1611628	0.11886311
2	M5	6786	0.1909813	0.10865095
2	M6	10600	0.1702229	0.11140273
2	M7	12312	0.1687561	0.11358470
3	M5	12064	0.1854102	0.11187565
3	M6	18840	0.1903322	0.11334308
3	M7	21888	0.1897796	0.11312697
4	M5	18850	0.1850332	0.11205576

Table 8.2: c_d and c_l for different meshes and polynomial orders for the viscous flow, $M = 0.5$, $Re = 500$, $\alpha = 2°$

is much better even for the second degree. The thickness of the boundary layer is smaller than the first boundary element, but fully developed boundary layer is obtained for the third and fourth polynomial degree expansion. The fact that we are able to obtain a thinner boundary layer than the first boundary element is an another advantage of the high order methods. It allows us to use larger elements to simulate the boundary layer. The mesh can be coarser and the time step can be higher.

8.5 Viscous flow, M = 0.5, Re = 2000, $\alpha = 2°$

Using higher Reynolds number requires smaller elements near the airfoil because the boundary layer is thinner. The mesh used for the computation is displayed in Figure 8.18. We used HLLC Riemann solver for the computation of the advective numerical flux and the first, second and third polynomial degree. The isolines of the Mach number are displayed in Figure 8.19. The boundary layer is fully developed and the solution is relatively smooth, for the third degree.

8.6 Unsteady flow

The last simulations concern with an unsteady flow. The first test case was the simulation of the viscous flow with $M = 0.5$, $Re = 5000$ and the angle of attack was $\alpha = 2°$. We used the mesh containing $N = 806$ elements and second polynomial order expansion. In Figure 8.20 the values and isolines of the Mach number are displayed.

The second test case was the viscous flow with $M = 0.85$, $Re = 10000$ and no angle of attack. We used the mesh containing $N = 1060$ elements and second polynomial order expansion. In Figure 8.21 we can see the results of the solution. We note that we did not present the shock capturing method, because adding an another diffusive term did not lead to better results.

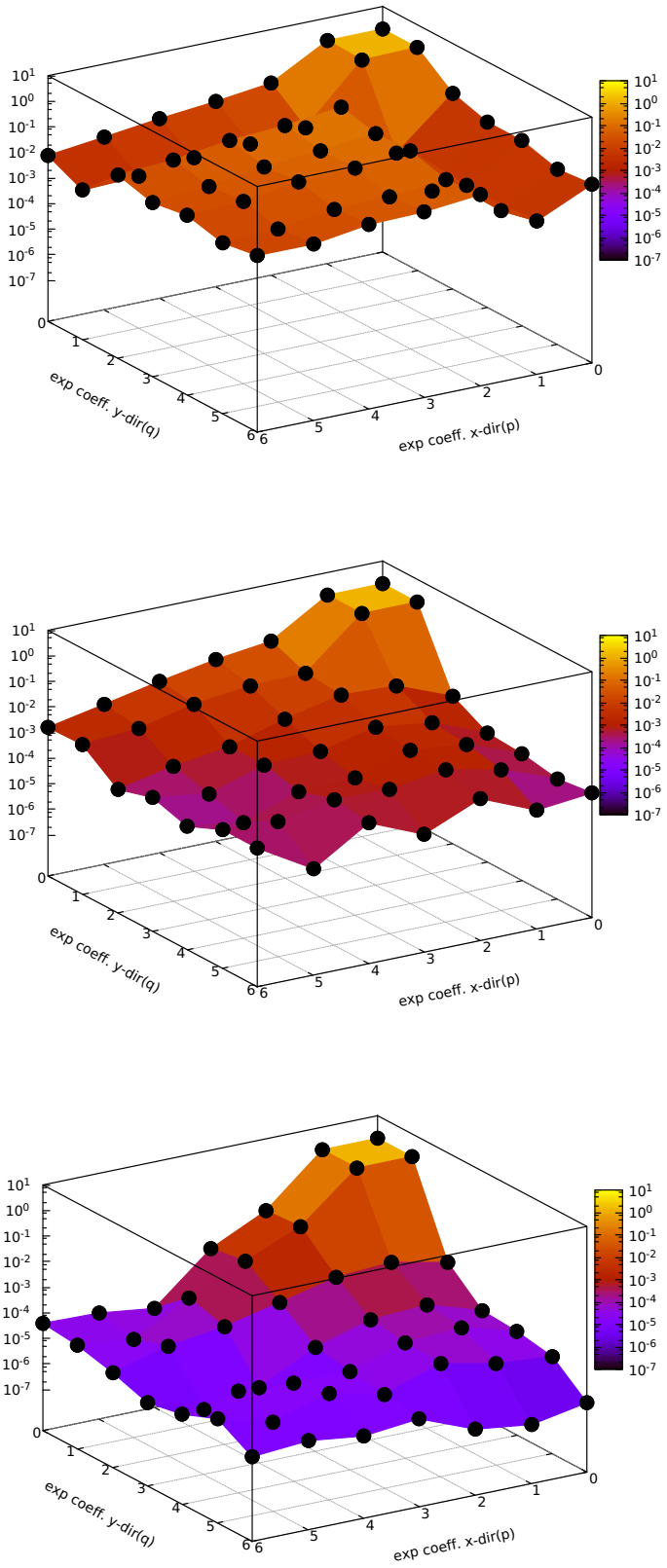

Figure 8.6: Polynomial expansion coefficients for the element from the first boundary layer at the top of the airfoil (top), second boundary layer at the top of the airfoil (middle) and boundary layer on the side of the airfoil (lower), for the elements see 8.7

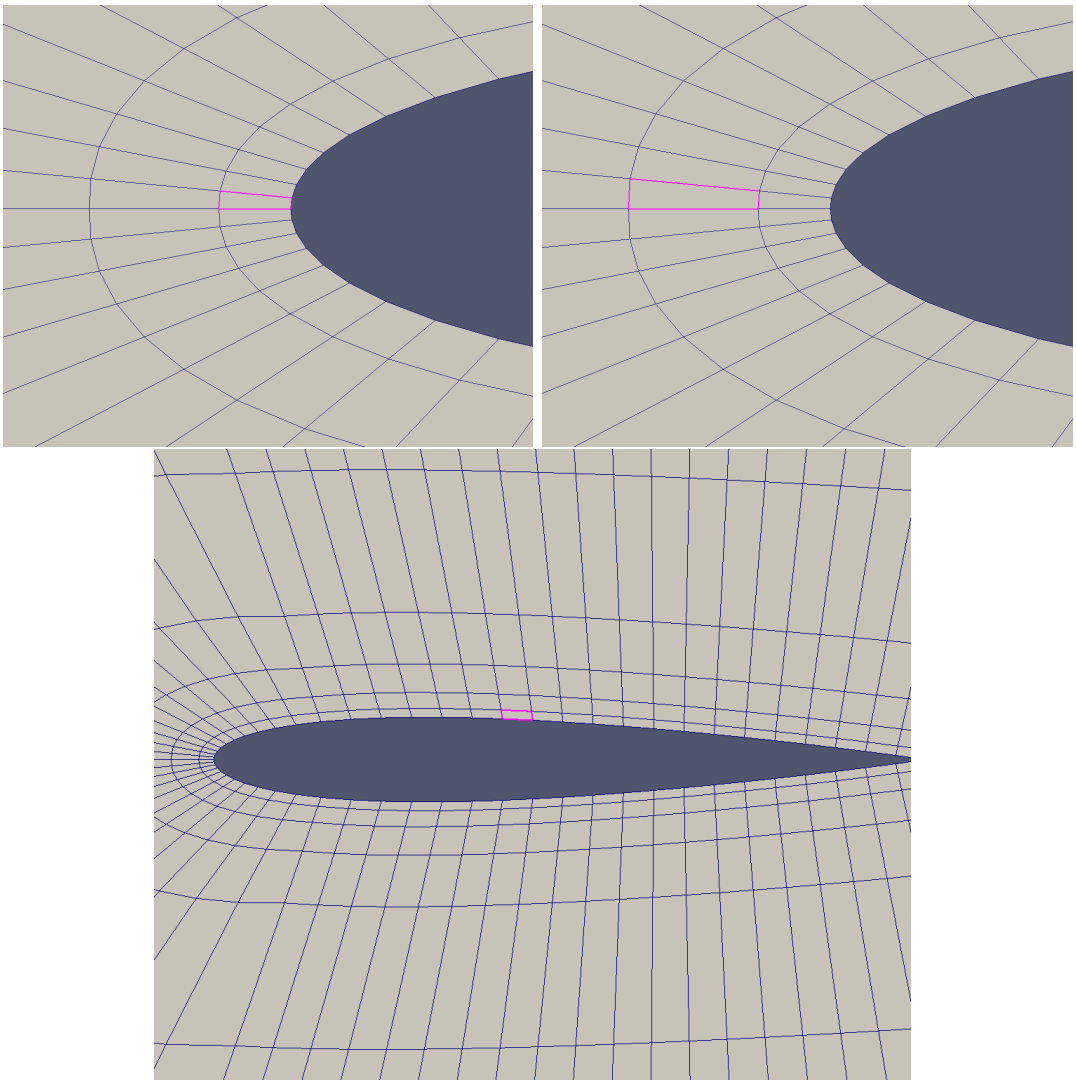

Figure 8.7: Elements, where the expansion coefficients are studied, see Figure 8.6.

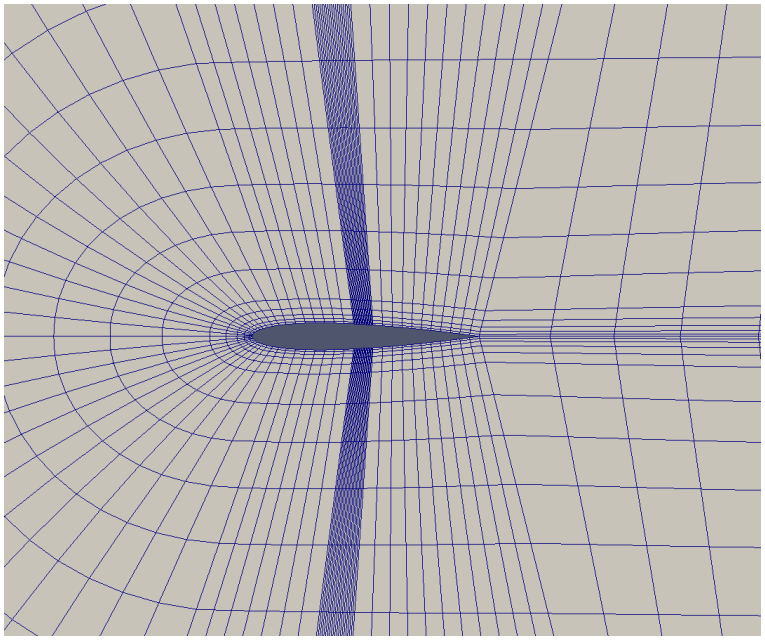

Figure 8.8: The mesh used for the transonic inviscid flow

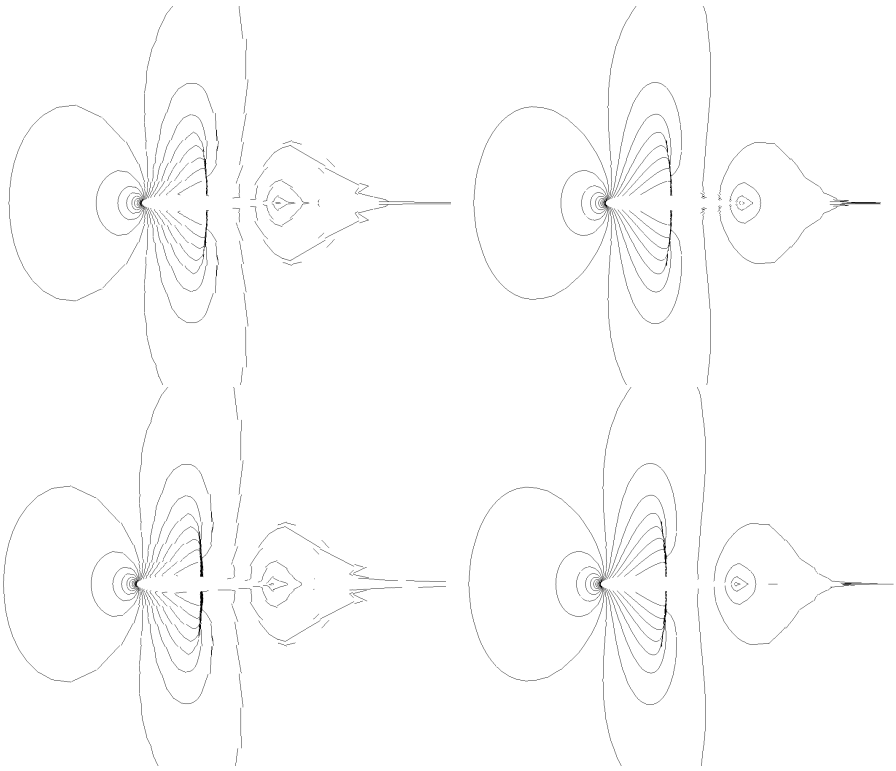

Figure 8.9: Isolines of the Mach number without the shock capturing using first order (top left), second order (top right) and with the shock capturing using first order (lower left) second order (lower right)

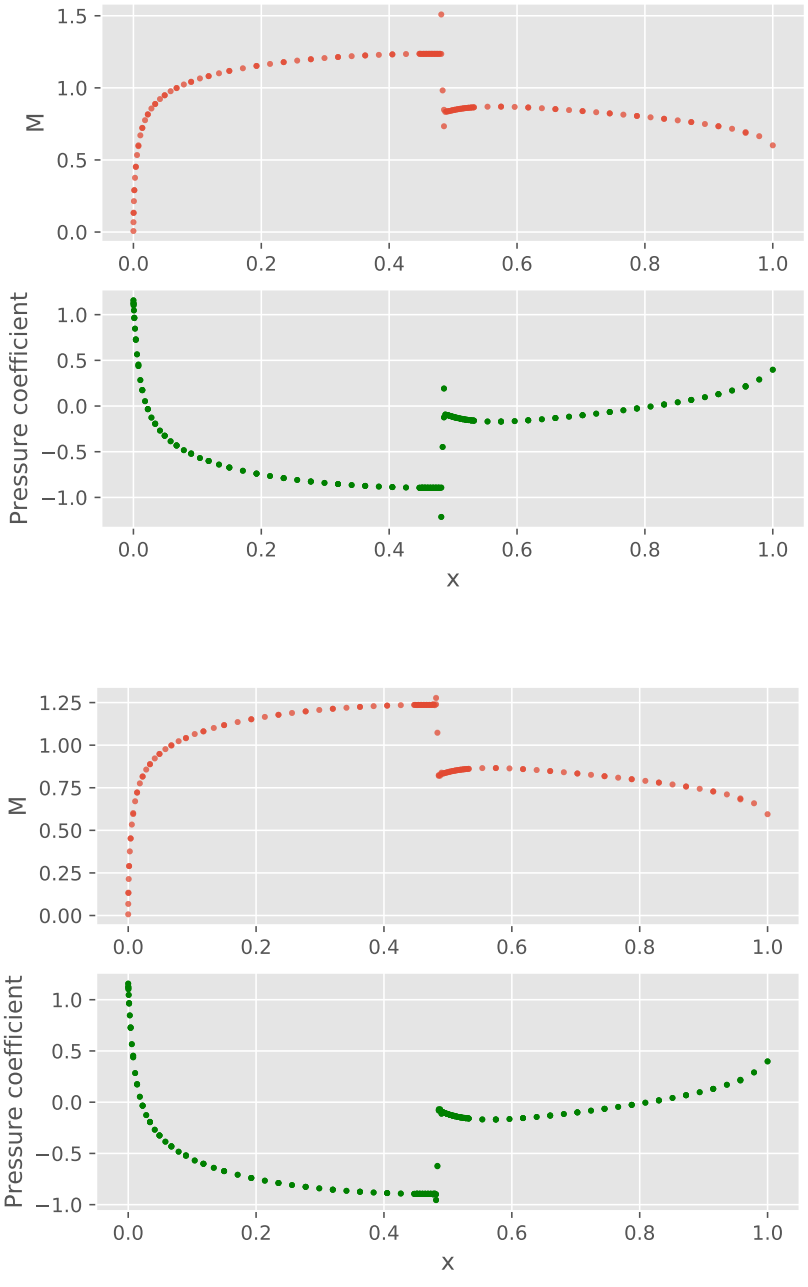

Figure 8.10: Dependence of the Mach number and the pressure coefficients on distance x on the airfoil (top of the airfoil: $x = 0$) without the shock capturing (top) and with the shock capturing method (lower)

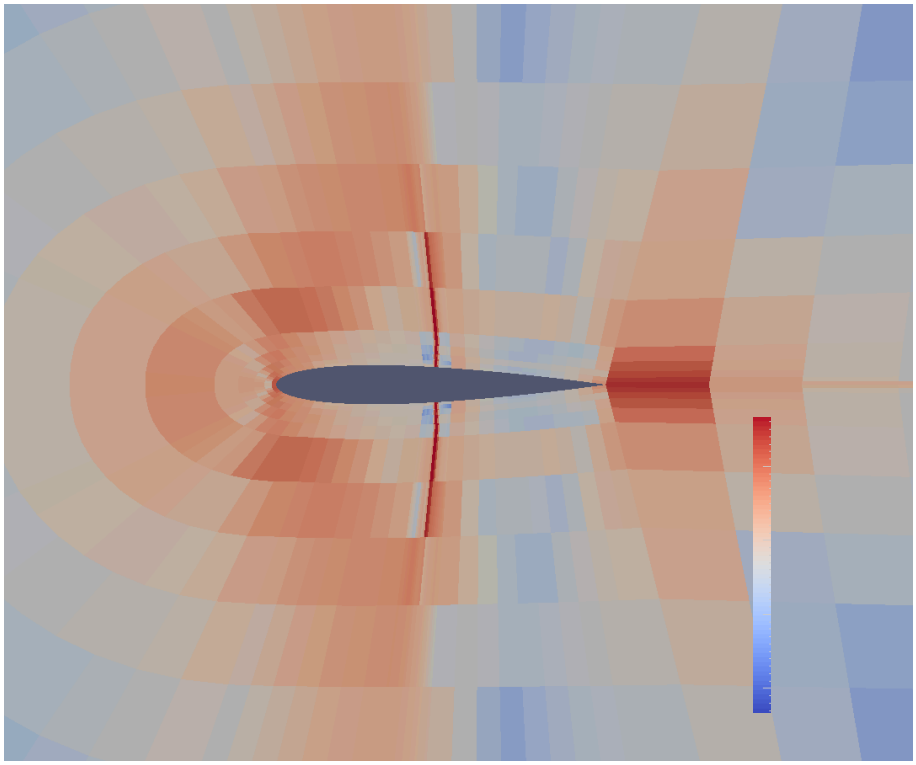

Figure 8.11: The values of the sensor variable for the inviscid transonic flow

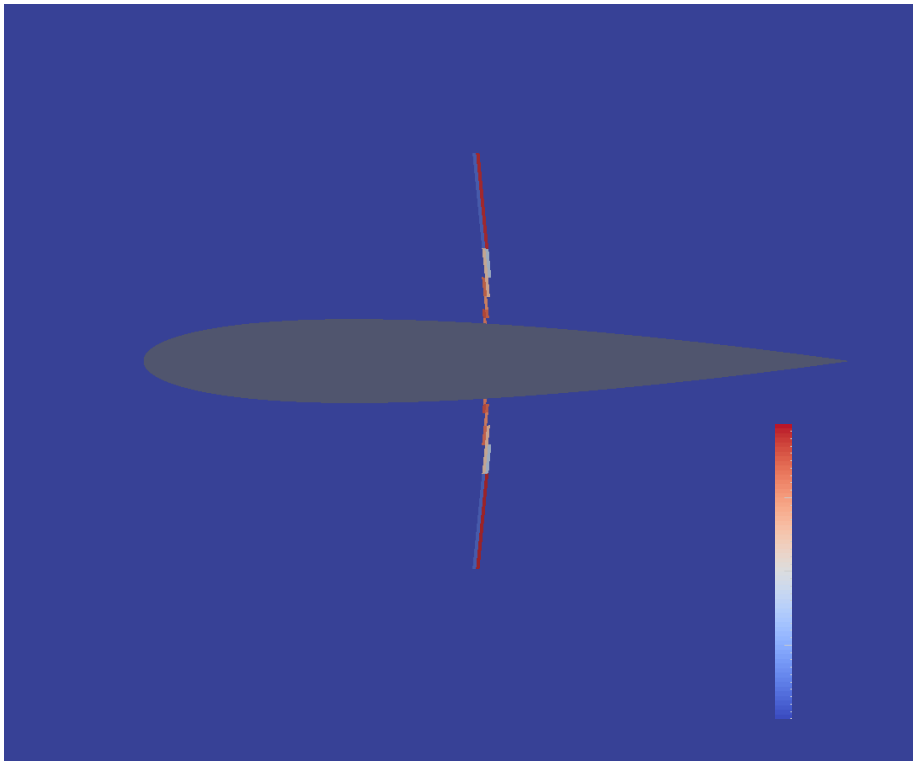

Figure 8.12: The values of the artificial viscosity for the inviscid transonic flow

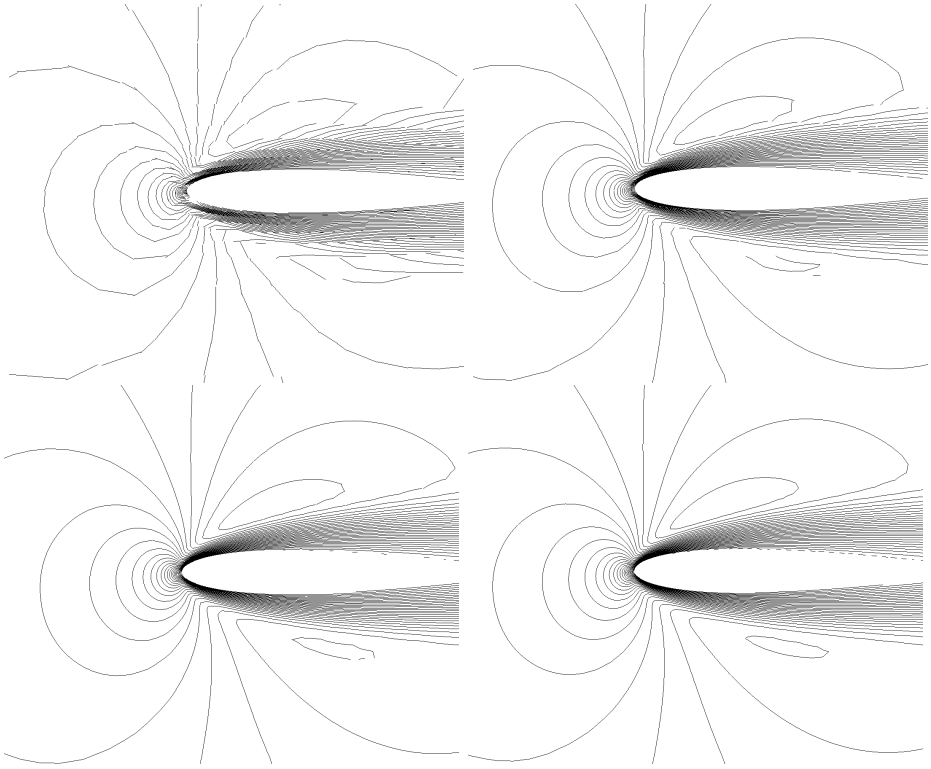

Figure 8.13: Isolines of the Mach number for mesh M5 using first (top left), second (top right), third (lower left) and fourth (lower right) degree for the viscous flow, $M = 0.5$, $Re = 500$, $\alpha = 2^\circ$

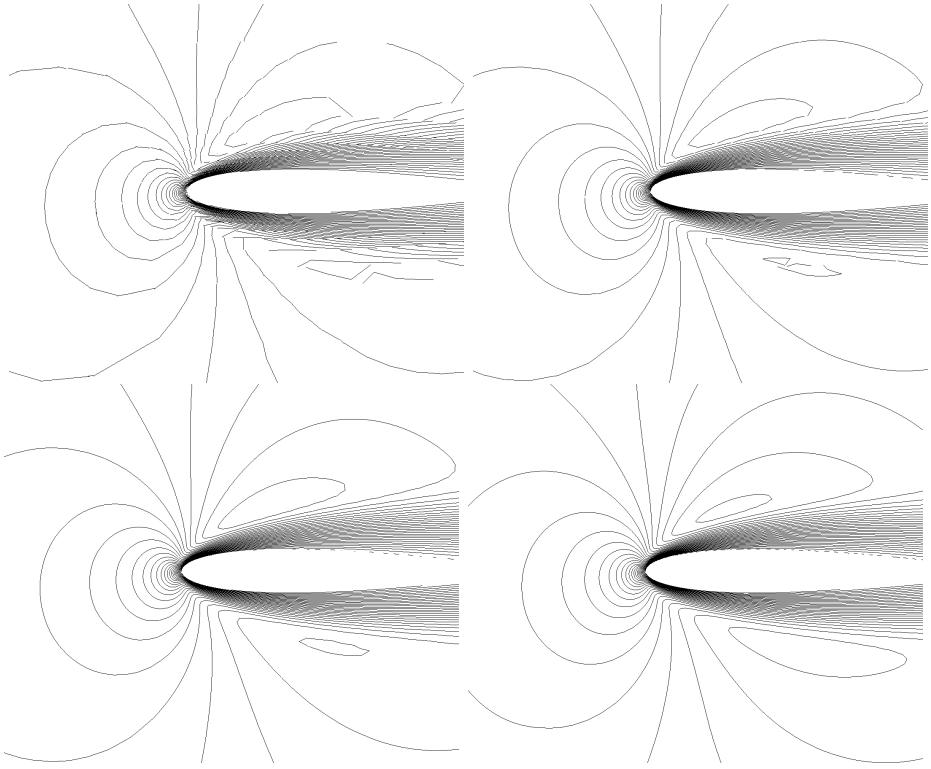

Figure 8.14: Isolines of the Mach number for mesh M6 using first (top left), second (top right), third (lower left) and fourth (lower right) degree for the viscous flow, $M = 0.5$, $Re = 500$, $\alpha = 2^\circ$

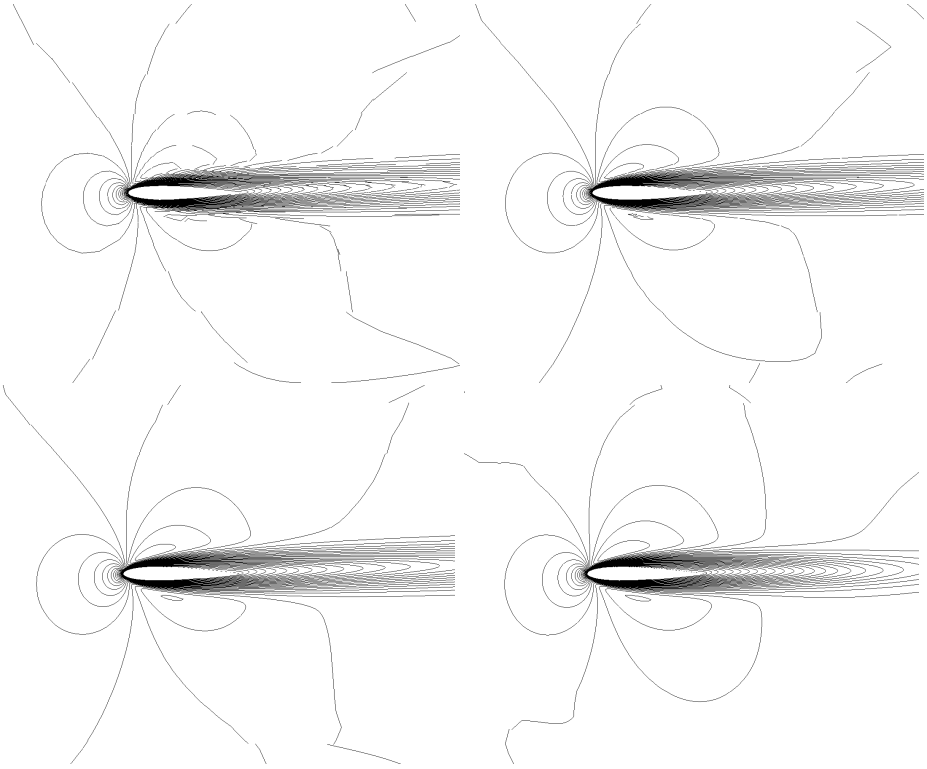

Figure 8.15: Isolines of the Mach number for mesh M5 using first (top left), second (top right), third (lower left) and fourth (lower right) degree for the viscous flow, $M = 0.5$, $Re = 500$, $\alpha = 2°$

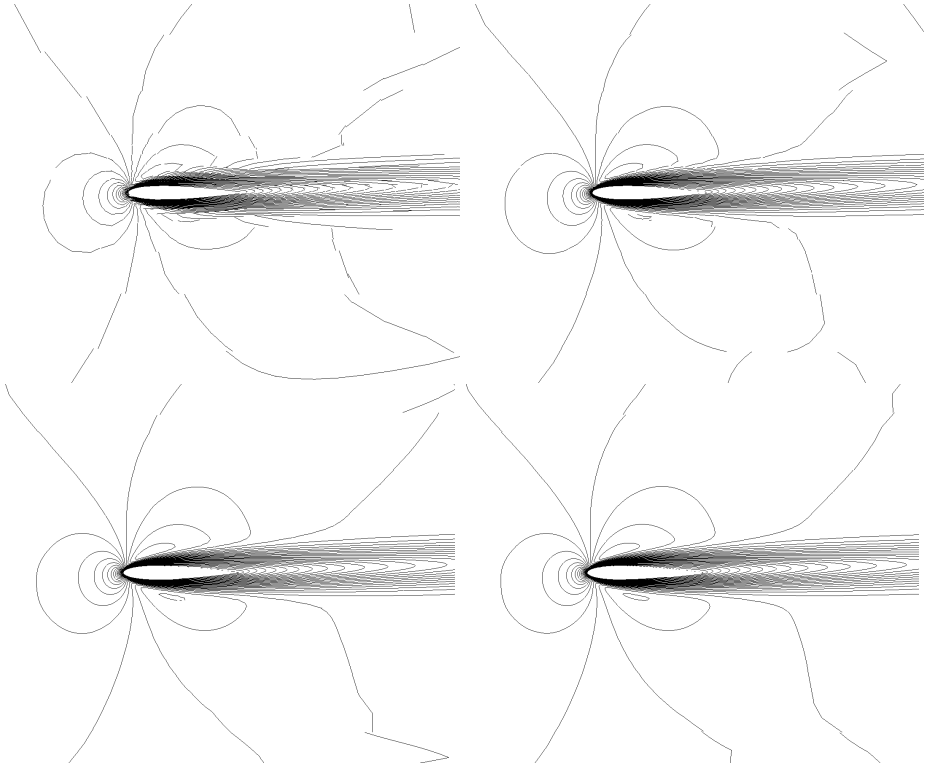

Figure 8.16: Isolines of the Mach number for mesh M6 using first (top left), second (top right), third (lower left) and fourth (lower right) degree for the viscous flow, $M = 0.5$, $Re = 500$, $\alpha = 2°$

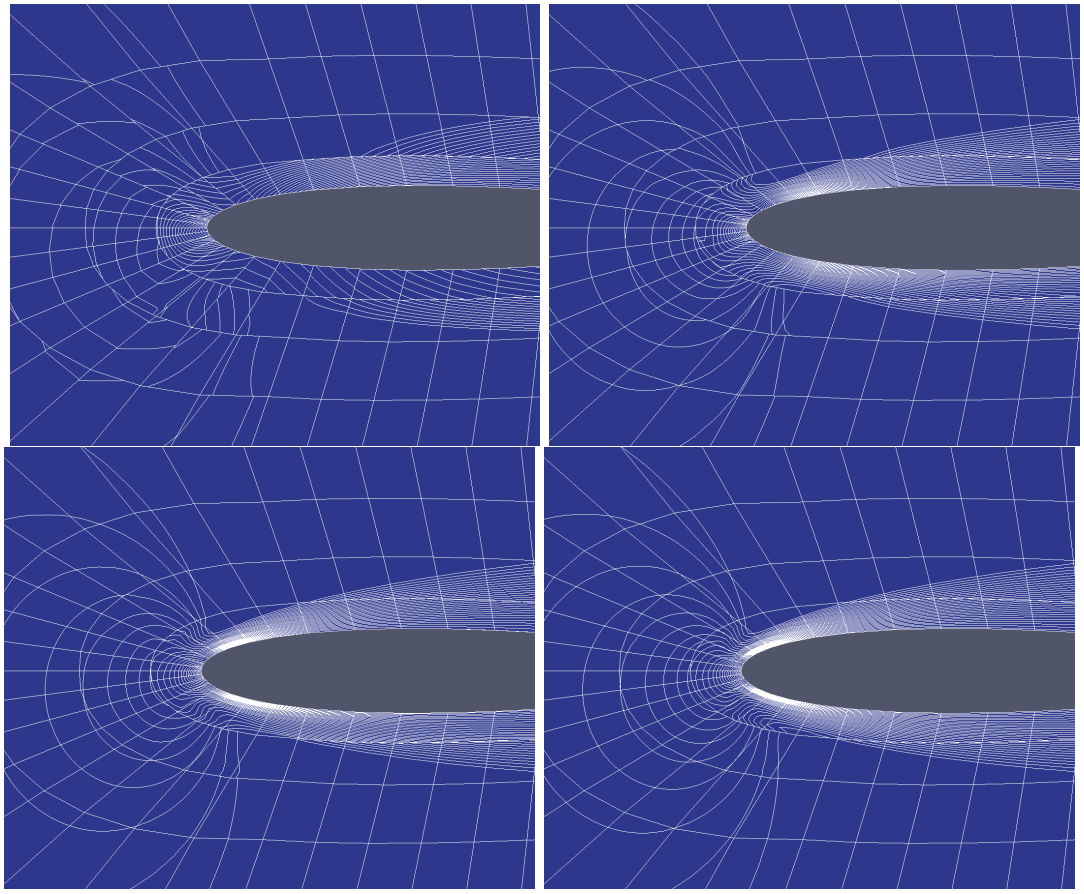

Figure 8.17: Simulations of the boundary layer for $M = 0.5$, $Re = 500$, $\alpha = 2°$ for first (top left), second (top right), third (lower left) and fourth (lower right) polynomial order expansion

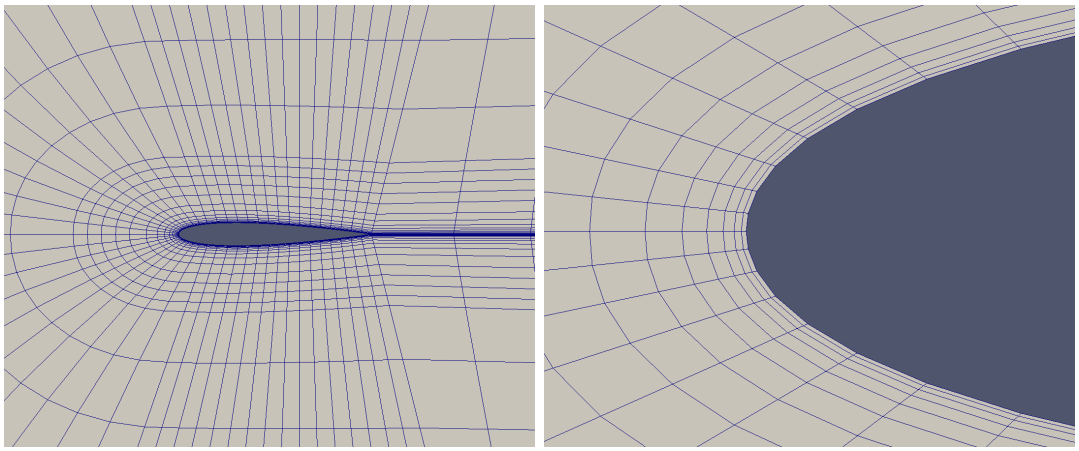

Figure 8.18: The mesh for the viscous, $M = 0.5$, $Re = 2000$, $\alpha = 2°$

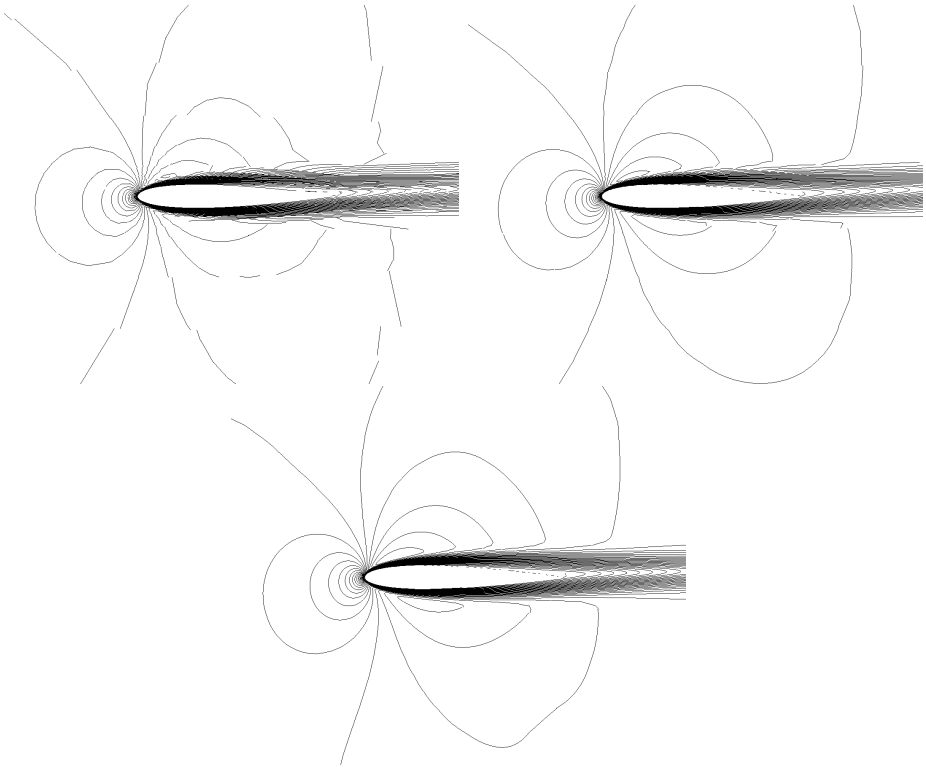

Figure 8.19: Isolines of the Mach number for $M = 0.5$ and $Re = 2000$, $\alpha = 2^\circ$ for first (top left), second (top right) and third (lower) polynomial degree

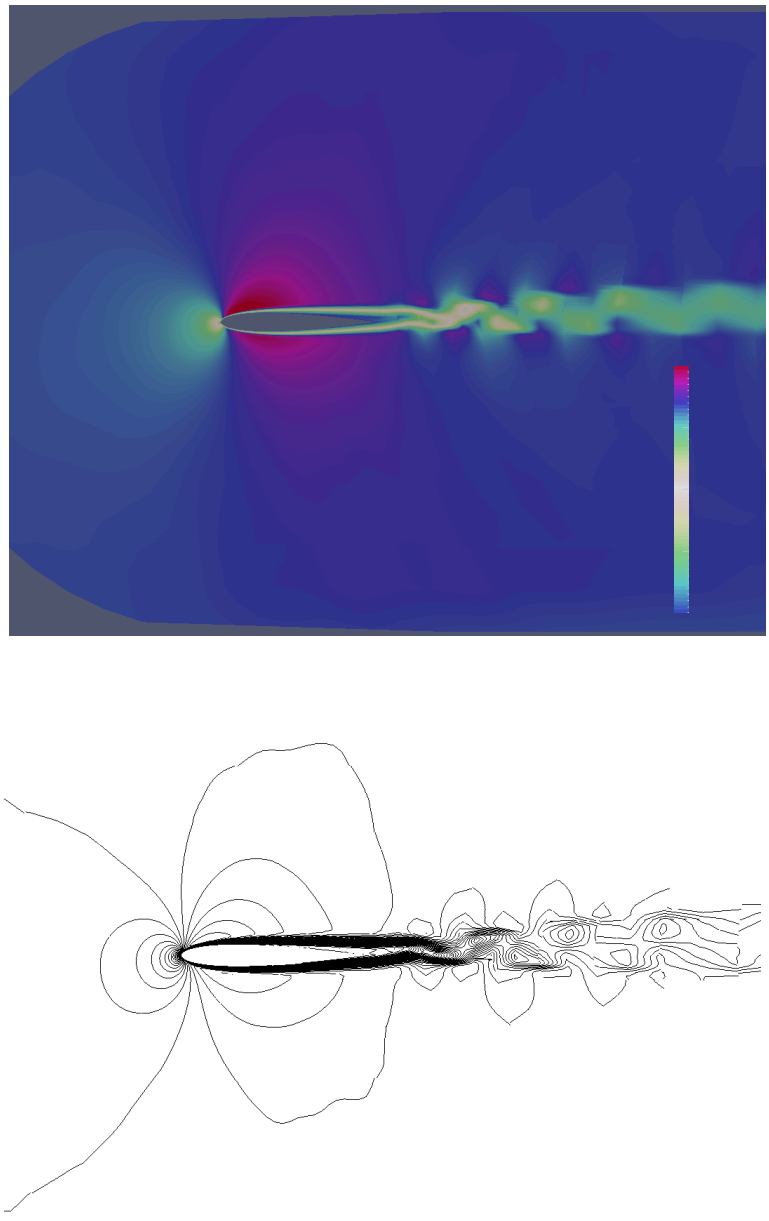

Figure 8.20: Mach number (top) and isolines of the Mach number (lower) for $M = 0.5$, $Re = 5000$ and $\alpha = 2°$

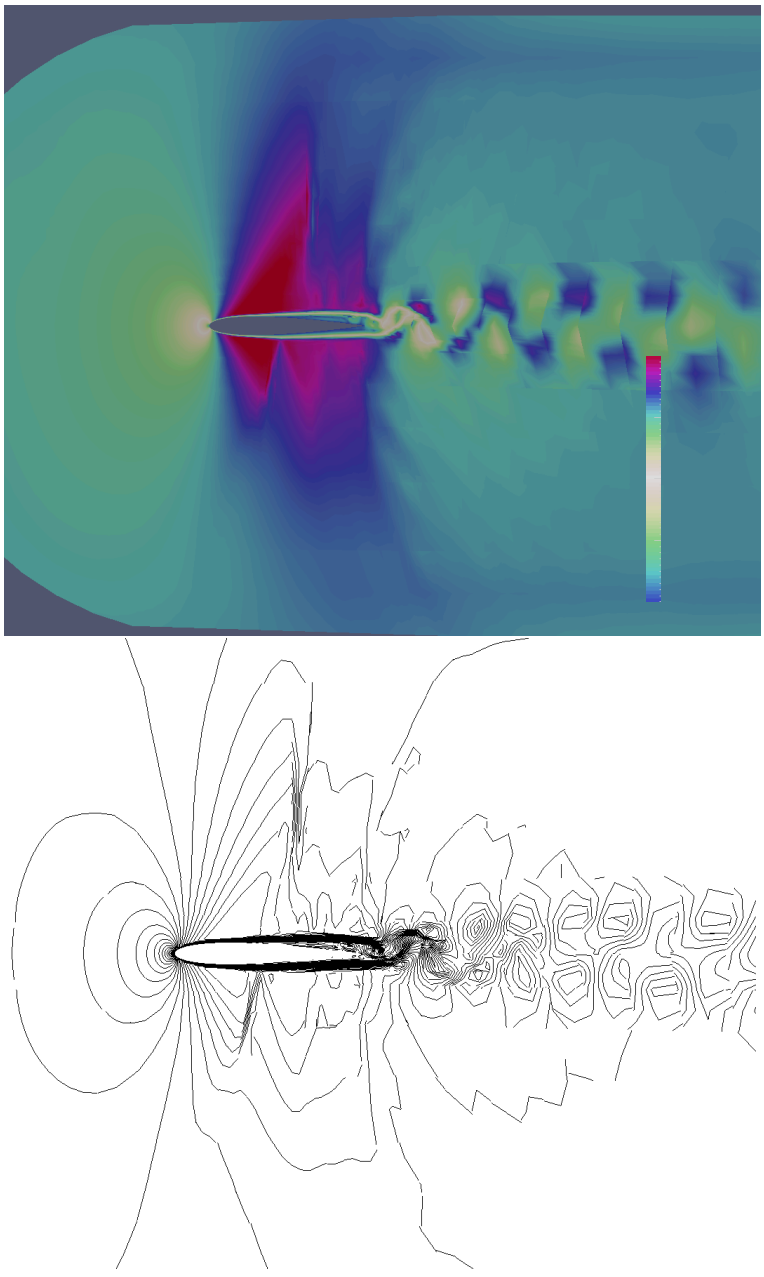

Figure 8.21: Mach number (top) and isolines of the Mach number (lower) for $M = 0.85$, $Re = 10000$

Conclusion

As we described above, high order methods provide a very powerful tool for computational fluid dynamics and are considered as the methods with a very high potential in the future, because of the computational efficiency. We mentioned, that because of the hyperbolic character of the governing equations describing the compressible flow, it is convenient to use the discontinuous projection. For this purpose, we used the discontinuous Galerkin method, which is suitable as the high order method.

In the work we used open-source framework Nektar++ which provides the spectral element discretization using the discontinuous Galerkin method. The goals of the thesis were the following: study and describe the discretization schemes implemented in Nektar++ and test the abilities of the framework for the various regimes of the compressible flow.

At first, we summarized the Euler and the Navier-Stokes equations, see Chapter 2. Then, the discretization of the discontinuous Galerkin method for the compressible Euler and the Navier-Stokes equations was summarized. We note, that the official documentation for the software does not provide a sufficient insight into the discretization of the equations and the description was not a trivial part of the work. In order to find out all the important parts of the implementation we had to examine the source code of Nektar++.

We introduced finite dimensional spaces S_p and $\mathbb{S}_p$, and discussed the necessity of defining the advection numerical flux. In Section 3.4, we presented one exact and three approximated Riemann solvers. Then, we discussed the discretization of the Navier-Stokes equations and the local discontinuous Galerkin method, with diffusion numerical fluxes, see Section 3.5. In Section 3.7, we presented the implementation of boundary conditions using the weak boundary condition approach. In Section 3.8 we defined the approximative solutions. Finally we discussed the shock capturing method, see Section 3.9.

In Chapter 4, the expansion basis was discussed. Then, we introduced the integration and differentiation operations on the reference and general elements. In Chapter 5, we formulated the matrix form of the discretization. The time integration methods were discussed in Chapter 6 using Runge-Kutta methods.

Before carrying out the numerical experiments, the high order meshes needed to be created and converted into format suitable for Nektar++. The process is described in Section 8.1, we implemented a part of this process in framework Nektar++ using C++ programing language. We note that various simulations required various meshes in order to remain precision and the computational efficiency.

The first test case was the inviscid, subsonic flow, see Section 8.2. We computed the solution for various meshes and polynomial orders up to the sixth order. The computational time was too high for higher polynomial orders. We implemented the computation of the lift and drag coefficients into Nektar++ and observed the results, see Table 8.1 and Graph 8.4. From Figure 8.5 we can observe that it is more advantageous to use a higher polynomial expansion than a finer mesh to obtain a smoother solution. We implemented the output of the expansion coefficients from Nektar++ and displayed them in Figure 8.6.

Further, we studied the inviscid, transonic flow and observed the typical shock wave in the solution, see Figure 8.9. In Graph 8.10, we demonstrated the usage of the shock capturing method by suppressing the oscillations at the area of the shock and obtained a smoother solution. For this purpose, we implemented the computation of the pressure coefficient into Nektar++.

In the next test cases, we studied viscous flows. At first, the subsonic flow was investigated. We computed the lift and drag coefficients for $Re = 500$ up to the fourth polynomial order expansion, for results see Table 8.2, Figure 8.13 and Figure 8.14. Further, we studied the creation of the boundary layer. We observed that for the first polynomial degree expansion the boundary layer was the size of the first boundary element. We obtained thinner boundary layer, even thinner than the first element for higher polynomial orders. This is not possible to obtain using the finite volume scheme. Finally we studied the unsteady flow, see Section 8.5.

In the work, we tested various compressible flows and confirmed the potential of the high order methods. We demonstrated the computational efficiency. On the other hand, high order methods require a good mesh with curved boundaries. Otherwise, the solution diverged at the top or at the bottom of the profile. In the work we unsuccessfully simulated the viscous subsonic flow past turbine cascade SE 1050 using triangular meshes. The solution diverged near the trailing edge of the turbines. However, the simulation were mostly successful for NACA0012 airfoil.

Bibliography

Nektar++: Spectral/hp element framework version 4.4.1 user guide, 2017. `http://www.nektar.info/downloads/file/user-guide-pdf-3/`.

C.D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R.M. Kirby, and S.J. Sherwin. Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications*, 192:205 – 219, 2015.

Bernardo Cockburn and Chi-Wang Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

Vít Dolejší and Miroslav Feistauer. *Discontinuous Galerkin Method*. Springer, 2015.

M. Feistauer, J. Felcman, and Z. Vlášek. *E2D2 software*, 1992.

Miroslav Feistauer, Jiri Felcman, and Ivan Straškraba. *Mathematical and Computational Methods for Compressible Flow*. Oxford University Press, 2003.

George Karniadakis and Spencer Sherwin. *Spectral/HP Element Methods for Computational Fluid Dynamics*. Oxford University Press, 2005.

Gianmarco Mengaldo, Daniele De Grazia, Joaquim Peiro, Antony Farrington, Freddie Witherden, Peter Vincent, and Spencer Sherwin. A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics. *AIAA AVIATION 2014 -7th AIAA Theoretical Fluid Mechanics Conference*, 2014.

Kroll Norbert, Bieler Heribert, Deconinck Herman, Couaillier Vincent, van der Ven Harmen, and Sørensen Kaare. *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*. Springer, 2010.

Per-Olof Persson and J Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *AIAA paper*, 2, 2006.

Wm. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. (LA-UR–73-479), 1973.

Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 2009. ISBN 978-3-540-25202-3.

List of Figures

List of Tables

A. Attachments

A.1 Jacobi polynomials

Jacobi polynomials are denoted by $P_p^{\alpha,\beta}(x)$ and represent a class of orthogonal polynomials. These polynomials are defined as a solution to Sturm-Liouville problem defined as:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[(1-x)^{1+\alpha}(1+x)^{(}1+\beta)\frac{\mathrm{d}}{\mathrm{d}x}u_p(x)\right] = \lambda_p(1-x)^{\alpha}(1+x)^{\beta}u_p(x), \tag{A.1}$$

for $x \in [-1,1]$, where

$$\lambda_p = -p(\alpha+\beta+p+1), \tag{A.2}$$

and $u_p(x)$ represents the Jacobi polynomial: $u_p(x) = P_p^{\alpha,\beta}(x)$.

As we mentioned, an important property of Jacobi polynomial is the orthogonality given by the following condition:

$$\int_{-1}^{1}(1-x)^{\alpha}(1+x)^{\beta}P_p^{\alpha,\beta}(x)P_q^{\alpha,\beta}(x)\mathrm{d}x = C\delta_{pq}, \tag{A.3}$$

where

$$C = \frac{2^{\alpha+\beta+1}}{2p+\alpha+\beta+1}\frac{\Gamma(p+\alpha+1)\Gamma(p+\beta+1)}{p!\,\Gamma(p+\alpha+\beta+1)}. \tag{A.4}$$