Airfilm cooling through laser drilled holes

Airfilm Cooling through Laser Drilled Holes

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Sizov, Mikhail A.

Airfilm Cooling through
Laser Drilled Holes /
by Mikhail A. Sizov. -
Eindhoven: Eindhoven University of Technology, 2007. Proefschrift. -
ISBN 978-90-386-1026-9

NUGI 811
Subject headings: boundary value problems; numerical methods /
high order compact schemes; direct numerical simulation /
turbulence
2000 Mathematics Subject Classification: 65N50, 65N55, 65N06, 80A25

Airfilm Cooling through Laser Drilled Holes

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College
voor Promoties in het openbaar te verdedigen
op woensdag 4 juli 2007 om 16.00 uur

door

Mikhail Anatolievits Sizov

geboren te Leningrad, Rusland

Dit proefschrift is goedgekeurd door de promotor:

prof.dr. R.M.M. Mattheij

Copromotor:
dr.ir. M.J.H. Anthonissen

This research was financially supported by the Dutch Technology Foundation STW (project EWO:5478)

Contents

Chapter 1

Introduction

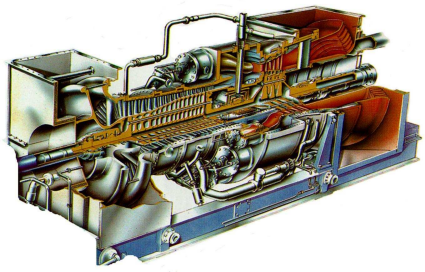
1.1 Background

Two primary applications of combustion turbines are aero propulsion and power generation. Turbofan engines typically use a dual spool configuration consisting of a high-pressure turbine coupled with a high-pressure compressor and a low-pressure turbine coupled with a fan (see Figure 1.1). The bypass ratio, or the ratio of fluid bypassing the core to fluid passing through the core, is varied to accommodate efficiency and performance needs. Thrust is developed by the momentum of the fluid exiting the engine. Commercial airliners typically use high bypass ratios, generating thrust by moving large quantities of air, because this is more efficient from the fuel economy point of view. Military engines put a premium on performance and often use low bypass ratios and high jet velocities sacrificing some efficiency.

The second wide spread application of combustion turbine technology is power generation. Power turbines are much larger than aero turbines because weight and space are not an issue. They are frequently as large as a school bus and place a premium on efficiency. Units range in size from output measured in kW to 500 MW. A GE MS7001 power turbine is shown in Figure 1.1. Combustion turbines are ideal for peaking power (meeting maximum power needs) because they can be started and provide power to the grid in minutes. In contrast, a nuclear power plant, which is used for base load power, must go through multiple modes and testing at startup and may take up to three days to reach full power. Combustion turbines have also found wide spread use in the power industry as merchant plants. Merchant plants are stationed in areas of high power demand, such as California, and when the local utility has a shortage of power or a plant goes off-line, the owner of the merchant plant can fire the plant and sell the electricity.

The basic components of the *combustion turbine* are labeled in Figure 1.2. Air enters the machine from the atmosphere through the intake and absorbs work from the com-

(a) Rolls-Royce Trent aero engine (Courtesy of Rolls-Royce)



(b) General Electric MS7001 power turbine (Courtesy of General Electric)

Figure 1.1: Examples of gas turbines

pressor which raises both the pressure and temperature of the air. The air then passes through the combustor where heat energy is added in the form of fuel. The combustion gases are then expanded through the turbine, extracting useful work, and finally exhausted to the atmosphere out of the nozzle. The turbine is coupled with the compressor and this unit together with the combustor is known as the gas generator. The turbine extracts only enough work to continue driving the compressor. In turbojet and turbofan applications, the relatively high enthalpy fluid leaving the turbine is then accelerated through the exit nozzle creating a high momentum jet which delivers thrust. In other applications where thrust is not desirable, such as power plants or helicopter engines, the fluid is then further expanded through a second turbine known as the power turbine. The power turbine is coupled to a generator, propeller, or some other shaft driven device.

Different methods have been used in the development of turbine blades (which rotate) and vanes (which are static) in order to deal with increasing turbine inlet temperature. One of the major problems in enhancing the efficiency in gas turbines is the maximum possible value of the turbine inlet temperature allowed in the view of blade material properties (see Figure 1.3). Up to about 1300 K uncooled blades can be used.

The first method is concerned with the material airfoils are made of and how they are casted. A first aspect is that airfoils material has resulted in better mechanical and heat resistance properties. Better casting techniques made the blade stronger with respect to both mechanical and heat resistance. This led from the conventionally casted turbine blade with good mechanical properties in all directions and homogenous crystal structure in all directions via the directionally solidified turbine blade with improved mechanical properties in the longitudinal direction and a columnar crystal structure to single crystal turbine blade with excellent mechanical properties in longitudinal axis and improved heat resistance.

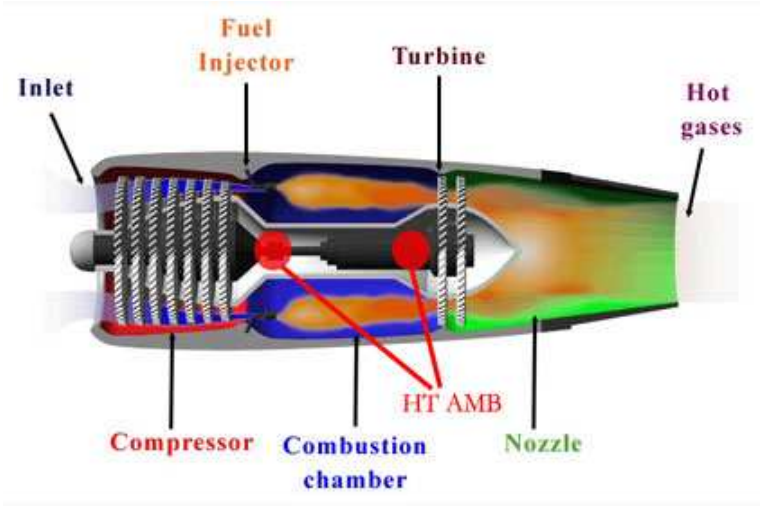


Figure 1.2: Scheme of an aero-engine (courtesy of ASL/ETHZ).

A second technique to be able to increase the thermal load on the turbine blades and vanes is to cover the airfoil with a coating which creates a sort of thermal barrier. The third method is to cool the blades. This cooling is done both internally and through *film cooling*. In both techniques (relatively) cold air is used to cool the blades. In the past the blades were cooled only internally through either drilled longitudinal holes or casted cavities. With the increasing turbine inlet temperature this was not enough, so in the seventies film cooling was introduced. In film cooling compressed air is injected along the blade surface, forming a cold boundary layer, thus separating the hot gas from the blades. Nowadays film cooling is used together with internal cooling. The cooling nozzles are usually produced by some kind of drilling. There are several methods to do this, but most of them have the drawback of low drilling speed. Mechanical drilling is not suited for superalloys, although it is fast, but it is limited to holes with a diameter larger than 3 mm. Electro-Chemical Drilling (ECD) is done by electrolysis. This is a process in which an electric potential difference is imposed on an anode and a cathode. The electrolyte, usually a sulphuric acid, tends to corrode the anode surface in the electric field. ECD is able to produce very neat holes, since the scale on which the drilling has to be performed is very small, but it is slow and produces a lot of waste. The ECD was modelled in [51].

Electric Discharge Machining (EDM) is a method for producing holes and slots, or other shapes, by using an electric discharge (spark) to remove unwanted material. It is also called spark erosion. Sometimes it is used to produce a part, such as producing a slot in a very hard metal, and sometimes it is used to "rescue" a part such as removing a broken tap. The basic idea is to move an electrode very close to the work piece, and repeatedly produce a spark between the two. This is best done while immersed in a dielectric liquid rather than in air, and it helps if the proper distance can be automatically maintained. Note that the electrode gets eaten as well as the workpiece, and some com-

Figure 1.3: Example of an overheated blade

pensation must be made for this. Very good finish can be achieved, though at reduced speed. EDM is not a fast method; some jobs can take days to produce holes, so its use is limited to jobs that cannot easily be done in other ways (e.g. oblong slots or complex shapes, sometimes in very hard material). Also EDM cannot be used for coated materials. ECD and EDM have typically drilling speeds of 1-10 mm/min, but several holes can be drilled at the same time, using multiple electrodes.

Yet another technique is Electron Beam Drilling (EBD). An electron beam machine works in much the same way as a cathode ray tube in a television. The electron beam machining process is fairly straightforward. First a stream of electrons is started by a voltage differential. The concave shape of the cathode grid concentrates the stream through the anode, much like the way a concave mirror focuses a light beam from a flashlight. The anode applies a potential field that accelerates the electrons. This stream of electrons is then forced through a valve in the electron beam machine. The valve is used for controlling the beam and the duration of a machining process. If the impulse of the electron beam is too large, the part will overheat and potentially ruin the machined piece by either distorting a feature or relaxing strength built up from material cold-working or tempering. Once passing through the valve, the beam is then focused onto the surface of the work material by a series of electromagnetic lens and deflector coils. The entire process occurs in a vacuum chamber. The reason for the need of a vacuum for the electron-beam machining process is that air molecules can adversely interact with the beam of electrons. A collision between an electron and an air molecule causes the

electron to veer off course.

In contrast to the electron-beam machining, a laser machining process can be done outside of a vacuum, in ambient atmospheric conditions because the size and mass of a photon is numerous times smaller than the size of an electron. Laser drilling offers an alternative to mechanical drilling, punching, broaching and wire EDM. It is especially adaptable for small holes with large depth-to-diameter ratios. With laser drilling, a wide range of hole diameters are obtainable. Material such as steel, nickel alloys, aluminum, copper, brass, borosilicate glass, quartz, ceramic, plastic and rubber are all being successfully laser drilled. The laser is so fast and so repeatable that it is ideal for high production volumes associated with fully automated or semi-automated tooling applications. Laser drilling is the process of repeatedly pulsing focused laser energy at a material, vaporizing layer by layer until a thru-hole is created. This is what is called a popped or percussion drilled hole. Depending upon material and material thickness, a popped hole could be as small as 0.04 mm in diameter. If a larger hole is required, the laser, once through the material, is moved with respect to the work piece to contour the desired diameter. This is called trepanning. The end result is a fast, efficient way to create holes.

To increase the turbine inlet temperature several parts have to be cooled and, as mentioned above, one way to do this is to drill cooling nozzles in the components. There are several groups of components where drilling is needed - the blades, the vanes and the combustion chambers. For the blades the typical number of holes to be drilled is around 300 film cooling holes per blade. These holes are cylindrical or fan shaped with the diameter ranges from 0.5 to 1.0 mm and depths varying between 3 and 10 mm. For the cylindrical holes laser drilling is often used, although for the fan shaped holes EDM is better suited. For the vanes, approximately 500 holes per part need to be drilled, both cylindrical and fan shaped. Again, for the cylindrical holes laser is used, while EDM is used for the fan shaped ones. In the inserts on average of 300 holes need to be drilled. The holes are cylindrical, 0.3-3 mm in diameter. For such type of the job laser drilling gives the best results. For the drilling of the holes in the combustion chambers more than 100,000 holes need to be drilled and this is the area where high drilling speed is essential and where laser drilling is the method to use. As one can see, laser drilling can be used extensively for drilling different holes in gas turbine components.

There are roughly three techniques to drill with a laser. The simplest way is to remove material through a single laser pulse. This technique is mainly used for drilling narrow ($<$ 1 mm) holes through thin ($<$ 1 mm) plates. Another method, used to drill wider ($<$ 3 mm) holes in plates ($<$ 10 mm), is to cut a contour out of the plate. This technique is called laser trepanning drilling. The drilling process in which the laser operates in a repeated manner, with short pulses, ranging from 10^{-12} to 10^{-3} s, which are separated by longer time periods, is called laser percussion drilling. In this way the laser builds up energy and operation in this manner allows for large bursts of energy.

Laser percussion drilling is favored over the older drilling techniques and the other laser drilling techniques because it is by far the quickest. However, it still suffers from some drawbacks. The first drawback is that a so-called recast layer, that is, resolidified

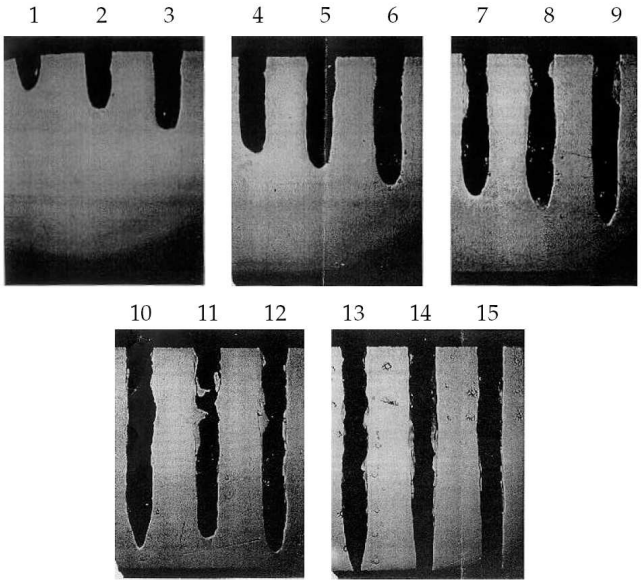


Figure 1.4: A series of photos of holes produced by laser percussion drilling. The number of pulses to produce each hole is indicated. (Courtesy of ELDIM BV)

material remains at the wall of the hole. Some resolidified material can normally also be found at the entrance and exit of the hole, in which cases it is called spatter and dross, respectively. Furthermore, the holes normally show some tapering: the decrease of hole diameter with depth therefor control of the taper angle and reproducibility is needed. Finally, occasionally the hole resulting from a laser percussion drilling process shows barrelling or a bellow shape: the local increase of hole diameter. The process of laser percussion drilling was studied in detail in [80].

1.2 Overview of some previous numerical investigations

Three-dimensional calculations of film cooling of real turbine blade models are less abundant than calculations of jets in crossflow over flat plates, e.g. [7, 35, 41, 83]. There exist even fewer calculations of surface heat transfer of fully film-cooled rotating turbine blades [22]. Various other investigations exploring the effect of several parameters on the flow and temperature fields over different blade prototypes revealed the importance of extending the calculations into the cooling channel, at least for low blowing rates. Turbulence models that have so far been employed in film cooling calculations do not go beyond the two-equation type. An exception is the work reported in [18], where the predictive performance of various two-equation turbulence models were compared to the sophisticated Reynolds stress model RSM. Surprisingly, RSM calculations were

not better than those with eddy viscosity models. Authors of [23] reported on calculations for the C3X vane and the VKI rotor where they used the Baldwin-Lomax model and various two-equation turbulence models. No clear picture emerged from the calculations indicating which of the employed models is superior. The same Baldwin-Lomax model was also employed in [11] in their computations of the film cooling of the two-dimensional AGTB cascade. For this particular configuration, which has been widely adopted as a benchmark for code validation, [31] and [73] used the standard $k-\epsilon$ model with wall functions.

In [72], the authors report on the simulation of the flow around the three-dimensional version of the AGTB test-case. Owing to the difficulties in calculating these flows using a full three-dimensional RSM under low Reynolds number conditions authors of [72] and [36] opted for another strategy. This consists in retaining the basic architecture of the $k-\epsilon$ model, while an anisotropic eddy-viscosity/diffusivity correction for secondary stresses is introduced. The extension of the model to low-Re number conditions was in all cases achieved via a dynamic, zonal, two-layer approach [60]. This practice was first applied to jet injection over a flat plate [35], then to real blade models [36, 72].

1.3 Direct numerical simulation of turbulent flows

"Turbulence has been the victim of many colorful descriptions over the years from Lamb's (1916) scholarly *chief outstanding difficulty of our subject* to Bradshaw's (1994) inspired *invention of the Devil on the seventh day of creation.* This apparent frustration results largely from the mixture of chaos and order and the wide range of length and time scales that turbulent flows possess" [48]. The complex behaviour of turbulence is the consequence of a set of equations, the Navier-Stokes equations. However, analytical solutions to even the simplest turbulent flow problems do not exist. A complete description of a turbulent flow, where the flow variables (e.g., velocity and pressure) are known as a function of space and time can therefore only be obtained by numerically solving the Navier-Stokes equations. The range of scales in turbulent flows increases rapidly with the Reynolds number (as $Re^{\frac{9}{4}}$). As a result, most engineering problems, e.g., the flow around a car, have too wide a range of scales to be computed using Direct Numerical Simulation (DNS). The engineering computation of turbulent flows therefore relies on simpler descriptions: instead of solving for the instantaneous flowfield, the statistical evolution of the flow is sought. Approaches based on the Reynolds-averaged Navier-Stokes (RANS) equations are the most prevalent (see [67] for a review) and involve computing one-point moments such as mean velocity and turbulent kinetic energy. Another approximation, large eddy simulation (LES), is intermediate in complexity between DNS and RANS. Large eddy simulation directly computes the large energy-containing scales, while modelling the influence of the small scales (see [40, 47] for a review).

Statistical descriptions suffer from the problem of closure, i.e. the equations describing the statistical evolution of the flow contain terms that cannot be obtained from the

Navier-Stokes equations and therefore require modelling. The search for better turbulence models, and better parameterization of the turbulence, is what drives most turbulence research.

In contrast to its incompressible counterpart, DNS of compressible turbulent flows is fairly recent. The early 1980s saw DNS of homogeneous compressible turbulence being initiated. However, it was not until a decade later that serious study of compressible homogeneous turbulence (isotropic and sheared) was undertaken [10, 17, 37, 65]. Wall-bounded flows such as the compressible channel [14] and a turbulent boundary layer [57] have only recently been attempted. High-speed turbulent mixing layers have been the focus of much experimental attention; DNS of this flow was performed by Vreman et al [82]. An exciting new development has been the field of computational aeroacoustics, where both the fluid motion and the sound it radiates are directly computed (see [39,70] for reviews).

1.4 Requirements on the solution strategy for the film cooling problem

Often the variations of the solution are large only in a part of the domain and small elsewhere. The film cooling problem sketched in Figure 1.5 is a typical example of such a problem. It is (still) beyond computer facilities to use a uniform grid over the whole domain with a resolution that captures the local phenomena because of the large number of points required. To approximate the solution, a large system of algebraic equations resulting from discretization has to be solved and information has to be stored at a large number of grid points. In general the advantages of using a uniform grid, like simple data structures and the existence of simple, accurate discretization stencils and fast solution techniques for the resulting system of algebraic equations do not counterbalance the disadvantage of having so many redundant points, and the use of a global uniform grid is computationally inefficient. Local grid refinement allows us to keep the advantages of the uniform grid, while having high resolution where necessary and avoiding redundant points. Let us consider a typical example of the film cooling problem (see Figure 1.5). From the experimental data it is known [32] that the crucial part of the domain with respect to the heat loads is the area just downstream the cooling nozzle, so we would like to have highest resolution in this area. The idea of local grid refinement is quite simple - we have one coarse grid which covers the whole domain and one (or more) fine grids in the area we are interested in (for the film cooling this is the area downstream the cooling nozzle). In the present work we use two local grid refinement techniques, namely Local Uniform Grid Refinement (LUGR) and Local Defect Correction (LDC) which are studied in more detail in Chapters 3, 4, 5.

In film cooling, an important and not yet fully investigated question is the profile (temperature, velocity, etc.) of the cooling jet at the exit of the cooling nozzle, where it interacts with the main flow. Most attempts to clarify the situation include the use of unstructured meshes to model underlying geometry and are thus not as precise as DNS-

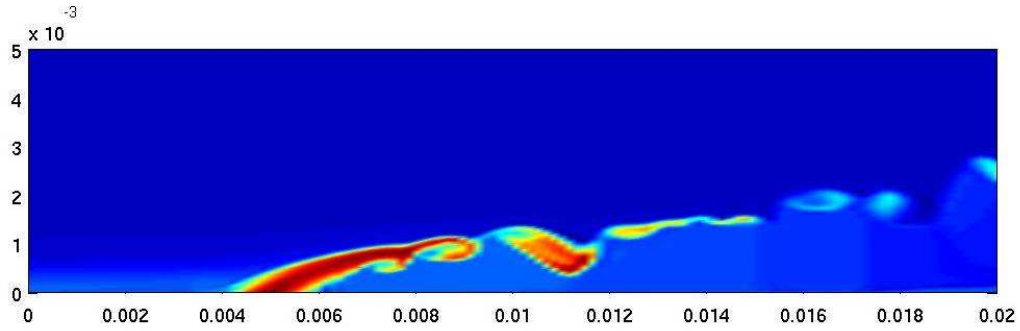


Figure 1.5: Typical example of the film cooling problem. Temperature distribution for the inverse problem (cold main flow, hot "cooling" jet) on the center plane. Film cooling nozzle is located between 0.004 and 0.006 mm.

type simulations. In the present work we propose another approach, based on domain decomposition (see Figure 1.6). We discuss it in more details in Chapter 7.

1.5 Outline of this thesis

In Chapter 2 of this thesis, we describe the physical picture, which takes place during air film cooling as well as the mathematical model of such a process. First we introduce the system of equations for flow (namely Navier Stokes equations for compressible gas), then we describe briefly the appropriate boundary conditions.

In Chapter 3, we concentrate on discussing issues connected with Local Grid Refinement. We discuss two techniques - Local Uniform Grid Refinement and Local Defect Correction (LDC). In these techniques the discretization on the *composite grid* is based on a combination of standard discretizations on several uniform grids with different grid sizes that cover different parts of the domain. At least one grid, the coarse grid, should cover the entire domain, and its size should be chosen in agreement with the relatively smooth behavior of the solution outside the high activity areas. Apart from this *global coarse grid*, one or several *local fine grids* are used in the high activity areas. These grids are also uniform. Each of the local grids covers only a (small) part of the domain and contains a high activity region. The LDC method is an iterative process: a basic global discretization is improved by local discretizations.

Chapter 4 is devoted to the convergence analysis of the Local Defect Correction technique. First we present a theoretical analysis of the convergence behavior for simple convection-diffusion equations and different discretization schemes, that is upwind and central differences. We conclude Chapter 4 with a comparison of the theoretical conver-

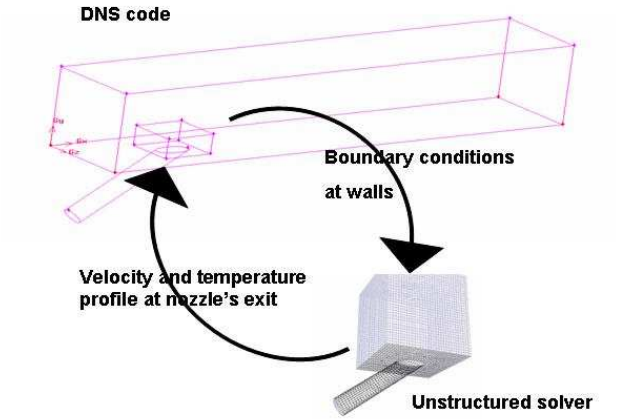


Figure 1.6: Domain decomposition strategy.

gence rate with one we see during actual numerical calculation.

In Chapter 5 we discuss the combination of the Local Defect Correction technique with high order discretization schemes. High order discretization schemes are widely used for DNS calculations. We start with a short introduction to the high order (compact) schemes. Next we propose a method to combine those schemes with the LDC method, introduced earlier in Chapter 4. We conclude the chapter with numerical examples which illustrate properties of the proposed method, such as accuracy and efficiency.

In Chapter 6 we discuss the issue of the boundary conditions for compressible flows. We start with describing the existing approaches for boundary conditions in case of turbulent flows. Next, we formulate so-called non-reflecting boundary conditions based on work of Lele and Poinsot [55]. For our grid refinement problem we have to introduce artificial boundary conditions for the fine grid. Normally this is done with the use of interpolation from the coarse grid. In case of compressible flows use of interpolation could lead to reflections from the artificial boundaries. In order to investigate this problem, in the last section of Chapter 6 we present a numerical example, which shows absence of the reflections for the problem of interest.

Chapter 7 of this thesis is devoted to numerical simulations of the air film cooling problem, based on the methods described in previous chapters. First we present a full mathematical formulation of the problem of interest. Next in Section 2 we discuss numerical techniques to solve the problem of interest. Section 3 is devoted to the discussion of domain decomposition and parallelization for the three-dimensional flow of compressible fluid. We conclude this chapter with a short discussion of the numerical results

obtained, such as inflow profiles for the cooling jet, accuracy and efficiency of the local uniform grid refinement.

We finish this thesis with conclusions and future recommendations in Chapter 8.

Chapter 2

Problem description and mathematical model

2.1 Air film cooling

Film cooling gives rise to very complex flow and heat transfer processes. Its efficiency can be influenced by a number of parameters: the blade geometry and curvature, the shape of the injection hole, the injection angle (which can be perpendicular, streamwise inclined or spanwise inclined), the blowing rate (also known as the mass-flux ratio), the density and temperature ratios, the freestream turbulence and compressibility effects. The flow in the vicinity of the discharge holes depending of flow conditions can be particularly complex due to the interaction of the coolant jet with the flow around the blade. The individual jets are bent over by the mainstream, leading to the formation of longitudinal vortices and a reverse-flow region below the jet. The flow becomes even more complicated when the injection is lateral, which is often the case in practice, since then the cooling film covers the area to be cooled better. The formation and location of the longitudinal vortices depend strongly on inclination and blowing rate. More precisely, the strength and elevation of these vortices depend on the penetration of the injected coolant into the cross flow. In the case of streamwise injection two counter-rotating vortices form, while in the lateral injection configuration there is only one large-scale vortex. The vortices entrain ambient hot gas and move it to the vicinity of the wall and hence adversely influence the cooling effectiveness. This phenomenon is more pronounced at higher blowing rates for which the jets penetrate more deeply into the oncoming flow and the vortices are lifted further from the surface.

The foregoing description is in fact an idealization of the flow as if it were perfectly stable and steady, and that is exactly what turbulence models of all types are supposed to predict. The reality of jets in cross flow is more complex than that: the flow features

a broad band of vortical, large-scale structures, dominated by the well-known kidney vortex, but also includes other unsteady structures such as the horse-shoe vortex, the wake vortices, and the shear layer vortices; these can be reproduced only by means of nontime-averaged-based concepts such as direct and large eddy simulation.

It has been well established (see [32] and references therein) that the geometry of the nozzles is an important parameter in film cooling. The cooling is effectively influenced by local flow phenomena like separation of the flow in the cooling hole, relaminarization and reattachment. In film cooling cold air is injected into the boundary layer through small nozzles in the blade surface. The flow through these nozzles is laminar with a Reynolds number of typically 100 – 1000 (based on the diameter of the cooling jet). The speed in the nozzles is of the same order of magnitude as the free stream velocity. Interaction of the jets with the (laminar) boundary layer flow (with a Reynolds number typically 1000 based on the domain length and free stream velocity) is essentially three-dimensional. The collision of the laminar jet with the boundary layer flow produces a local turbulent shear layer and changes the local heat transfer to the blade (when poorly constructed it may even increase the local heat transfer). Since, the interaction is three-dimensional, the problem at hand must be modelled using the three-dimensional flow equations. However, solving the full Navier-Stokes equations for this problem on all scales (including the smallest turbulent length scale) is (still) beyond computer facilities.

As stated the major deficiency of turbulence models is their basic assumption of local isotropic diffusive energy, which is not true in the transition phenomenon. We, therefore, propose another approach. The origin of the problem lies in the fact that solving the flow equations on all scales in the full domain is computationally expensive. However, the flow consists of a main flow with laminar boundary layer and a laminar jet, which are one by one more easy to solve. It is only the interaction of the flows which poses a problem. The high resolution is in fact only necessary in a very limited part of the domain. Hence we propose to use a so called local grid refinement method, which is able to solve the problem on simple (thus structured) grids. Only where it is necessary (say, around cooling nozzle where we have vortex formation) we then need to refine.

Moreover, instead of looking at the flow around a whole blade (which is (still) beyond present computer facilities with the approach proposed), we simplify our problem and concentrate on one hole only. The problem of interest is sketched in Figure 2.1. In this figure we schematically represent the film cooling problem. There is a main flow around the blade (marked with ”main stream”), which creates a laminar boundary layer near the blade’s surface and there is a cooling jet coming out of the nozzle (marked with ”jet” and ”nozzle” respectively).

2.2 Governing equations

The governing equations of fluid flow represent mathematical statements of balance laws of physics:

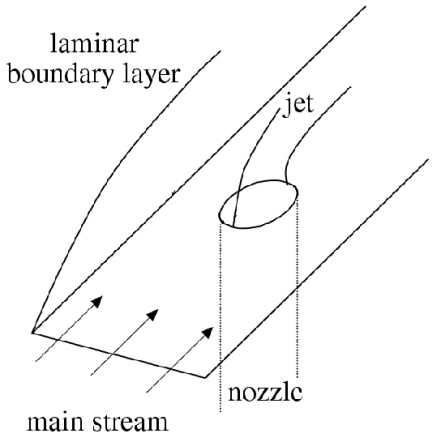


Figure 2.1: Sketch of the problem. Nozzle is located 1/4 from the beginning of the domain.

- The mass of a fluid is conserved in the absence of sources.
- The rate of change of momentum equals the sum of the forces on a fluid particle (Newton's second law).
- The rate of change of energy is equal to the sum of the rate of heat addition to and the rate of work done on a fluid particle (first law of thermodynamics).

All fluid properties are functions of space and time, so we write $\rho(\mathbf{x}, t)$, $p(\mathbf{x}, t)$, $T(\mathbf{x}, t)$ and $\mathbf{u}(\mathbf{x}, t)$ for the density, pressure, temperature and velocity vector respectively.

The following equation represents mass conservation for the unsteady flow of a compressible fluid

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

where $D\rho/Dt = \partial\rho/\partial t + \mathbf{u} \cdot \nabla\rho$ is the material derivative, ∇ is the gradient operator and $\nabla\cdot$ is the divergence operator.

Newton's second law states that the rate of change of momentum of a fluid particle equals the sum of the forces on the particle. We distinguish two types of forces which act on a fluid particle: surface forces (pressure forces, viscous forces) and body forces (gravity force, centrifugal force, Coriolis force, electromagnetic force). It is common practice to highlight the contributions due to the surface forces as separate terms in

the momentum equation and to include the effects of body forces as source terms. The state of stress of a fluid particle is defined in terms of the pressure and the viscous stress components. The pressure, a normal stress, is denoted by p. Viscous stresses are expressed by the stress tensor τ. The usual suffix notation τ_{ij} is applied to indicate the direction of the viscous stresses. The suffices i and j in τ_{ij} indicate that the stress component acts in the j-direction on a surface normal to the i-direction. So we can write the following expression for the balance of momentum:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla \cdot \tau + \rho \mathbf{f}, \tag{2.2}$$

where $\mathbf{f}$ is the external force per unit mass of fluid.

In the present work we neglect the body forces ($\mathbf{f} = \mathbf{0}$) and assume that the fluid is Newtonian. In a Newtonian fluid the viscous stresses are proportional to the rates of deformation of the fluid particle. The three-dimensional form of Newton's law of viscosity for compressible flows involves two constants of proportionality: the (first) dynamic viscosity, μ, to relate stresses to linear deformations, and the second viscosity, λ, to relate stresses to the volumetric deformation. We can write the following expressions for the components of the stress tensor τ in three dimensions in Cartesian coordinates:

$$\tau_{xx} = 2\mu \frac{\partial u}{\partial x} + \lambda \nabla \cdot \mathbf{u},\ \tau_{yy} = 2\mu \frac{\partial v}{\partial y} + \lambda \nabla \cdot \mathbf{u},\ \tau_{zz} = 2\mu \frac{\partial w}{\partial z} + \lambda \nabla \cdot \mathbf{u}, \tag{2.3}$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right),\ \tau_{xz} = \tau_{zx} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right),\ \tau_{yz} = \tau_{zy} = \mu \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right), \tag{2.4}$$

where u, v, w are the components of the velocity vector $\mathbf{u}$ in Cartesian coordinates.

Not much is known about the second viscosity, however for gases a good approximation can be obtained by taking the value $\lambda = -\frac{2}{3}\mu$. In summary, substituting expressions for the stress tensor components (2.3)-(2.4) into (2.2), we can write the following equations for the momentum balance:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}, \tag{2.5}$$

where $\nu = \mu/\rho$ is the kinematic viscosity.

The energy balance for general compressible fluid flow is described by the following equation:

$$\rho \frac{Di}{Dt} = -p \nabla \cdot \mathbf{u} - \nabla \cdot \mathbf{q} + \Phi, \tag{2.6}$$

where i is the internal energy, $\mathbf{q}$ is the heat flux vector and Φ is the viscous dissipation function. Fourier's law of heat conduction relates the heat flux to the local temperature gradient:

$$\mathbf{q} = -k \nabla T, \tag{2.7}$$

where k is the heat conduction coefficient. Substitution of (2.7) into (2.6) gives

$$\rho \frac{Di}{Dt} = -p \nabla \cdot \mathbf{u} + \nabla \cdot (k \nabla T) + \Phi. \tag{2.8}$$

All the effects due to viscous stresses in the internal energy equation (2.8) are described by the dissipation function Φ, which can be shown (see [28]) to be equal to:

$$\Phi = \mu \left\{ 2 \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + \left(\frac{\partial w}{\partial z} \right)^2 \right] + \right.$$
$$\left. + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right)^2 + \lambda \nabla \cdot \mathbf{u} \right\}. \quad (2.9)$$

The motion of the fluid in three dimensions in Cartesian coordinate system is described by a system of five partial differential equations: mass conservation (2.1), the $x-$, $y-$, $z-$ projections of momentum equation (2.5) and the energy equation (2.8). The number of unknowns is equal to seven (pressure p; three velocity components u, v, w, temperature T, density ρ, internal energy i). Hence we need to formulate two additional equations. Among the unknowns are four thermodynamic variables: density ρ, pressure p, internal energy i and temperature T. Relationships between the thermodynamic variables can be obtained through the assumption of thermodynamic equilibrium. The fluid velocities may be large, but they are usually small enough that, even though the properties of the fluid particle change rapidly from place to place, the fluid can thermodynamically adjust itself to new conditions so quickly that the changes are effectively instantaneous. Thus the fluid always remains in thermodynamic equilibrium. The only exceptions are certain flows with strong shock waves.

We can describe the state of a substance in thermodynamic equilibrium by means of just two state variables. Equations of state relate the other variables to the two state variables. If we use ρ and T as state variables we can state equations for pressure p and specific internal energy i:

$$p = p(\rho, T), \; i = i(\rho, T). \quad (2.10)$$

For a perfect gas the following equations can be used:

$$p = \rho R T, \; i = c_v T, \quad (2.11)$$

where R is the universal gas constant and c_v is the constant volume gas heat capacity.

In the flow of compressible fluids the equations of state link the energy equation with mass conservation and momentum equation. This link arises through the possibility of density variations as a result of pressure and temperature variations in the flow field. In the case of an incompressible fluid the energy equation can be decoupled from the system and can be solved separately.

2.3 Mathematical formulation of the problem

For compressible flow of an ideal gas for which gravity forces are neglected the fluid dynamics equations, are

Figure 2.2: Sketch of the computational domain. Arrow indicates cooling jet.

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \mathbf{u} = 0, \tag{2.12}$$

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}, \tag{2.13}$$

$$\rho\frac{Di}{Dt} = -p\nabla \cdot \mathbf{u} + \nabla \cdot k\nabla T + \Phi, \tag{2.14}$$

$$p = \rho RT, \text{ and } i = c_v T, \tag{2.15}$$

These equations hold in the domain sketched in Figure 2.2. It is given in Cartesian coordinates by $\Omega = \left\{(x, y, z) \in \mathbb{R}^3 | 0 \leq x \leq X, 0 \leq y \leq Y, 0 \leq z \leq Z\right\}$, where typical values are $X = 10^{-2}$ m, $Y = 10^{-3}$ m, $Z = 10^{-3}$ m. In order to close the mathematical formulation we have to provide initial and boundary conditions. For initial conditions we assume that at time $t = 0$ we have a boundary profile along the plate with a known (Blasius) profile. A Blasius boundary layer profile is described by the solution of the following differential equation

$$ff'' + 2f''' = 0,\ f(\eta = 0) = 0,\ f'(\eta = 0) = 0, f'(\eta = \infty) = 1, \tag{2.16}$$

where $f(\eta)$ is a dimensionless stream function and $\eta = y\left(\frac{U_\infty}{\nu x}\right)$ is the similarity variable and U_∞ is the main stream velocity at infinity. The equation (2.16) does not have an analytic solution and must be solved numerically. The relation between η and corresponding velocity components are given by

$$U_b = U_\infty f',\ V_b = \frac{1}{2}\sqrt{\frac{\nu U_\infty}{x}}(\eta f' - f),\ W_b = 0, \tag{2.17}$$

Dimension	Two-dimensional		Three-dimensional	
Boundary type	Euler	Navier-Stokes	Euler	Navier-Stokes
Supersonic inflow	4	4	5	5
Subsonic inflow	3	4	4	5
Supersonic outflow	0	3	0	4
Subsonic outflow	1	3	1	4

Table 2.1: Number of physical boundary conditions required for well-posedness of two- and three-dimensional flows [69].

where U_∞ is the free stream velocity and underscore b stands for Blasius. As initial condition for temperature we assume uniform temperature T_∞ in whole computational domain.

Mathematically, the boundary conditions of a system of equations are subject to certain requirements to make the problem well posed. The number of physical boundary conditions for the well-posedness requirement for the Euler and Navier-Stokes equations has been derived by Strikwerda in [69] and is stated in Table 2.1 for two- and three-dimensional cases. These boundary conditions are provided by information about the external flow, adjacent to the boundaries. In some cases, however, no accurate external flow information is available, such as at the outflow boundary. Although mathematically only a certain number of boundary conditions is required, depending on the local flow condition, numerically we should specify all the dependent variables. We call the conditions completing the specification of the dependent variables the numerical boundary conditions. Whereas the mathematical requirement for well-posedness got general acceptance, there is still no method to specify numerical boundary conditions if the required external information is lacking. Different methods are used in literature. We discuss some in Chapter 6.

A detailed description of the boundary conditions can be found in Chapter 5, below we just list all of them. As one can see from Figure 2.2, we have six planes which are boundaries of the domain. At the side planes of the computational domain ($z = 0$ and $z = Z$) we apply periodic boundary conditions. From the mathematical point of view this means that cooling nozzles are periodically distributed infinitely in spanwise (z) direction. At the inlet plane ($x = 0$) we prescribe boundary layer profile and temperature:

$$u(0, y, z, t) = U_b(y, z), \tag{2.18}$$

$$v(0, y, z, t) = V_b(y, z), \tag{2.19}$$

$$w(0, y, z, t) = W_b(y, z) = 0, \tag{2.20}$$

$$t(0, y, z, t) = T_b(y, z), \tag{2.21}$$

where underscore b stands for Blasius. Values of U_b, V_b one gets from the solution of the equation (2.16) using formulas (2.17), also we assume $W_b = 0$. For temperature we prescribe $T_b = T_\infty$, which is given.

At the bottom plane ($y = 0$) we assume adiabatic wall conditions except for the cooling nozzle, where the profile and temperature of the cooling jet are prescribed, so everywhere at $y = 0$ except for the hole

$$u(x, 0, z, t) = v(x, 0, z, t) = w(x, 0, z, t) = 0, \tag{2.22}$$

$$\frac{\partial T(x, 0, z, t)}{\partial y} = 0. \tag{2.23}$$

At the exit of the film cooling hole we prescribe velocity profile and temperature distribution

$$u(x, 0, z, t) = U_j(x, z, t), \tag{2.24}$$

$$v(x, 0, z, t) = V_j(x, z, t), \tag{2.25}$$

$$w(x, 0, z, t) = W_j(x, z, t), \tag{2.26}$$

$$t(x, 0, z, t) = T_j(x, z, t), \tag{2.27}$$

where underscore j stands for jet. There are several ways to prescribe velocity profile and temperature distribution. The easiest is to assume a certain profile (like parabolic profile for the velocity and uniform temperature distribution) or, if possible, we can use results from experiments or external calculations. This is discussed in more details in Chapter 7.

At the top plane ($y = Y$) we prescribe boundary conditions of radiation type (more details one can find in Chapter 6).

At the outflow boundary ($x = X$) we assume stress free outflow in the tangential direction and that the derivative of the heat fluxes leaving the domain in streamwise direction is zero

$$\frac{\partial \tau_{xy}}{\partial x} = 0, \ \frac{\partial \tau_{xz}}{\partial x} = 0, \ \frac{\partial q}{\partial x} = 0. \tag{2.28}$$

The pressure at the outlet boundary is forced towards its reference value by imposing

$$\frac{\partial p}{\partial x} = \frac{p_\infty - p}{X}, \tag{2.29}$$

where subscript ∞ indicates a reference condition (again, as for velocity U_∞, p_∞ is free stream pressure).

More about boundary conditions and their derivation and numerical implementation one can find in Chapter 6.

Chapter 3

Local refinement strategies for flow problems

Many boundary value problems produce solutions that have highly localized properties. Quite a number of methods have been developed to solve such problems efficiently, most of them use the idea of grid refinement in the high activity regions and truly non-uniform methods are widely used. However, the use of uniform grids has several advantages compared to non-uniform ones. Most important advantages are simple discretization stencils, fast solution techniques for solution of the systems, which results from discretizations of the governing equations on the uniform grids, simple data structures. That is why the idea of local uniform grid refinement has been introduced back in the 70's. The most well known include adaptive mesh refinement (AMR) [9,52], which was successfully applied to hyperbolic problems, the fast adaptive composite grid (FAC) method [43] introduced for elliptic problems and local uniform grid refinement (LUGR) [76,81].

In this chapter we consider boundary value problems with solutions that have one or a few small regions with high activity, of which the film cooling problem is a typical example and present two methods to solve such problems, namely Local Uniform Grid Refinement and Local Defect Correction [25]. The essence of both is the use of several uniform grids with different grid sizes that cover different parts of the domain. At least one grid should cover the entire domain and in the regions with high activity we use local fine grids. The aim of the global coarse grid is to represent relatively smooth behavior of the solution outside the high activity regions and the use of one or more local fine grids allows us to represent the behavior of the solution in the corresponding high activity region. Note that such composite grids are highly structured and hence very simple data structures can be used, simplifying programming and data handling.

3.1 A complexity estimate

It has been mentioned before that one of the main advantages of using local grid refinement is the fact that while keeping the required resolution in the high activity area, we reduce the total number of grid points in the whole computational domain. Let us consider the efficiency of this procedure. An important theoretical instrument to judge the efficiency of a method is a complexity estimate. Below we present a typical case to compare the efficiency of the local grid refinement algorithms with an equivalent uniform fine grid.

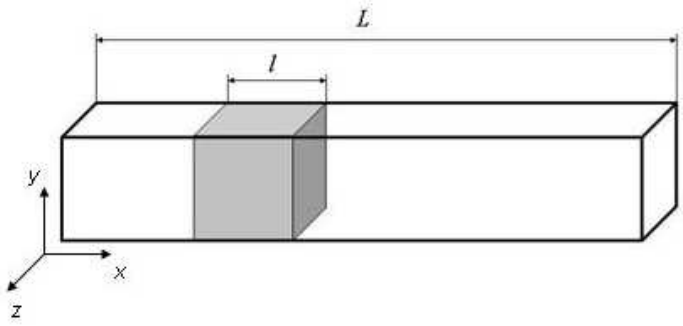


Figure 3.1: Cubic high activity area in a stretched domain.

To start with, consider a problem where the high activity subdomain is a cube within the global domain, as sketched in Figure 3.1. We assume that the global domain is a stretched block with a square cross-section. The area of high activity is a cube inside the global domain (see Figure 3.1).

Assume that the global domain is covered by a uniform grid. For simplicity we assume that the grid size is the same in each direction. In particular, the number of grid points is equal along the y- and z-axes, say N. Since the length of the global domain is larger and we assume that the mesh sizes are identical in every direction, the number of grid points in x-direction is larger than in the other two directions and is, say, AN ($A > 1$). The number of coarse grid points in the global domain is therefore equal to AN^3. Let the area of the high activity be covered by a fine mesh with the same grid size in every direction. We introduce the refinement factor $\sigma := H/h$, where H is the step size in the global grid and h is the step size for the local fine grid. Then in y- and z-direction we have again the same number of grid points, say M. Let the length of the high activity region in x-direction be l and let it be proportional to the length of the global domain L with factor β, $\beta = l/L$, $0 < \beta \leq 1$. Then the number of points needed for the local fine grid is $AN^3\sigma^3\beta$ and the total number of points needed for the local uniform grid refinement method is

$$N_{LUGR} = AN^3 + A\beta N^3\sigma^3. \tag{3.1}$$

For the equivalent uniform fine grid which covers the whole global domain we need the following number of points:

$$N_{un} = A^3 N^3 \sigma^3. \tag{3.2}$$

We can define the gain G in the number of grid points of LUGR over the number of points for the equivalent uniform fine grid as the following ratio

$$G := \frac{N_{un}}{N_{LUGR}}. \tag{3.3}$$

Substituting (3.1) and (3.2) into (3.3), we find that

$$G = \frac{AN^3\sigma^3}{AN^3 + A\beta N^3\sigma^3} = \frac{\sigma^3}{1+\beta\sigma^3}. \tag{3.4}$$

Assuming that $\beta \ll 1$ (the fine grid covers a small part of the global domain), we get that

$$G \approx \sigma^3. \tag{3.5}$$

3.2 Basics of the Local Uniform Grid Refinement technique

Local Uniform Grid Refinement (LUGR) is an adaptive grid technique for computing solutions of partial differential equations, that have local high activity regions. Using nested, finer and finer, uniform subgrids, the LUGR technique allows to refine the space grid locally around these high activity regions, and to avoid discretization on a very fine grid covering the entire physical domain. By refining only parts of the domain LUGR saves computer memory and computational time. LUGR should be contrasted to pointwise grid refinement, which leads to nonuniform grids. Typical problems solved with LUGR include steep moving fronts, boundary layers, moving pulses, etc [76]. For time-dependent problems, LUGR can be combined with static regridding. Static regridding means that while the solution advances in time, the space grid is adapted at discrete times.

The ideas, which lead to the LUGR technique, first appeared in [52]. The properties of LUGR for stationary problems were analysed in [81], in combination with explicit time schemes (namely Runge-Kutta) in [77] and for implicit Euler in [76].

The idea of the LUGR method can be briefly described as follows. Given a coarse base space grid and a base temporal step size, finer and finer uniform subgrids are generated in the areas of high activity (that is where solution changes rapidly). These subgrids have in general nonphysical boundaries and on each of these subgrids we compute local solution. They are generated up to the level of refinement good enough to resolve the anticipated fine scale structures (this scale is know in advance). Having completed

the refinement for the current base time step, the process is continued to the next base time step, while starting again from the base coarse grid.

An attractive feature of the static regridding LUGR approach is the possibility of dividing the whole solution into the following computational procedures: spatial discretization, temporal integration, error estimation, regridding and interpolation. Depending on the application, these individual procedures can be quite simple or very sophisticated. This flexibility is attractive since it makes it possible to treat different types of problems with almost the same code assuming that the grid structure and the associated data structure remains unchanged.

There are several versions of the LUGR algorithm, the most simple one uses two grids for stationary problems (see Figure 3.2).

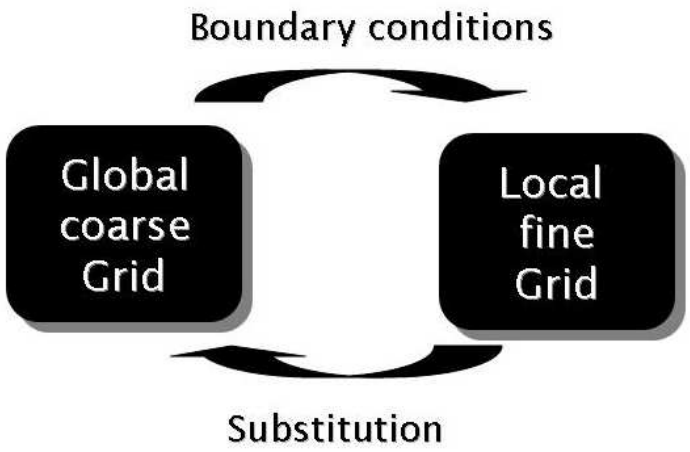


Figure 3.2: Scheme of the LUGR algorithm with two grids for stationary problems.

Algorithm 1

LUGR algorithm with two grids for stationary problems

1. Discretize and solve on the coarse grid
2. Decide where to place local fine grid
3. Get boundary conditions for the fine grid problem from the coarse grid
4. Discretize and solve on the fine grid
5. Update the coarse grid with the new solution of the fine grid

For time-dependent problems the LUGR algorithm has to be corrected taking into account the following issues: high activity can travel with a certain velocity through the computational domain and time advancement schemes have requirements on the ratio between time and space step sizes (like CFL condition). The description of the modified LUGR algorithms and their analysis for time-dependent problems one can find in [77] for explicit time schemes (namely Runge-Kutta) and in [76] for implicit Euler. For the film cooling problem we have a situation where the high activity region we are interested in is stationary in the space domain. Below one can find the LUGR algorithm for time-dependent problems with a stationary high activity region in case of an explicit time advancement scheme (see Figure 3.3). Due to the stability restrictions on the size of the time step, if we make the space step size smaller, we also have to take smaller time steps. Starting from the coarse grid solution at time t^n we make one "coarse" grid time step and several "fine" grid time steps to arrive at the point t^{n+1}. At t^{n+1} we substitute the fine grid solution for the coarse grid points that are located within the area of refinement. This all is summarized in Algorithm 2.

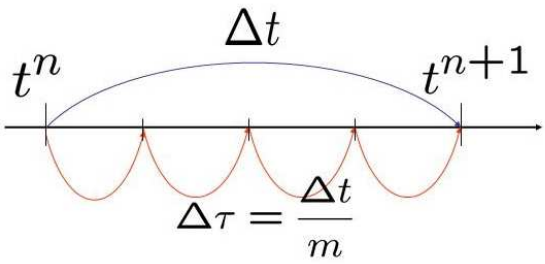


Figure 3.3: Scheme of the LUGR algorithm with two grids for time-dependent problems.

Algorithm 2

LUGR algorithm with two grids for time-dependent problems

1. Discretize and solve on the coarse grid with the coarse grid time step Δt from $t = t^n$ to $t = t^{n+1}$

2. Get boundary conditions for the fine grid problem from the coarse grid at $t = t^n$

3. Discretize and solve m-times on the fine grid with the time step $\tau = \frac{\Delta t}{m}$ starting from $t = t^n$, where m is the refinement factor in time.

4. Update the coarse grid at $t = t^{n+1}$ with the new solution of the fine grid at $t = m\tau$

3.3 Basics of the Local Defect Correction technique

As mentioned in the previous section, the LUGR technique has quite a lot of advantages, which makes it favorable for time-dependent problems with localized high activity areas, like boundary layer type of problems. However, as one can see from [45], there are certain situations when the LUGR algorithm is not able to produce adequate results. As was pointed out in [4], one of the main features of LDC compared to other local grid refinement methods is the use of two-way coupling between grids, as a simple approach of only coarse to fine grid may work for hyperbolic problems but not for elliptic [45]. Therefore in this section we introduce another local grid refinement strategy, namely Local Defect Correction (LDC) method (see [25]). In this method, which is an iterative process, a basic global discretization is improved by local discretizations defined in the subdomains. The update of the coarse grid solution is achieved by putting a defect correction term in the right hand side of the coarse grid problem. At each iteration step, the process yields a discrete approximation of the continuous solution on the composite grid. The discrete problem that is actually being solved is an implicit result of the iterative process. Therefore, the LDC method is both an iterative discretization and solution method.

The LDC technique was analysed in combination with finite difference discretizations in [20, 21] for diffusion and convection-diffusion problems . Combination of the LDC method with finite volume discretizations was proposed in [3,6,79] and application of it for numerical simulations of the flow and heat transfer in a glass tank was done in [49]. LDC for time-dependent problems is studied in [44, 45]. Further development of the LDC technique for parabolic problems in finite volume context one can find in [46]. The LDC method also allows to combine different grid types. In [50] a combination of a Cartesian coarse grid and a local polar grid is presented and in [24] a Cartesian coarse grid is combined with a slanted local one.

Let us consider the following stationary problem

$$\begin{cases} Lu = -\epsilon\nabla^2 u(x,y) + \mathbf{c}\cdot\nabla u(x,y) = f(x,y) \text{ in } \Omega, \\ u = g \text{ on } \Gamma. \end{cases} \tag{3.6}$$

In (3.6), L is a linear elliptic differential operator, and f and g are the source term and Dirichlet boundary condition, respectively, $\epsilon > 0$ is the diffusion coefficient, $\mathbf{c}$ is the convection coefficient, u is the unknown function, Ω is the domain of interest and Γ is the boundary of this domain.

Following [5], we introduce the basics of the Local Defect Correction technique. In order to discretize (3.6), we first choose a global coarse grid (grid size H), which we denote by Ω^H . The next step is to find an initial approximation $\mathbf{u}_0^H$ on Ω^H by solving the system

$$\mathbf{L}^H\mathbf{u}_0^H = \mathbf{f}^H, \tag{3.7}$$

which is a discretization of boundary value problem (3.6). In (3.7), the right hand side $\mathbf{f}^H$ incorporates the source term f as well as the Dirichlet boundary condition g.

Now, assume that the continuous solution u of (3.6) has a high activity region in some (small) part of the domain. This high activity could either be caused by the boundary conditions or the source term. We would like to capture this high activity of u by discretizing (3.6) on a composite grid. The position of this high activity region can be detected by various methods, for example by calculating the gradient, see [8] for details. So we choose $\Omega_l \subset \Omega$ such that the high activity region of u is contained in Ω_l. If we have more than one high activity region, one may take more regions of refinement. In Ω_l, we choose a local fine grid (with grid size h), which we denote by Ω_l^h. The fine grid is chosen such that $\Omega^H \cap \Omega_l \subset \Omega_l^h$, i.e., grid points of the global coarse grid that lie in the area of refinement belong to the local fine grid too. See Figure 3.4 for an example composite grid.

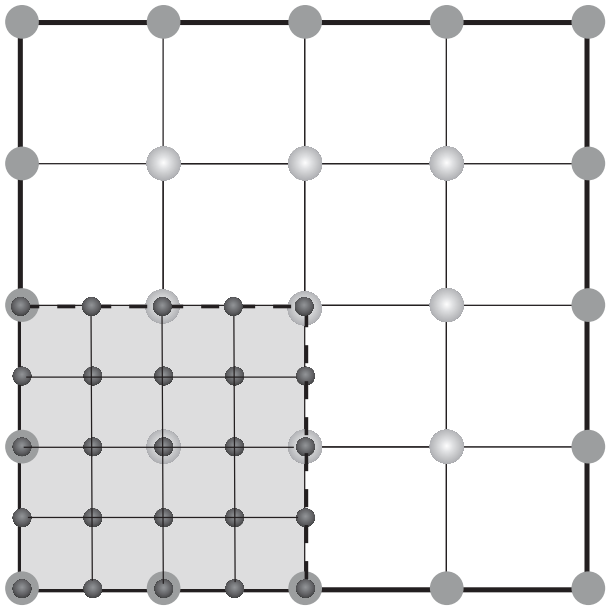


Figure 3.4: A global coarse and a local fine grid. The darker area is the area of high activity Ω_l. The interface Γ is dashed. Large light colored circles are nodes of the coarse grid, large dark colored circles are boundary points, smaller circles, not located on the domain boundary or the interface, are nodes of the local fine grid.

Now we have to define a local discrete problem on Ω_l. So we define artificial boundary conditions on Γ, the interface between Ω_l and $\Omega \setminus \Omega_l$. Since on Γ we have more fine grid points than coarse grid ones, we prescribe artificial Dirichlet boundary conditions by applying an interpolation operator $P^{h,H}$. The operator $P^{h,H}$ maps function values at grid points of the coarse grid that lie on the interface, denoted by Γ^H, to function values at grid points of the fine grid that lie on the interface, denoted by Γ^h. If we denote the

vector space of grid functions on Γ^H by $G(\Gamma^H)$, and likewise introduce $G(\Gamma^h)$, we have $P^{h,H} : G(\Gamma^H) \longrightarrow G(\Gamma^h)$. In practice, we take $P^{h,H}$ to be the linear interpolation operator on the interface for simplicity.

In this way, we find the following approximation $\mathbf{u}^h_{l,i}$, $i = 0$, on Ω^h_l

$$\mathbf{L}^h_l \mathbf{u}^h_{l,i} = \mathbf{f}^h_l \left(\mathbf{u}^H_i|_\Gamma \right). \tag{3.8}$$

In (3.8), the matrix $\mathbf{L}^h_l$ is a discrete approximation of the differential operator L on the subdomain Ω_l, and the first term on the right hand side $\mathbf{f}^h_l$ incorporates the source term f as well as the Dirichlet boundary condition g on $\partial\Omega_l \backslash \Gamma$ given in (3.6). The dependence on the coarse grid approximation via the artificial Dirichlet boundary condition is made explicit by writing $\mathbf{f}^h_l(\mathbf{u}^H_i|_\Gamma)$.

When boundary value problem (3.6) has been discretized and solved on a coarse grid, and when an area of the coarse grid has been refined and a local solution has been calculated on the finer grid, we can define a composite grid approximation $\mathbf{u}^{H,h}_0$ as

$$\mathbf{u}^{H,h}_0(x,y) := \begin{cases} \mathbf{u}^h_{l,0}(x,y), & (x,y) \in \Omega^h_l, \\ \mathbf{u}^H_0(x,y), & (x,y) \in \Omega^H \setminus \Omega^h_l. \end{cases} \tag{3.9}$$

So for the coarse grid points within the region of the refinement we have two solutions, one coming from the coarse grid and another from the fine grid. We will now use the local fine grid solution to update the coarse grid approximation. This update can be achieved by projecting the more accurate fine grid solution onto the local coarse grid, and by calculating the residual of the projected solution. Since the coarse grid points belong to the fine grid too, we simply restrict the fine grid solution to those points. The calculated residual is an estimate of the local discretization error of the coarse grid discretization. The estimate is used to formulate a modified discrete problem on the coarse grid. This is considered in more detail below.

The grid points of the coarse grid will be partitioned as $\Omega^H = \Omega^H_l \cup \Gamma^H \cup \Omega^H_c$, where $\Omega^H_l := \Omega^H \cap \Omega_l, \Gamma^H := \Omega^H \cap \Gamma$ and $\Omega^H_c := \Omega^H \setminus (\Omega^H_l \cup \Gamma^H)$. If we would substitute the projection on Ω^H of the exact solution u of boundary value problem (3.6) into the coarse grid discretization (3.6), we would find the local discretization error or defect $\mathbf{d}^H$, given by $\mathbf{L}^H(u|_{\Omega^H}) = \mathbf{f}^H + \mathbf{d}^H$. If we would know the values of the defect $\mathbf{d}^H$, we could use them to find a better approximation on the coarse grid. This could be achieved by putting the defect vector on the right hand side of (3.6). However, as we do not know the exact solution of the boundary value problem, we cannot calculate $\mathbf{d}^H$. What we can do though, is to use the approximation $\mathbf{u}^h_{l,0}$ calculated on the local fine grid to estimate $\mathbf{d}^H$ at the coarse grid points $(x,y) \in \Omega^H_l$. Define $\mathbf{w}^H_0 \in G(\Omega^H)$ as the global coarse grid function of best approximations sofar, i.e.

$$\mathbf{w}^H_0(x,y) := \begin{cases} \mathbf{u}^h_{l,0}(x,y), & (x,y) \in \Omega^H_l, \\ \mathbf{u}^H_0(x,y), & (x,y) \in \Gamma^H \cup \Omega^H_c. \end{cases} \tag{3.10}$$

Next, we estimate the defect by $\mathbf{d}^H = \mathbf{L}^H(u|_{\Omega^H}) - \mathbf{f}^H \approx \mathbf{L}^H\mathbf{w}_0^H - \mathbf{f}^H =: \mathbf{d}_0^H$. Assuming that the stencil at grid point (x, y) involves (at most) function values at $(x + iH, y + jH)$ with $i, j \in \{-1, 0, 1\}$, $\mathbf{d}_0^H$ provides an estimate of the local discretization error of the coarse grid discretization at all points of Ω_l^H. Therefore, we can update the coarse grid approximation by placing the estimate at the right hand side of the coarse grid equation (3.6). This leads to the coarse grid correction step to find $\mathbf{u}_i^H$, $i = 1$, on the coarse grid

$$\mathbf{L}^H\mathbf{u}_i^H = \mathbf{f}_{i-1}^H, \tag{3.11}$$

where

$$\mathbf{f}_i^H(x, y) := \begin{cases} \mathbf{f}^H(x, y) + \mathbf{d}_i^H(x, y), & (x, y) \in \Omega_l^H, \\ \mathbf{f}^H(x, y), & (x, y) \in \Gamma^H \cup \Omega_c^H. \end{cases}$$

The correction step (3.11) produces a new solution $\mathbf{u}_1^H$ on the coarse grid. Because (3.11) incorporates estimates of the local discretization error of the coarse grid discretization, the new solution $\mathbf{u}_1^H$ is assumed to be more accurate than $\mathbf{u}_0^H$. Hence, the new solution $\mathbf{u}_1^H$ provides a better boundary condition on the interface. A better solution on the local fine grid can be found as before by solving (3.8) with $i = 1$.

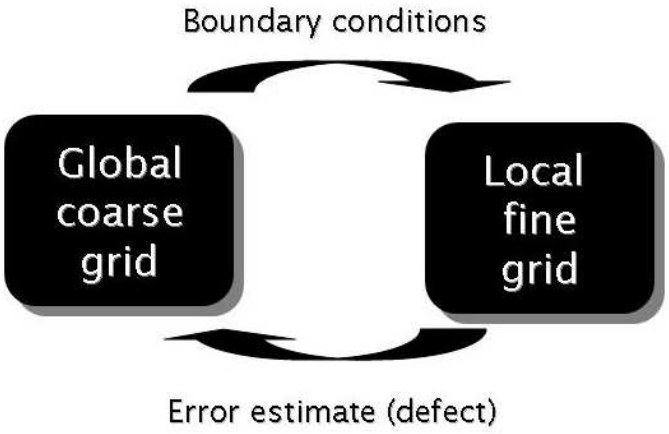


Figure 3.5: Scheme of the LDC algorithm with two grids.

To summarize, we have the following iterative method (with desired accuracy δ) (see Figure 3.5).

Algorithm 3.
Two-grid LDC algorithm with area of refinement chosen *a priori*
Initialization

- Solve the basic coarse grid problem (3.7).

- Solve the local fine grid problem (3.8).

Iteration, $i = 1, 2, \ldots$, *until* $\|u_i^{H,h} - u_{i-1}^{H,h}\|_\infty \leq \delta$

- Solve the updated coarse grid problem (3.11).
- Solve the local fine grid problem (3.8).

For the time–dependent case let us consider the following two-dimensional problem:

$$\begin{cases} \dfrac{\partial u(x,y,t)}{\partial t} = Lu(x,y,t) + f(x,y,t) \text{ in } \Omega \times \Theta, \\ u(x,y,t) = \psi(x,y,t) \text{ on } \Gamma \times \Theta, \\ u(x,y,0) = \phi_0(x,y), \end{cases} \tag{3.12}$$

where $\Theta = (0, t_{end})$ is the time interval, Ω, Γ, L were introduced before, $\psi(x,y,t)$ is the boundary condition and $\phi_0(x,y)$ is the initial condition. We assume that u has a region of high activity that covers a small part of Ω. We discretize our problem (3.12) in time using an implicit time scheme (one can think of implicit Euler as the simplest example). As for space discretization, we can use any scheme of our choice, which provides stable results (for example second–order finite differences for diffusive terms and first-order upwind for convective). Again, as in previous examples, the LDC method does not restrict us to use the same schemes for the fine and coarse solution. In addition to the notation used in previous section, we introduce the following things: superscript ^{n} for indication of the time level n, coarse grid time step $\Delta t = t^{n+1} - t^n$, fine grid time step $\tau = \frac{\Delta t}{m}$, where m is the time refinement factor (one can think for simplicity that $m = \sigma$). For time-dependent problems we also have to provide initial conditions. The natural way to get an initial condition for the fine grid is to interpolate values from the available coarse grid solution. Another difficulty is that the artificial boundary conditions may change from one fine grid time step to another (but still being within the same coarse time step). In order to tackle this we can use time interpolation of the corresponding values between time t^n and t^{n+1}. More detailed information on time-dependent LDC techniques one can find in [44].

For time–dependent problems, we can use the following LDC algorithm [44]

Algorithm 4.
LDC algorithm for a time-dependent problem
FOR LOOP (time), $n = 1, 2, \ldots, N$
Initialization

- Solve the basic coarse grid problem and get $u_{H,0}^n$
- Choose a region of refinement, introduce a fine grid Ω_h^l on it, divide the time step $[t_{n-1}, t_n]$ in subintervals $\Delta\tau$.
- Compute an initial condition for the local problem.
- Compute a boundary condition for the local problem.

- Compute a local approximation $u_{h,0}^{n,l}$, solving the local problem
  Iteration, $i = 1, 2, \ldots, \kappa$
  - Use $u_{h,i-1}^{n,l}$ to compute an approximation of the local discretization error
  - Solve the updated coarse grid problem and get $u_{H,i}^{n}$.
  - Use $u_{H,i}^{n}$ to update boundary conditions for the local problem
  - Solve the local fine grid problem.

  End iteration on i
- Go to the next time step

END FOR LOOP (time)

Chapter 4

Analysis of the convergence of the Local Defect Correction technique

In this chapter we study the convergence behavior of the Local Defect Correction (LDC) technique, introduced in Section 3.3. It should be noted that previously LDC was used mostly for diffusion problems [3]. For flow problems (like the film cooling one) we have both convection and diffusion, so we need additional testing for LDC on such problems. In this chapter the major objective is to extend the results for pure diffusion problems to convection-diffusion problems. The model problem of interest we study in this chapter is given by

$$\begin{cases} \epsilon u''(x) + c u'(x) = f(x), & c, \epsilon > 0 \qquad x \in (0,1), \\ u(0) = u_0, \quad u(1) = u_1, \end{cases} \tag{4.1}$$

where the diffusion parameter ϵ may be (very) small. We introduce the parameter $\kappa = c/\epsilon$. Previous papers [3, 19] studied convection-diffusion equations, where the convection and diffusion coefficients are of the same order of magnitude. However, we would like to study the properties of the LDC technique for convection-dominated problems too. For our experiments we specifically take the convection coefficient c to be fixed and the diffusion coefficient ϵ to be variable. For small ϵ this approach could also be interpreted as a transformation of the pure convection problem into a two point boundary value problem by means of adding 'small' diffusion.

4.1 Theoretical estimation of the convergence rate

As one can see from Section 3.3, the LDC algorithm is an iterative process. According to [5], the LDC iteration can be expressed in terms of an iteration matrix, whose properties are studied analytically and experimentally for the two-dimensional Poisson equation discretized by finite differences. In general, it is observed that LDC converges very fast and that iteration errors are reduced by several orders of magnitude at each iteration step. In [5] it is shown that the LDC iteration can be formulated in terms of coarse grid points located at the interface only. Suppose we have a high activity area in the left part of the one-dimensional domain, from point $x = 0$ to point $x = \gamma$, so $\Omega_l = (0, \gamma)$. For our one-dimensional problem, the interface between the global coarse and local fine grid consists of the single point $x = \gamma$. For this reason, the iteration matrix $\mathbf{M}$ from [5] is a scalar quantity that we will denote by M. We would like to have fast convergence of our iterative process and therefore we want to study the influence of various parameters on the convergence rate. In the following sections we present first theoretical results for the convergence rate, which in the subsequent section are illustrated by numerical experiments.

We can write the following expression M

$$M = \mathbf{M}_1 \mathbf{M}_2, \tag{4.2}$$

where

$$\mathbf{M}_1 = \begin{pmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \end{pmatrix} \left(\mathbf{L}^H\right)^{-1} \begin{pmatrix} \mathbf{I} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{4.3}$$

$$\mathbf{M}_2 = \mathbf{B}^H_{l,\Gamma} - \mathbf{L}^H_l \mathbf{R}^{H,h} \left(\mathbf{L}^h_l\right)^{-1} \mathbf{B}^h_{l,\Gamma}, \tag{4.4}$$

where $\mathbf{R}^{H,h}$ is a restriction operator from the fine to the coarse grid, $\mathbf{L}^H$ is is a discrete approximation of the differential operator L on the global domain, $\mathbf{L}^h_l$ is a discrete approximation of the differential operator L on the subdomain Ω_l and $\mathbf{B}^H_{l,\Gamma}$ reflects the dependence on artificial Dirichlet boundary condition.

In the following we look at $\mathbf{M}_2$ since it has the major effect on the convergence rate. Indeed, we have

$$|M| = \| \mathbf{M} \|_\infty \leq \| \mathbf{M}_1 \|_\infty \| \mathbf{M}_2 \|_\infty \leq \frac{1}{8} \| \mathbf{M}_2 \|_\infty .$$

The last inequality is a well known result, see, e.g., [26]. We would like to show that $|M| < 1$, which means that the LDC algorithm converges to a fixed point. For this reason we let $g \in \mathbb{R}$ and analyse $\mathbf{M}_2 g$. Consider

$$\mathbf{M}_2 g = \left(\mathbf{B}^H_{l,\Gamma} - \mathbf{L}^H_l \mathbf{R}^{H,h} \left(\mathbf{L}^h_l\right)^{-1} \mathbf{B}^h_{l,\Gamma}\right) g = \mathbf{B}^H_{l,\Gamma} g - \mathbf{L}^H_l \mathbf{R}^{H,h} \mathbf{v}^h_l, \tag{4.5}$$

where

$$\mathbf{v}^h_l = \left(\mathbf{L}^h_l\right)^{-1} \mathbf{B}^h_{l,\Gamma} g.$$

Rewriting, we find

$$\mathbf{L}_l^h \mathbf{v}_l^h + \mathbf{B}_{l,\Gamma}^h (g) = 0. \tag{4.6}$$

According to the definitions of the matrices $\mathbf{L}_l^h$, $\mathbf{B}_{l,\Gamma}^h$, this is a discretization on the *local fine grid* of the following boundary value problem

$$\begin{cases} \epsilon u''(x) + c u'(x) = 0, \\ u(0) = 0, \quad u(\gamma) = -g. \end{cases} \tag{4.7}$$

In the following we study the convergence behavior of the LDC algorithm and consider the following combinations of discretization schemes

- Central differences both for fine grid and coarse grid. Here (on the coarse grid e.g.) we approximate the second derivative u_i'' by the following expression

$$u_i'' = \frac{u_{i-1} - 2u_i + u_{i+1}}{H^2} \tag{4.8}$$

  and the first derivative u_i' by

$$u_i' = \frac{u_{i+1} - u_{i-1}}{2H}. \tag{4.9}$$

- Upwind both for fine grid and coarse grid. We approximate the second derivative by (4.8) and the first derivative by

$$u_i' = \phi \frac{u_i - u_{i-1}}{H} + (1 - \phi) \frac{u_{i+1} - u_i}{H}, \tag{4.10}$$

  where

$$\phi = \begin{cases} 1, & c < 0 \\ 0, & c \geq 0 \end{cases}$$

  and c is the convection coefficient in (4.1).

- Upwind for coarse grid and central differences for fine grid. In this case we use (4.8) and (4.10) on the coarse grid and (4.8) and (4.9) on the fine grid.

4.2 Analysis for central differences on both grids

Assume we use the central difference scheme (see (4.8) and (4.9)) both for fine and coarse grids. Using the fact that we can get an explicit formula for the solution $\mathbf{v}_l^h$ (see, for example [64]), we can write the following

$$\mathbf{v}_l^h(i) = -g \frac{1 - \tau^i}{1 - \tau^n}, \qquad i = 0, 1, \ldots, n, \tag{4.11}$$

with

$$\tau = \frac{1+d}{1-d}, \qquad d = -\frac{ch}{2\epsilon} = -\frac{\kappa h}{2}.$$

Now we analyse

$$\mathbf{z} := \mathbf{M}_2 g = \mathbf{B}^H_{l,\Gamma} g - \mathbf{L}^H_l \mathbf{R}^{H,h} \mathbf{v}^h_l. \tag{4.12}$$

According to (4.4), $\mathbf{z}$ is the residual we find by substituting the local fine grid solution $\mathbf{v}^h_l$ into the coarse discretization.

Therefore we can write the following expression for z_i, the i-th component of $\mathbf{z}$

$$\begin{aligned} z_i =& \frac{1}{H^2}\left[\left(-1-\frac{\kappa H}{2}\right)\left(\mathbf{v}^h_l\right)_{i-1} + 2\left(\mathbf{v}^h_l\right)_i + \left(-1+\frac{\kappa H}{2}\right)\left(\mathbf{v}^h_l\right)_{i+1}\right] \\ =& \frac{-g}{H^2}\left[\left(-1-\frac{\kappa H}{2}\right)\frac{1-\tau^{\sigma(i-1)}}{1-\tau^n} + 2\frac{1-\tau^{\sigma i}}{1-\tau^n} + \left(-1+\frac{\kappa H}{2}\right)\frac{1-\tau^{\sigma(i+1)}}{1-\tau^n}\right], \end{aligned} \tag{4.13}$$

where $\sigma := H/h$ is the refinement factor and the subscript i, $i = 1, 2, \ldots, k-1$ denotes the coarse grid point inside the fine grid region.

We find

$$z_i = \frac{-g\tau^{\sigma(i-1)}}{1-\tau^n}\frac{1}{(1-d)^\sigma}\frac{1}{H^2}\left[(1+\sigma d)(1-d)^\sigma - 2(1+d)^\sigma + (1-\sigma d)\frac{(1+d)^{2\sigma}}{(1-d)^\sigma}\right]. \tag{4.14}$$

As one can see from (4.14), the convergence rate depends on the following factors: coarse grid mesh size H, refinement factor σ (or, which is the same, fine grid mesh size h), factor $\kappa = c/\epsilon$. Below we study the influence of each of these parameters on z_i separately.

First we study the influence of the coarse grid mesh size H on the convergence rate, so we fix κ and σ and make a Taylor series expansion around $d = 0$ (since $d = ch/(2\epsilon)$, so $d = cH/(2\epsilon\sigma) = \kappa H/(2\sigma)$ and $d \to 0$ is equivalent with $H \to 0$).

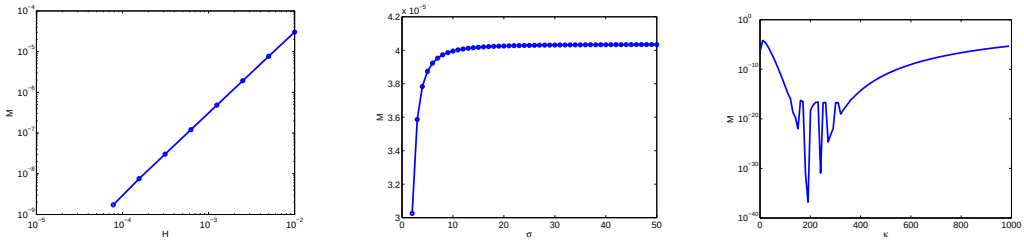
$$\begin{aligned} z_i =& \frac{-g\tau^{\sigma(i-1)}}{1-\sigma_1^n}\frac{1}{(1-d)^\sigma}\frac{1}{H^2}\left[\frac{-4}{3}\left(1-\frac{1}{\sigma^2}\right)(\sigma d)^4 + O(d^5)\right] \\ =& \left(1-\frac{1}{\sigma^2}\right)\frac{g\tau^{\sigma(i-1)}}{1-\tau^n}\frac{1}{(1-d)^\sigma}\frac{\kappa^4}{12}H^2 + O(H^3). \end{aligned} \tag{4.15}$$

For fixed κ and σ, $\frac{1}{(1-d)^\sigma}$ converges to 1 for $H \to 0$. The only terms that depend on H left are $\frac{g\tau^{\sigma(i-1)}}{1-\tau^n}$ and H^2. We analyze the behavior of the first for $H, h \to 0$. Expansion gives:

$$\tau = \frac{1-\frac{\kappa h}{2}}{1+\frac{\kappa h}{2}} = \left(1-\frac{\kappa h}{2}+O(h^2)\right)^2 = 1-\kappa h + O(h^2),$$

$$\tau^n = \left(1 - +\kappa h + O(h^2)\right)^n = \left(1 - \frac{\kappa\gamma}{2} + O\left(\frac{1}{n^2}\right)\right)^n \to \exp\left(-\kappa\gamma\right) \qquad (n \to \infty).$$

From (4.15) we can see that the convergence rate depends on the coarse grid size H as $O(H^2)$ and does not depend on the fine grid size h. In Figure 4.1 we present dependence of the convergence factor M on different variables: coarse grid mesh size H (in this case we keep $\kappa = 20$, $\sigma = 2$), refinement factor σ ($\kappa = 20$, $H = 0.01$) and factor $\kappa = c/\epsilon$ ($H = 0.01$, $\sigma = 2$). As one can see in Figure 4.1 (a), we have $O(H^2)$ dependence of the convergence rate on coarse grid mesh size H indeed.



(a) $\kappa = 20$, $\sigma = 2$ (b) $\kappa = 20$, $H = 0.01$ (c) $H = 0.01$, $\sigma = 2$

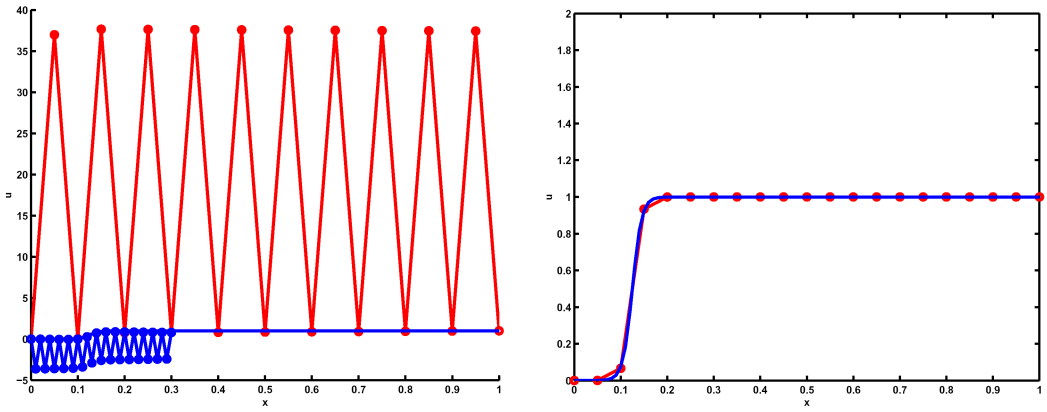
Figure 4.1: Dependence of the convergence factor M on the coarse grid mesh size H (a), refinement factor σ (b) and factor $\kappa = c/\epsilon$. (c)

Next we study the influence of the refinement factor σ on the convergence rate. From (4.14) we expect to have a weak dependence on the refinement factor. To see if this is true, we fix all parameters for the fine grid except the fine grid step size h, and therefore σ, and calculate the convergence factor M. As can be seen from Figure 4.1 (b), indeed we have minor dependence of the convergence factor M on the refinement factor.

Finally, we study the influence of the factor $\kappa = c/\epsilon$ on the convergence rate. Since our problem of interest is convection dominated, using the central difference scheme for the coarse grid discretization is not desirable. Indeed, the stability of central differences requires the mesh size to be of order ϵ to avoid oscillations; and that is the behavior we see in Figure 4.1 (c). As long as the Péclet number is smaller than 2 (stability constraint for central differences), the convergence factor decreases as κ increases. For $\kappa > 200$, the central difference scheme breaks down. This is too restrictive and in such a case we do not need LDC at all, since with such a small step size we will resolve the whole problem. However, the amount of computational work increases tremendously. For this reason the central difference approximation is not suitable as a coarse grid approximation in our LDC method for convection-dominated problems.

Nevertheless, note that one of the nice properties of the LDC technique is that it converges, even for unstable numerical schemes, both on fine and coarse grids. An illustration of this fact can be found in Figure 4.2. The Péclet number Pe, calculated for

the coarse grid mesh size is 5000 and for the fine grid 1000, so our central difference scheme is completely unstable (see Figure 4.2 (a), solid line is exact solution). However, after a number of iterations the solution becomes better and better, and the LDC method slowly converges (see Figure 4.2 (b)). Because the local grid is much finer than



(a) Initial solution (b) Solution after 20 iterations

Figure 4.2: Central differences both for fine and coarse grids.

the global one, there is one more possibility to use central differences namely for the fine grid approximation, a detailed analysis of which is presented in Section 4.4.

4.3 Analysis for upwind scheme on both grids

Next we consider the upwind discretization scheme for the convection term in (4.1) (see (4.8)) and we use it both for the fine and coarse grid discretization. The first part of the analysis is the same as in Section 4.1 up to (4.7). For the upwind scheme we get the following expression for $\mathbf{v}_l^h$

$$\mathbf{v}_l^h(i) = -g\frac{1-\tau^i}{1-\tau^n}, \qquad i = 0, 1, \ldots, n, \tag{4.16}$$

with

$$\tau = 1 + d, \qquad d = -\frac{ch}{\epsilon} = -\kappa h.$$

Proceeding as in Section 4.2 (see (4.13)-(4.14)), the components of

$$\mathbf{z} := \mathbf{M}_2 g = \mathbf{B}_{l,\Gamma}^H g - \mathbf{L}_l^H \mathbf{R}^{H,h} \mathbf{v}_l^h \tag{4.17}$$

can be written as

$$z_i = \frac{-g\tau^{\sigma(i-1)}}{1-\tau^n}\frac{1}{H^2}\left[1+\sigma d-2(1+d)^\sigma-\sigma d(1+d)^\sigma+(1+d)^{2\sigma}\right]. \tag{4.18}$$

As one can see from (4.18), the convergence rate depends on the coarse grid mesh size H, the refinement factor σ and the factor $\kappa = c/\epsilon$. We now analyse the effect of H, σ and κ more carefully.

First we study the influence of the coarse grid mesh size H on the convergence rate. We can show that for fixed κ and σ we have

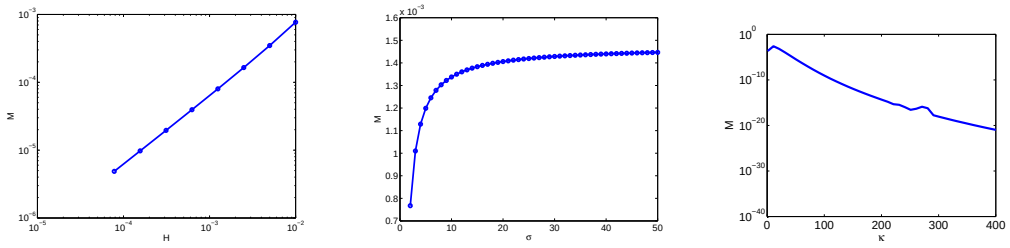
$$z_i = \frac{1}{2}g\frac{\tau^{\sigma(i-1)}}{1-\tau^n}\kappa^3\frac{\sigma-1}{\sigma}H + O(H^2). \tag{4.19}$$

Now we can analyse each term in the relation (4.19). The factor $\frac{1}{2}g\kappa^3\frac{\sigma-1}{\sigma}$ is just a constant. The only term left is $\frac{\tau^{\sigma(i-1)}}{1-\tau^n}$. We analyse its behavior for $H, h \to 0$.

Expansion gives:

$$\tau^n = \left(1-\kappa h+O(h^2)\right)^n = \left(1-\frac{\kappa\gamma}{n}+O\left(\frac{1}{n^2}\right)\right)^n \to \exp\left(-\kappa\gamma\right) \qquad (n\to\infty).$$

From (4.19) we can see that our convergence rate depends on the coarse grid size H as $O(H)$ and does not depend on the fine grid size h. This type of behavior one can observe in Figure 4.3 (a). Next we study influence of the refinement factor σ on the convergence rate. If we fix H and σ, we expect the convergence factor to have a minor dependence on the refinement factor. To see if this is true, we fix all the parameters except fine grid step size h, and therefore σ, and calculate the convergence factor M. As can be seen from Figure 4.3 (b), indeed we have only weak dependence of the convergence factor M on the refinement factor. Finally, we investigate the influence of the factor $\kappa = c/\epsilon$ on



(a) $\kappa = 20$, $\sigma = 2$ (b) $\kappa = 20$, $H = 0.01$ (c) $H = 0.01$, $\sigma = 2$

Figure 4.3: Dependence of the convergence factor M on the coarse grid mesh size H (a), refinement factor σ (b) and factor $\kappa = c/\epsilon$. (c)

the convergence rate. If we fix H and σ, we expect fast convergence for $\kappa \gg 1$, because

in this case the factor $\frac{\tau^{\sigma(i-1)}}{1-\tau^n}$ in (4.19) tends to zero. Indeed, as one can see in Figure 4.3 (c), with ϵ tending to zero (that is $\kappa = c/\epsilon \to \infty$) we have a dramatic improvement of the convergence factor M.

4.4 Analysis for upwind scheme on the coarse grid and central difference scheme on the fine grid

As it was mentioned in Section 4.2, there is one more possibility to use central differences, namely for the fine grid approximation. In this case we have to use a small mesh size precisely in the region where we really need it. In order to better understand the convergence properties of the combination upwind for the coarse grid and central differences for the high activity region, we perform an analysis, similar to those in Sections 4.2, 4.3. For this discretization we obtain the following expression for z_i , $i = 1, 2, \ldots, k-1$

$$z_i = \frac{1}{H^2}(-g)\frac{\tau^{\sigma(i-1)}}{1-\tau^n}\left[1 + 2\sigma d - 2\left(\frac{1+d}{1-d}\right)^{\sigma} - 2\sigma d\left(\frac{1+d}{1-d}\right)^{\sigma} + \left(\frac{1+d}{1-d}\right)^{2\sigma}\right], \quad (4.20)$$

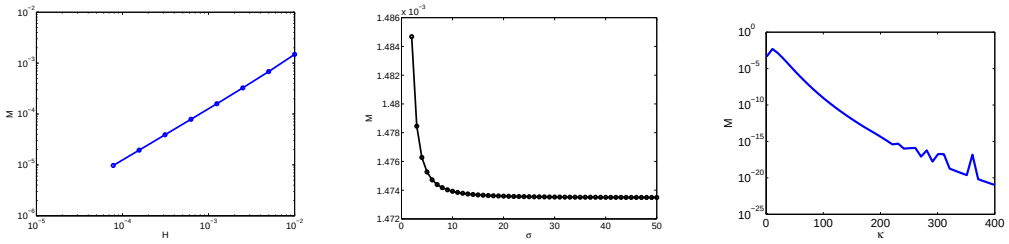
where

$$\sigma = H/h, \qquad d = -\frac{ch}{\epsilon} = -\kappa h, \qquad \tau = 1 + d.$$

For fixed κ and σ we get

$$z_i = (-g)\frac{\tau^{\sigma(i-1)}\kappa^3}{1-\tau^n}H + O(H^2). \quad (4.21)$$

As can be seen from (4.21), for the combination of upwind for the coarse grid and central difference for the fine grid our convergence factor M depends on the coarse grid size H like $O(H)$ and this is the same behavior as for the combination upwind plus upwind. This can be seen in Figure 4.4 (a). Again we want to study the influence of different factors on the convergence rate. We expect to have weak dependence on the refinement factor. To see if it is true, we fix all the parameters except the fine grid step size h, and therefore σ, and calculate the convergence factor M. As can be seen from Figure 4.4 (b), indeed we have minor dependence of the convergence factor M on the refinement factor. Finally we study the influence of the refinement factor $\kappa = c/\epsilon$ on the convergence rate. For fixed H and σ, we expect fast convergence in the case $\kappa \gg 1$. Indeed, as one can see in Figure 4.4 (c), with ϵ tending to zero (that is $\kappa = c/\epsilon$ tends to infinity) we have a dramatic improvement of the convergence factor M. This can be explained by the fact that in this case the factor $\frac{\tau^{\sigma(i-1)}\kappa^3}{1-\tau^n}$ in (4.21) tends to zero.



(a) $\kappa = 20$, $\sigma = 2$ (b) $\kappa = 20$, $H = 0.01$ (c) $H = 0.01$, $\sigma = 2$

Figure 4.4: Dependence of the convergence factor M on the coarse grid mesh size H (a), refinement factor σ (b) and factor $\kappa = c/\epsilon$ (c)

4.5 Numerical results

The first goal of the numerical simulations is to check whether our LDC algorithm works for convection-diffusion equations and whether the convergence in the numerical simulations is as predicted in Sections 4.1-4.4. We will also compare the observed convergence behavior with the predictions of the convergence rate according to [5, Theorem 2].

We apply the theory now to (4.1). The source term f and the boundary conditions u_0 and u_1 are chosen such that

$$u(x) = \frac{1}{2}(\tanh(50(x - 1/8)) + 1). \tag{4.22}$$

The following parameters were chosen for computations: $\kappa = 100$ (convection-dominated) or $\kappa = 0.1$ (diffusion-dominated), $\gamma = 0.3$. As discretization schemes we choose central differences or upwind.

We estimate the convergence rate M by M_i defined as

$$M_i := \|\Delta f_i^H\|_\infty / \|\Delta f_{i-1}^H\|_\infty, \qquad i = 1, 2, \ldots \tag{4.23}$$

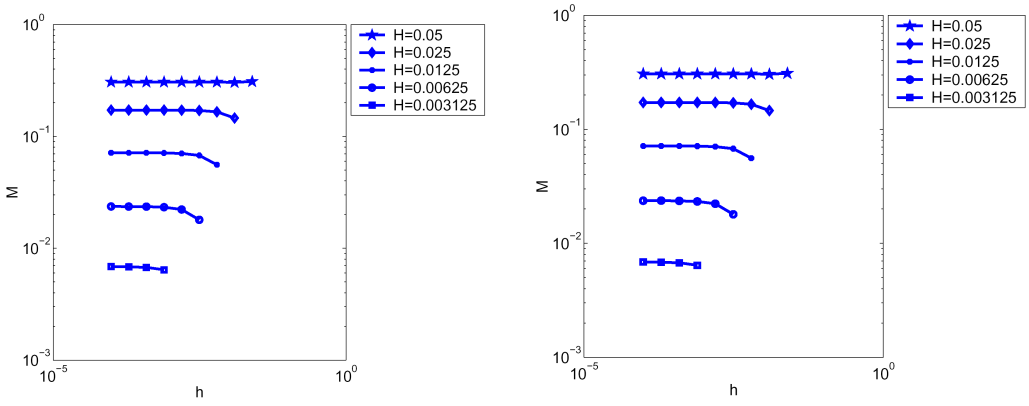
where $\Delta f_i^H = f_i^H - f_{i-1}^H$ and f_i^H is

$$f_i^H(x, y) := \begin{cases} f^H(x, y) + d_i^H(x, y), & (x, y) \in \Omega_l^H, \\ f^H(x, y), & (x, y) \in \Gamma^H \cup \Omega_c^h. \end{cases} \tag{4.24}$$

Of course, as one can see from (4.23), our estimate becomes less reliable if we have fast convergence and thus small M_i , say $M_i < 10^{-8}$. In Section 4.1 we have found that the convergence in the numerical simulations depends on H, σ and κ. To verify the dependence found in theory, we will first study convection-dominated problems ($\kappa = 100$) and then diffusion-dominated problems ($\kappa = 0.01$).

As for numerical experiments, we provide several examples, which represent typical results we get.

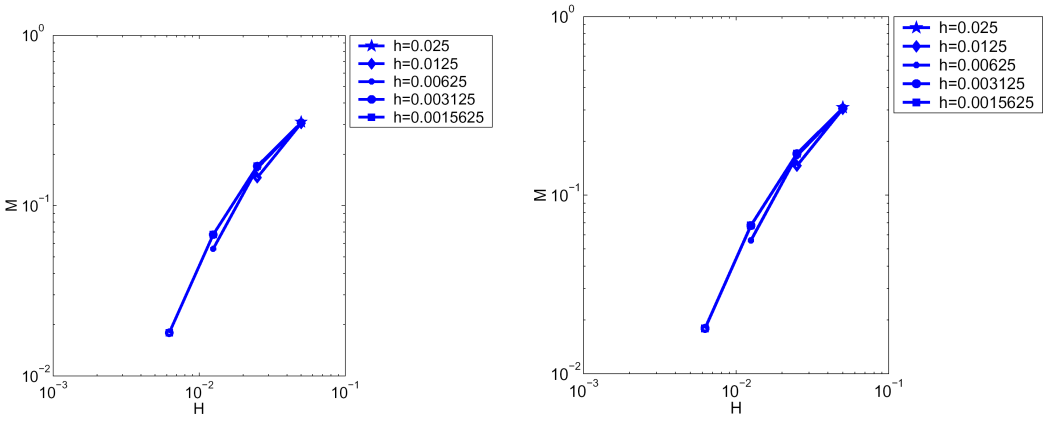
Example 1. Convection-dominated problems. Central difference scheme. First we would like to study the dependence of the convergence factor M when we use central difference discretizations on both the global coarse and local fine grids. To this end we vary the coarse grid and fine grid mesh sizes H and h; if we fix H and vary h, we can study the influence of the refinement factor $\sigma = H/h$. In Figure 4.5 we show the dependence of the convergence rate on h and hence σ for several values of H. Figure 4.5 (a) shows the theoretical convergence rate given by [5, Theorem 2]; Figure 4.5 (b) shows the estimate M_i from (4.23) for $i = 2$. Indeed, as it was predicted, there is only a weak dependence on σ. As one can see from Figure 4.5 (b), we have a flat graph of the convergence rate as h tends to zero, which means we have no dependence of the convergence rate on σ. The values of h are decreasing from right to left, so we can say that from right to left h tends to zero (and consequently σ tends to infinity). From the theory it was predicted that we have no dependence of the convergence rate M on the fine grid mesh size h (and refinement factor σ), so from our test we want to have a flat line. This is indeed true in Figure 4.5. In Figure 4.6 we show the dependence of



(a) Theoretical result (b) Result of real computation

Figure 4.5: Dependence of the convergence rate on h. Central differences, $\kappa = 100$ (Example 1).

the convergence rate on H for several values of h. Figure 4.6 (a) shows the theoretical convergence rate given by [5, Theorem 2]; Figure 4.6 (b) shows the estimate M_i from (4.23) for $i = 2$. In Figure 4.6 (b) one can see the typical dependence of the convergence rate on the coarse grid size H. From the theory in Section 4.2 we expect our convergence rate M not to depend on the fine grid mesh size h and to depend on the coarse grid mesh size as $O(H^2)$. This is the behavior we find in Figures 4.5 and 4.6. The numerical results confirm the analytical results from Section 4.2. We should remark that these results correspond to the ones in [5, Theorem 2]. The insight given by theory about the dependence of the convergence factor M on the coarse grid mesh size H is confirmed



(a) Theoretical result.

(b) Result of real computation.

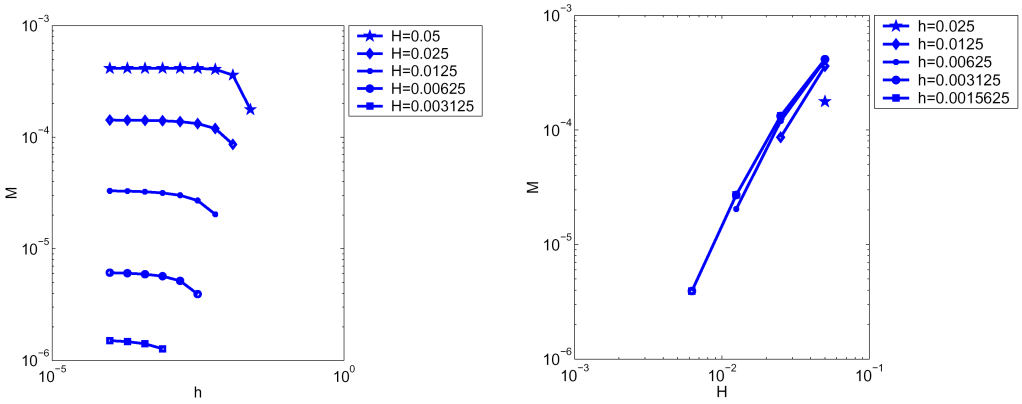
Figure 4.6: Dependence of the convergence rate on coarse grid size H. Central differences, $\kappa = 100$ (Example 1).

by the numerical experiment. We are able to correctly predict the convergence rates.

Example 2. Convection-dominated problems. Upwind scheme. In this section we use the upwind discretization on both grids and carry out the same experiments as for central differences. First we study if the convergence factor M depends on the refinement factor σ and on the coarse grid mesh size H. From the theory presented in Section 4.4 we expect to have no dependence of the convergence factor on the fine grid mesh size. As for coarse grid mesh size H we expect an $O(H)$ dependence. This dependence is indeed observed in the numerical experiments. This can be seen in Figure 4.7 in which we show the estimate M_i for the convergence rate for varying h and H. We have not provided the convergence rate predicted by theory as these figures are almost identical (just like in Figures 4.5 and 4.6 in the previous example).

Example 3. Diffusion-dominated problems. Central difference scheme. For diffusion-dominated problems (here we consider $\kappa = 0.01$) the convergence factor M is typically of order 10^{-8}. Therefore the LDC iteration reaches its fixed point in one or two steps and it is hard to estimate the convergence factor reliably (see (4.23)). It was known from previous articles (see [19–21]) that the LDC algorithm performs well for pure diffusion problems, so it is not surprising that in the case of diffusion-dominated problems (with presence of small convection) the LDC method gives fast convergence. Some results for the diffusion-dominated problem one can find in Figure 4.8.

Example 4. Pure convection problem. By means of the LDC technique we are able to solve not only convection-diffusion problems and pure diffusion problems, but also



(a) Fine grid mesh size.

(b) Coarse grid mesh size.

Figure 4.7: Dependence of the convergence rate on coarse H and fine h grid sizes. Upwind scheme, $\kappa = 100$ (Example 2).

convection problems like

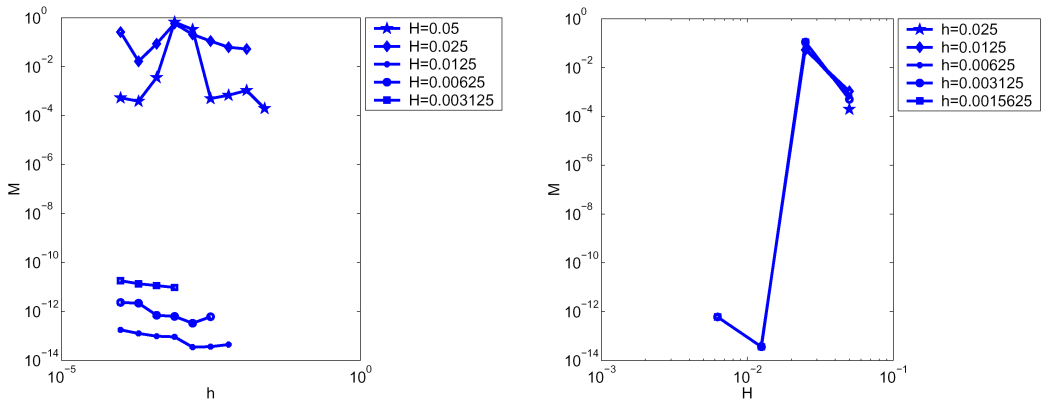
$$\begin{cases} c\frac{\partial u}{\partial x} = f, x \in \Omega = (0,1), \\ u(0) = u_0. \end{cases} \tag{4.25}$$

The idea is to transform this problem into a two-point boundary value problem and to solve the latter one by the LDC technique. In order to do this, we first extend our domain to the right and introduce a diffusive term $\epsilon\frac{\partial^2 u}{\partial x^2}$ into (4.25) as well as an extra boundary condition on the right side. Since the correct boundary value on the right is not known, we will have a boundary layer of order ϵ. So instead of problem (35) we solve the following one

$$\begin{cases} \epsilon\frac{\partial^2 u}{\partial x^2} + c\frac{\partial u}{\partial x} = f, \qquad c > 0, \qquad x \in \widetilde{\Omega} = (0, 1+\epsilon) \\ u(0) = u_0,\ u(1+\epsilon) = u_{guess}. \end{cases} \tag{4.26}$$

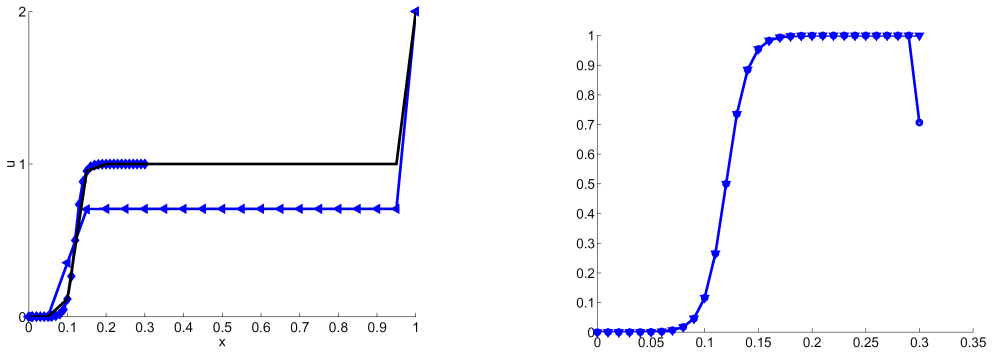
We introduce two fine grids: one to resolve the high activity area and another one to resolve the boundary layer. For this example we use the same fine grid step sizes for both fine grids, although it is not required. For the second grid we choose region $(1-\gamma+\epsilon, 1+\epsilon)$ and $u_{guess} = 2$.

The results of the numerical tests with Péclet number Pe, equal to 100000 are presented below. We used the upwind scheme on all grids. For results presented in Figure we used following parameters $\gamma = 0.3$, $\epsilon = 10^{-6}$, $c = 1$, $\sigma = 20$, $h = 0.01$. Depending on the fine grid mesh size h it is also possible to use central differences in the high activity region, even in the case when the scheme is not stable. So instead of solving a pure convection problem (which might be difficult to solve), we transform it to into a two-point boundary value problem and solve the latter one efficiently by the LDC technique.



(a) (b)

Figure 4.8: Dependence of the convergence rate on fine grid size h and coarse grid mesh size H. Central difference scheme, $\kappa = 0.1$ (Example 3).



(a) Initial solution (marked with triangles), exact solution (solid line) and second fine grid solution (marked with diamonds)

(b) Sequence of the fine grid solutions - initial fine grid solution (marked with circles) and second fine grid solution (marked with triangles)

Figure 4.9: Convection problem. Upwind scheme, $c = 1$, $\epsilon = 10^{-6}$ (Example 4).

Chapter 5

Combination of the LDC technique with high order compact finite difference schemes

As was mentioned in Chapter 1, compact finite differences are widely used for DNS calculations of turbulent flows. In this chapter we present an algorithm which allows for combining the Local Defect Correction (LDC) technique, presented in Chapter 3 and analysed in Chapter 4, with compact finite differences.

5.1 High order compact finite difference schemes

Implicit finite difference relations for the first and second derivatives have been given a variety of names. Many can be found in [15], under the names of *Mehrstellen* or *Hermitian* methods by analogy of Hermitian finite elements. They are also known as Padé differencing approximations [33]. In the 70-80's a large number of applications for solving fluid-mechanics equations have been developed [29,34,61,62]. From the more recent papers [38] and [42] should be mentioned. Following [54] all implicit formulae can be derived in a systematic way from a Taylor series expansion.

The traditional numerical methods for the DNS of transitional and turbulent flows have been spectral methods because of their high accuracies. But applications of spectral methods have been limited to flows in simple domains. Several alternative numerical methods have been used for general geometries. Examples are the spectral element methods [53], high-order compact (Padé) finite-difference methods [38], and high-order non-compact (explicit) finite-difference methods [58].

Finite-difference methods have recently received much attention for the DNS of transitional and turbulent flows [2, 38, 56, 58, 59] because they can easily be applied to complex geometries. Finite-difference schemes include both traditional explicit schemes and compact [38] schemes. High-order schemes are required because traditional second order schemes do not have an adequate accuracy for DNS. High-order finite-difference methods used most are central difference schemes [38, 56], which introduce only phase errors but no dissipative errors in numerical solutions. The drawback of central schemes is that they are not robust enough for convection-dominated hypersonic flow simulations. Extra filtering procedures, which are equivalent to adding numerical dissipation in an ad hoc manner, are needed in order to stabilize the computations and control the aliasing errors. For example, central difference schemes of fourth order or higher are unstable when they are coupled with high-order boundary schemes using one-sided finite-difference approximations [12, 56]. In [12] it is shown that for a sixth-order inner central compact scheme, only a third-order boundary scheme can be used without introducing instability. This results in a globally fourth-order accurate scheme, even though the inner scheme is sixth-order accurate.

On the other hand, upwind-bias schemes are very robust even when they are made high-order accurate, see [58], where a spatially fifth-order upwind finite-difference scheme using an upwind-bias stencil for the Navier–Stokes equations is used. The numerical dissipation in the upwind-bias schemes is enough to control the aliasing errors. In recent years, many other upwind high-order schemes have been developed; [75] discusses a fifth-order compact upwind scheme for moisture transport in atmosphere. A fourth-order compact upwind scheme is used in [13] because the standard central compact schemes break down in convection–dominated problems. The accuracy of a fifth-order explicit upwind finite-difference scheme with built-in filtering terms in a central grid stencil for linear wave propagation problems was tested in [85]. In [66] explicit numerical damping was used to stabilize high-order finite-difference equations for the Navier–Stokes equations. [1] proposes fifth-order upwind compact schemes with spectral-like resolution using central grid stencils for the direct numerical simulation of shock-turbulence interaction. [84] uses upwind compact and explicit high order finite difference schemes for calculations of hypersonic boundary layer transitions.

Finite difference schemes may be classified as explicit or implicit. Explicit schemes express the nodal derivatives as an explicit weighted sum of the nodal values of the function, e.g., $u'_i = (u_{i+1} - u_{i-1})/2h$ and $u''_i = (u_{i+1} - 2u_i + u_{i-1})/h^2$ for the first and second derivative respectively. Throughout this section u_i and u^k_i denote the values of the function and its k-th derivative respectively, at the node $x = x_i$, and h denotes the uniform mesh spacing. By comparison, implicit (compact) schemes equate a weighted sum of the nodal derivatives to a weighted sum of the function, e.g., $u'_{i-1} + 4u'_i + u'_{i+1} = 3(u_{i+1} - u_{i-1})/h$ and $u''_{i-1} + 10u''_i + u''_{i+1} = 12(u_{i+1} - 2u_i + u_{i-1})/h^2$ for the first and second derivative respectively. It is well known [15, 33, 38] that implicit schemes are significantly more accurate for smaller scales than explicit schemes with the same mesh width. This increase in accuracy is achieved at the cost of inverting a banded (usually tridiagonal) matrix to obtain the nodal derivatives. Since tridiagonal matrices can be inverted quite efficiently [68], the implicit schemes are extremely attractive when explicit time advancement schemes are used. The most popular of the implicit schemes (also

called Padé schemes due to their derivation from Padé approximants) appear to be the symmetric fourth and sixth order versions (see, e.g. [38]).

When we restrict ourselves to three-point expressions the general form of an implicit finite difference relation between a function of one variable and its first two derivatives would read

$$\begin{aligned} a_+ u_{i+1} + a_0 u_i + a_- u_{i-1} + b_+ u'_{i+1} + b_0 u'_i + b_- u'_{i-1} \\ + c_+ u''_{i+1} + c_0 u''_i + c_- u''_{i-1} = 0. \end{aligned} \tag{5.1}$$

By imposing different constraints on the coefficients a, b and c, we can tune the numerical scheme in some sense. For example we can get a higher order or a better spectral resolution of the scheme [71]. In the following we address some of the possible high order compact finite difference schemes, which will later be used in numerical examples. In Chapter 6 we present some more compact schemes actually used for calculations.

The problem we study in this chapter is given by

$$\begin{cases} Lu = -\epsilon \nabla^2 u(x,y) + \mathbf{c} \cdot \nabla u(x,y) = f(x,y) \text{ in } (0,1), \\ u = g \text{ on } \Gamma. \end{cases} \tag{5.2}$$

In (5.2), L is a linear elliptic differential operator, and f and g are the source term and Dirichlet boundary value respectively, $\epsilon > 0$ is the diffusion coefficient, $\mathbf{c}$ is the convection coefficient, u is the unknown function of (x, y), Ω is the domain of interest and Γ is the boundary of this domain.

We start with a one-dimensional situation. In this case we have three sets of equations: a discretized version of our one-dimensional convection-diffusion equation

$$\begin{cases} -\epsilon u''(x) + c u'(x) = f(x), \qquad c, \epsilon > 0 \qquad x \in (0,1), \\ u(0) = u_0, \quad u(1) = u_1, \end{cases} \tag{5.3}$$

which looks like

$$Lu_i \equiv -\epsilon u''_i + c u'_i = f_i, \qquad i = 1, \ldots, N-1; \tag{5.4}$$

an expression for the second derivative (see Section 5.1.1); an expression for the first derivative (see Section 5.1.2).

5.1.1 Diffusive term

For the discretization of the diffusive term in (5.2) we can use the following schemes

- Padé scheme (3-point scheme [54])

$$u''_{i+1} + 10 u''_i + u''_{i-1} - \frac{12}{h^2}(u_{i+1} - 2u_i + u_{i-1}) = 0. \tag{5.5}$$

To close this relation at the boundary, we use

$$\epsilon u_1'' + c u_1' = f_1, \qquad \epsilon u_N'' + c u_N' = f_N,$$

where c and ϵ come from (5.4).

- High order compact upwind scheme of Zhong (5-point scheme [84])

$$25u_{i-1}'' + 60u_i'' + 15u_{i+1}'' = \frac{1}{h}\left(-\frac{5}{2}u_{i-1}' - \frac{160}{3}u_{i-1}' + 15u_i' + 40u_{i+1}' + \frac{5}{6}u_{i+2}'\right). \tag{5.6}$$

To close this relation at the boundary, we use [84]

$$60u_0'' + 180u_1'' = \frac{1}{h}(-170u_0' + 90u_1' + 90u_2' - 10u_3'), \tag{5.7}$$

$$15u_0'' + 60u_1'' + 15u_2'' = \frac{1}{h}(-45u_0' + 45u_2'), \tag{5.8}$$

$$15u_N'' + 60u_{N-1}'' + 15u_{N-2}'' = -\frac{1}{h}(-45u_N' + 45u_{N-2}'), \tag{5.9}$$

$$60u_N'' + 180u_{N-1}'' = -\frac{1}{h}(-170u_N' + 90u_{N-1}' + 90u_{N-2}' - 10u_{N-3}'). \tag{5.10}$$

Another possibility is to use the same scheme as for the convective term (see relation (5.12) and related closing relations in Section 5.1.2), but modified for second derivative

$$u_{i+1}'' + 4u_i'' + u_{i-1}'' - \frac{3}{h}(u_{i+1}' - u_{i-1}') = 0. \tag{5.11}$$

To close this relation at the boundary, we use

$$2u_1'' + u_0'' - \frac{1}{2h}(u_2' + 4u_1' - 5u_0') = 0, \qquad u_N'' + 2u_{N-1}'' - \frac{1}{2h}(5u_N' - 4u_{N-1}' - u_{N-2}') = 0.$$

5.1.2 Convective term

Central difference type schemes do not behave well when applied to convection-dominated problems. A way to overcome this difficulty is to use upwind schemes. In [12] stability of the numerical boundary treatment for compact high-order finite difference schemes was investigated and it was shown that the upwind compact schemes give better results. A lot of information about non-centered high-order compact finite difference schemes one can find in [75].

For the discretization of the convective term in (5.2) we can use the following schemes

- Padé scheme (3-point scheme [54])

$$u'_{i+1} + 4u'_i + u'_{i-1} - \frac{3}{h}(u_{i+1} - u_{i-1}) = 0. \tag{5.12}$$

To close this relation at the boundary, we use

$$2u'_1 + u'_0 - \frac{1}{2h}(u_2 + 4u_1 - 5u_0) = 0, \qquad 2u'_N + u'_{N-1} - \frac{1}{2h}(5u_N - 4u_{N-1} - u_{N-2}) = 0.$$

- High order compact upwind scheme of Zhong (5-point scheme [84]). This is the scheme (5.6)-(5.10), modified for the first derivative:

$$25u'_{i-1} + 60u'_i + 15u'_{i+1} = \frac{1}{h}\left(-\frac{5}{2}u_{i-1} - \frac{160}{3}u_{i-1} + 15u_i + 40u_{i+1} + \frac{5}{6}u_{i+2}\right). \tag{5.13}$$

To close this relation at the boundary, we use [84]

$$60u'_0 + 180u'_1 = \frac{1}{h}(-170u_0 + 90u_1 + 90u_2 - 10u_3),$$

$$15u'_0 + 60u'_1 + 15u'_2 = \frac{1}{h}(-45u_0 + 45u_2),$$

$$15u'_N + 60u'_{N-1} + 15u'_{N-2} = -\frac{1}{h}(-45u_N + 45u_{N-2}),$$

$$60u'_N + 180u'_{N-1} = -\frac{1}{h}(-170u_N + 90u_{N-1} + 90u_{N-2} - 10u_{N-3}).$$

5.2 Combination of LDC with HOCFD

In this section we present an algorithm which combines the Local Defect Correction technique with the schemes presented in Section 5.1. First we start with the one-dimensional version of equation (5.2) and after that we extend the algorithm to two- and three-dimensional problems.

5.2.1 One-dimensional problems

We can rewrite the system of equations (5.5), (5.12) and a discrete form of the differential equation (5.4) with corresponding boundary conditions in the following matrix-vector form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.14}$$

where $\mathbf{A}$ is a 3×3 block-diagonal matrix representing the discrete operator, $\mathbf{x} = (\mathbf{u}''^{\mathsf{T}}, \mathbf{u}'^{\mathsf{T}}, \mathbf{u}^{\mathsf{T}})$ is the vector of unknowns and $\mathbf{b} = (0, 0, \mathbf{f})^{\mathsf{T}}$ is the right-hand side. The matrix $\mathbf{A}$ has the following structure:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix}. \tag{5.15}$$

The condition number of the matrix $\mathbf{A}$ is quite large, even for "small" problems, due to the fact that the matrix is highly unbalanced (we have $O(1)$, $O(h)$ and $O(h^2)$ terms on the diagonal). Moreover, we should point out that the LDC algorithm is not applicable to the equation in the form (5.14). So we need a reformulation. One of the ways to solve this problem is to rearrange the matrix $\mathbf{A}$. We can write our system (5.14) as

$$\begin{aligned} \mathbf{A}_{11}\mathbf{u}'' + \mathbf{A}_{12}\mathbf{u}' + \mathbf{A}_{13}\mathbf{u} &= \mathbf{0}, && (5.16)\\ \mathbf{A}_{22}\mathbf{u}' + \mathbf{A}_{23}\mathbf{u} &= \mathbf{0}, && (5.17)\\ \mathbf{A}_{31}\mathbf{u}'' + \mathbf{A}_{32}\mathbf{u}' + \mathbf{A}_{33}\mathbf{u} &= \mathbf{f}. && (5.18) \end{aligned}$$

Rearranging the terms we can get the following equation:

$$\mathbf{A}_{11}^{-1}\left(\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{23} - \mathbf{A}_{13}\right)\mathbf{u} - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23}\mathbf{u} + \mathbf{A}_{33}\mathbf{u} = \mathbf{f}, \text{ or,} \tag{5.19}$$

$$\mathbf{A}_m\mathbf{u} = \mathbf{f}, \tag{5.20}$$

with $\mathbf{A}_m := \mathbf{A}_{11}^{-1}\left(\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{23} - \mathbf{A}_{13}\right) - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23} + \mathbf{A}_{33}$. Matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are non-singular. After these rearrangements our matrix $\mathbf{A}_m$ does not suffer from ill conditioning. After the reformulation like (5.20) it is possible to directly apply the algorithm presented in Section 3.3.

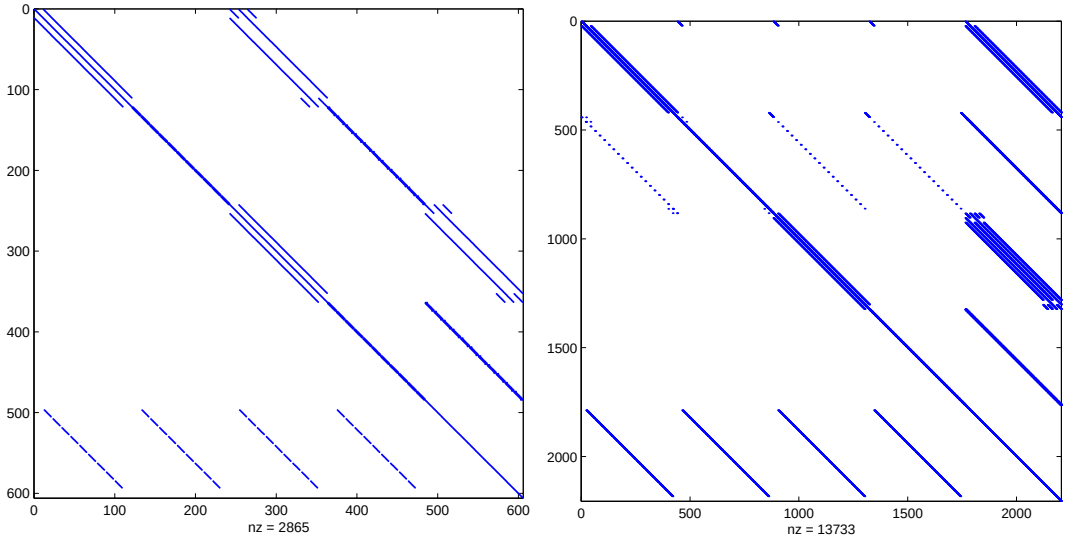
5.2.2 Two- and more dimensional problems

We would like to discretize (5.2) using the schemes from Section 5.1. We introduce the vector of unknowns $\mathbf{x} = (\mathbf{u}_{xx}^T, \mathbf{u}_{yy}^T, \mathbf{u}_x^T, \mathbf{u}_y^T, \mathbf{u}^T)^T$, with the size 5N where N is the number of grid points.

The corresponding matrix $\mathbf{A}$ is a 5×5 block matrix with the following structure

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{A}_{15} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} & \mathbf{A}_{25} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{A}_{34} & \mathbf{A}_{35} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{A}_{44} & \mathbf{A}_{45} \\ \mathbf{A}_{51} & \mathbf{A}_{52} & \mathbf{A}_{53} & \mathbf{A}_{54} & \mathbf{A}_{55} \end{pmatrix} \tag{5.21}$$

Entries $\mathbf{A}_{1,i}$ represent the discretization of u_{xx} by one of the possible discretization schemes (5.12)-(5.13), entries $\mathbf{A}_{2,i}$ of u_{yy} by (5.12)-(5.13), entries $\mathbf{A}_{3,i}$ of u_x by (5.5)-(5.6), entries $\mathbf{A}_{4,i}$ of u_y by (5.5)-(5.6), entries $\mathbf{A}_{5,i}$ of u; the latter represent the equation (5.2) as well as the boundary conditions. Depending on the type of discretization used, some of the off-diagonal submatrices $\mathbf{A}_{i,j}$ could be zero or singular (see Figure 5.1). The matrix $\mathbf{A}$ has quite a large condition number, so we use equilibration of rows in order to reduce it. The basic idea of the equilibration of rows is to multiply rows of the matrix such that we get $O(1)$ values on the main diagonal. More detailed information one can find in [78].



(a) Scheme (5.12)+(5.11), N=11 (b) Scheme (5.12)+(5.5), N = 21

Figure 5.1: Possible matrix structures depending on the scheme used

For two-dimensional problems it is quite difficult to perform explicit substitution like (5.19). We still do the same reduction of the matrix $\mathbf{A}$, but instead of explicitly expressing the matrix subblocks, we use block Gaussian elimination. In order so solve our problem we need to perform the following steps

1. Solve the coarse grid problem
   (a) Construct matrix $\mathbf{A}^H$ and right hand side $\mathbf{f}^H$. Matrix $\mathbf{A}^H$ has the form (5.15).
   (b) Perform block LU-decomposition of the matrix $\mathbf{A}^H$. As a result we have $\mathbf{A}^H = \mathbf{L}^H\mathbf{U}^H$ and we get $\mathbf{L}^H$ and $\mathbf{U}^H$ in the following form

$$\mathbf{L}^H = \begin{pmatrix} \mathbf{I} & 0 & 0 & 0 & 0 \\ \mathbf{L}_{21} & \mathbf{I} & 0 & 0 & 0 \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{I} & 0 & 0 \\ \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} & \mathbf{I} & 0 \\ \mathbf{L}_{51} & \mathbf{L}_{52} & \mathbf{L}_{53} & \mathbf{L}_{54} & \mathbf{I} \end{pmatrix},$$

$$\mathbf{U}^H = \begin{pmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \mathbf{U}_{13} & \mathbf{U}_{14} & \mathbf{U}_{15} \\ 0 & \mathbf{U}_{22} & \mathbf{U}_{23} & \mathbf{U}_{24} & \mathbf{U}_{25} \\ 0 & 0 & \mathbf{U}_{33} & \mathbf{U}_{34} & \mathbf{U}_{35} \\ 0 & 0 & 0 & \mathbf{U}_{44} & \mathbf{U}_{45} \\ 0 & 0 & 0 & 0 & \mathbf{U}_{55} \end{pmatrix}.$$

   (c) Define the vector $\mathbf{y}^H := \mathbf{U}^H\mathbf{x}$.
   (d) Solve $\mathbf{L}^H\mathbf{y}^H = \mathbf{f}^H$. Get $\mathbf{y}_5^H$.

(e) Solve

$$\mathbf{U}_{55}\mathbf{u}^H = \mathbf{y}_5^H \tag{5.22}$$

and get $\mathbf{u}^H$, the coarse grid solution.

2. Solve the fine grid problem

   (a) Get the boundary conditions for the fine grid boundary value problem and construct $\mathbf{A}_l^h$ and $\mathbf{f}_l^h$.

   (b) Solve the local grid problem

$$\mathbf{A}_l^h\mathbf{x}_l^h = \mathbf{f}_l^h \tag{5.23}$$

   and get $\mathbf{x}_l^h = ((\mathbf{u}_{xx}^h)^T, (\mathbf{u}_{yy}^h)^T, (\mathbf{u}_x^h)^T, (\mathbf{u}_y^h)^T, (\mathbf{u}^h)^T)$. Extract $\mathbf{u}_l^h$.

3. Calculate the defect

   (a) Construct the vector $\mathbf{w}^H$

$$\mathbf{w}^H(x,y) = \begin{cases} \mathbf{u}_l^h(x,y) & (x,y) \in \Omega_l^H \\ \mathbf{u}^H(x,y) & (x,y) \in \Omega^H \setminus \Omega_l^H \end{cases}$$

   (b) Construct the defect $\mathbf{d}_0^H = \mathbf{U}_{55}\mathbf{w}^H - \mathbf{y}_5^H$. Restrict the defect by setting it to zero outside the area of refinement.

4. Solve the updated coarse grid problem

$$\mathbf{U}_{55}\mathbf{u}_1^H = \mathbf{y}_5^H + \mathbf{d}_0^H \tag{5.24}$$

   and get the new coarse grid solution $\mathbf{u}_1^H$.

5. If not converged, go to step 2.

We have outlined all steps of the new LDC method. The iteration is as in Algorithm 1; the basic coarse grid problem, the local fine grid problem and the updated coarse grid problem are given by (5.22), (5.23) and (5.24), respectively.

5.3 Numerical results

In this section we present some typical numerical results for the convection-diffusion problem (5.2). As in the previous section, we start with a one-dimensional problem and then present results for a two-dimensional model problem.

Example 1

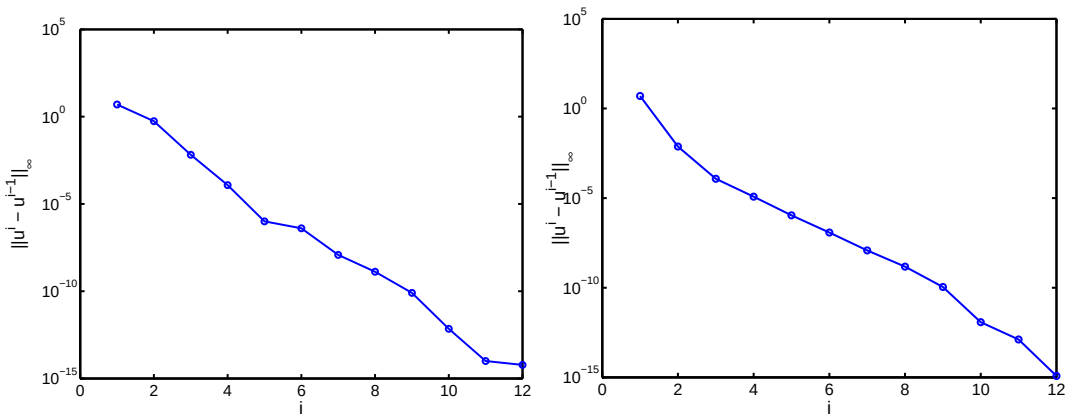
With the use of numerical simulations we want to investigate the following properties of our LDC method for high order compact finite difference schemes: convergence behavior, accuracy and efficiency.

For our numerical test we solve the one-dimensional boundary value problem (5.3) and we choose the source term f and the boundary conditions such that

$$u(x) = \frac{1}{2}\left(\tanh(50(x - 1/8)) + 1\right). \tag{5.25}$$

The following parameters are chosen for the computations: $c = \mp 0.99$ (convection dominated), $c = \mp 0.1$ (diffusion dominated); in case of one-dimensional problem, our border between coarse and fine grids Γ consists of one point only, which is called the interface point γ and for this example was chosen to be $\gamma = 0.3$; different number of coarse grid points N. We use the Padé scheme given by (5.12), (5.5).

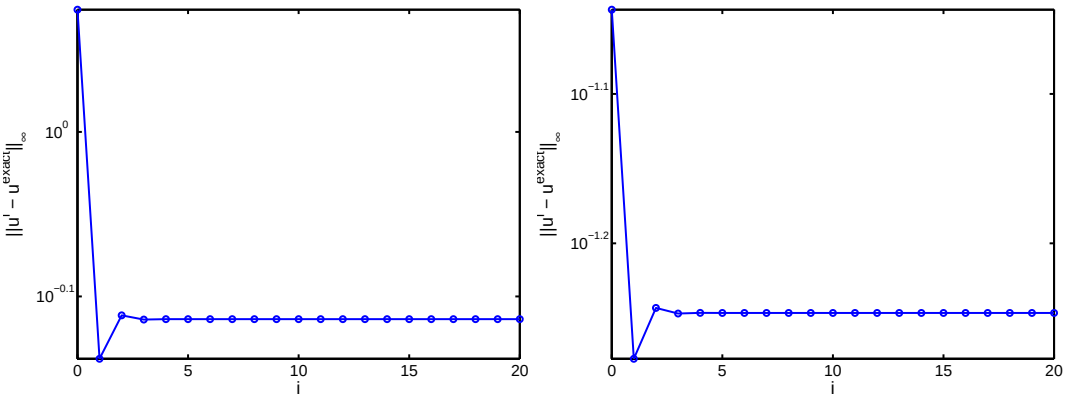
The typical results for the convergence behavior for the LDC technique one can find in Figures 5.2-5.3. In Figure 5.2 we plot $\|\mathbf{u}^i - \mathbf{u}^{i-1}\|_\infty$ against the iteration number i, where $\mathbf{u}^i$ is our numerical coarse grid solution on the ith iteration. As one can see in Figure 5.2, the LDC algorithm shows fast convergence. In Figure 5.3 we plot $\|\mathbf{u}^i - \mathbf{u}^{exact}\|_\infty$ against the iteration number, where $exact$ stands for the exact solution of the problem (which is known in our case). As one can see from Figure 5.3, we only need a small number of LDC iterations to reach the fixed point solution, and, as will be shown later, this is the solution we get on a fine uniform grid. These results are typical and do not change much for different values of N and c.



(a) $c = 0.9$, $N = 11$ (b) $c = 0.9$, $N = 21$

Figure 5.2: Convergence of the LDC algorithm

In order to compare the results of the LDC technique with those we get using a fine uniform grid, we performed a number of calculations. By fine uniform grid we mean a



(a) $c = 0.9,\ N = 11$ (b) $c = 0.9,\ N = 21$

Figure 5.3: Convergence of the LDC algorithm

c	N	LDC	fine uniform grid	c	N	LDC	fine uniform grid
0.0	11	0.067	0.056	0.5	11	0.071	0.052
	21	$2.52 * 10^{-4}$	$2.55 * 10^{-4}$		21	$2.21 * 10^{-4}$	$2.49 * 10^{-4}$
	41	$9.92 * 10^{-6}$	$9.92 * 10^{-6}$		41	$1.02 * 10^{-5}$	$1.02 * 10^{-5}$
	81	$7.41 * 10^{-6}$	$7.41 * 10^{-6}$		81	$7.57 * 10^{-6}$	$7.57 * 10^{-6}$
	161	$3.72 * 10^{-6}$	$3.72 * 10^{-6}$		161	$3.74 * 10^{-6}$	$3.74 * 10^{-6}$
0.1	11	0.067	0.056	0.9	11	0.050	0.041
	21	$2.48 * 10^{-4}$	$2.54 * 10^{-4}$		21	$3.13 * 10^{-4}$	$1.96 * 10^{-4}$
	41	$9.95 * 10^{-6}$	$9.95 * 10^{-6}$		41	$1.12 * 10^{-5}$	$1.12 * 10^{-5}$
	81	$7.43 * 10^{-6}$	$7.43 * 10^{-6}$		81	$7.83 * 10^{-6}$	$7.83 * 10^{-6}$
	161	$3.73 * 10^{-6}$	$3.73 * 10^{-6}$		161	$4.01 * 10^{-6}$	$4.01 * 10^{-6}$

Table 5.1: $\|\mathbf{u}^{exact} - \mathbf{u}^H\|_\infty$ for LDC algorithm and equivalent uniform grid

grid that covers the whole domain with a mesh size equal to the mesh size of the fine LDC grid. First we compare the errors. In Table 5.1 one can find estimation $\|\mathbf{u}^{exact} - \mathbf{u}^H\|_\infty$ for the LDC solution and for the fine uniform grid solution. For LDC we stop the iteration if $\|\mathbf{u}_i^{H,h} - \mathbf{u}_{i-1}^{H,h}\|_\infty < 1 * 10^{-5}$. The errors for the LDC algorithm and the fine uniform grid are of the same order.

We measure the CPU time spent on the computation by the LDC algorithm with refinement factor 4 and an equivalent uniform grid solution with spatial step $H/4$. It should be noted that the measurements of the CPU times cannot be considered as absolute, since they are machine-dependent; even on the same computer they could differ depending on the load of the machine at that particular moment. However, the results in Table 5.2 were obtained on the same machine and conditions (that is system activity, background processes, etc.) were approximately the same. Moreover the data in Table 5.2 is the averaged data after 25 runs of the same test. As one can see from Table 5.2, the LDC algorithm, even for one-dimensional problems, gives considerable savings in calculation time compared with an equivalent uniform grid. Taking into account that the accuracy

c	N	LDC (# iter)	equiv. uniform grid	c	N	LDC (# iter)	equiv. uniform grid
0.0	11	0.067(5)	0.026	0.5	11	0.0457(5)	0.0248
	21	0.085(4)	0.0737		21	0.0873(4)	0.0825
	41	0.209(3)	0.3461		41	0.2271(3)	0.3786
	81	0.574(2)	1.9890		81	0.6157(2)	1.9525
	161	1.8668(1)	7.6213		161	1.73(1)	11.6560
0.1	11	0.0465(5)	0.0272	0.9	11	0.0397(5)	0.0273
	21	0.0893(4)	0.0805		21	0.0725(4)	0.0785
	41	0.2299(3)	0.3818		41	0.2264(3)	0.3778
	81	0.7467(2)	2.0971		81	0.6222(2)	2.1028
	161	1.8111(1)	11.2755		161	1.7290(1)	11.5411

Table 5.2: Calculation time for LDC algorithm and equivalent uniform grid

of both methods is the same, we can conclude that for problems with some high activity regions and smooth solutions in the rest of the domain even in one-dimension, LDC is the method to use.

Example 2

Now we consider the LDC algorithm combined with the high order finite difference schemes in the algorithm described in Section 5.2.2. For the numerical experiment, we apply the LDC algorithm to the boundary value problem

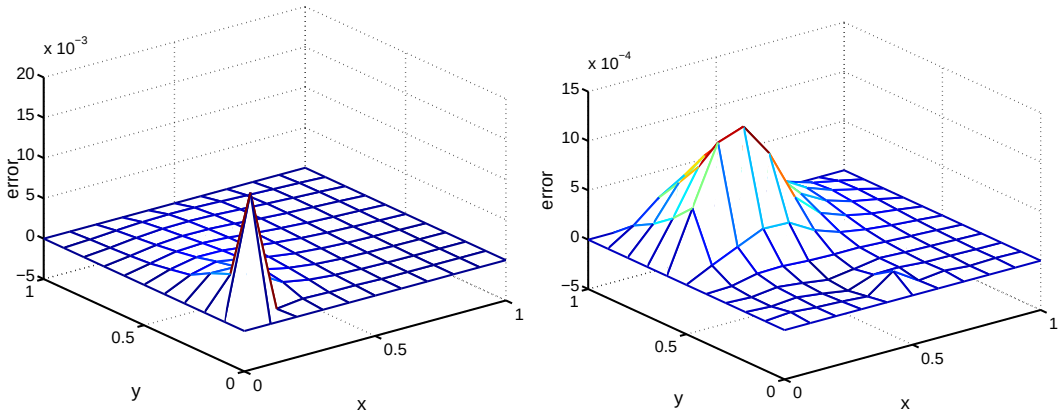
$$\begin{cases} -\epsilon_1 \dfrac{\partial^2 u}{\partial x^2} - \epsilon_2 \dfrac{\partial^2 u}{\partial y^2} + c_1 \dfrac{\partial u}{\partial x} + c_2 \dfrac{\partial u}{\partial y} = f, \qquad (x, y) \in \Omega = (0,1) \times (0,1), \\ u(x,0) = g_1(x),\ u(x,1) = g_2(x),\ u(0,y) = g_3(y),\ u(1,y) = g_4(y). \end{cases} \tag{5.26}$$

In (5.26), f and g_i have been chosen such that

$$u(x,y) = \tanh\left[25(x + y - 1/3)\right] + 1. \tag{5.27}$$

We choose a uniform coarse grid Ω^H in Ω with grid sizes $\Delta x = \Delta y = 1/(N-1)$, with $N = 11, 16, 21$. The area of refinement Ω_l is chosen as $\Omega_l = (0, 1/2)^2$. We choose a uniform fine grid Ω_l^h in Ω_l with grid sizes $\Delta x = \Delta y = h$ with $h = H/2, H/4$. For the computation presented below we choose $\epsilon_1 = \epsilon_2 = 0.1$ and $c_1 = c_2 = 1$. For the coarse grid discretization we present results for the Padé scheme (5.5) for the diffusive term and scheme of Zhong (5.13) for the convective term. For the fine grid we used the same schemes as for the coarse grid, although it is not required.

The typical results for the LDC algorithm can be found in Figure 5.4. All the numerical tests show fast convergence of the LDC algorithm. Usually the LDC algorithm reaches the fixed point solution in one or two iterations. We also compare the accuracy of results for the LDC solution and for the fine uniform grid - the accuracy is similar, although the LDC gives a substantial savings in the size of the problem. For instance, suppose we have a 21×21 uniform grid. This leads us to a linear system with a matrix of dimension



(a) First coarse grid error

(b) Coarse grid error after 1 iteration

Figure 5.4: Difference between exact and numerical results for LDC technique applied to 2D convection-diffusion equation.

Grid size	init	1 iteration	uniform grid
Coarse: 11×11, fine: 11×11, equiv.: 21×21	$2.56 * 10^{-2}$	$1.30 * 10^{-3}$	$1.26 * 10^{-3}$
Coarse: 11×11, fine: 21×21, equiv.: 41×41	$2.56 * 10^{-2}$	$1.28 * 10^{-3}$	$3.46 * 10^{-5}$
Coarse: 21×21, fine: 21×21, equiv.: 41×41	$1.26 * 10^{-3}$	$3.23 * 10^{-5}$	$3.46 * 10^{-5}$

Table 5.3: $\|\mathbf{u}^{exact} - \mathbf{u}^{H}\|_{\infty}$ for LDC algorithm and equivalent uniform grid

2205×2205. If we use LDC, we can get the same accuracy by using two 11×11 grids - coarse and fine. This leads to two linear systems with matrices of dimension 605×605.

We compare accuracy and performance of the LDC method with the solution on a fine uniform grid. The results are in Tables 5.3 and 5.4. The accuracy of the LDC method is exactly the same as for the equivalent uniform grid. In Table 5.3 the results for the LDC technique with an 11×11 coarse grid and a 21×21 fine grid do not differ from those for 11×11 for both coarse and fine grids due to the fact that the main error region is no longer in the area of refinement (see Figure 5.4).

We expect that the LDC algorithm should be more efficient than the uniform fine grid method. This is indeed what we see in Table 5.4 - LDC is much faster than the equivalent uniform grid method.

Grid size	1 iteration	equiv. uniform grid
Coarse: 11×11, fine: 11×11, equiv.: 21×21	2.60	7.71
Coarse: 11×11, fine: 21×21, equiv.: 41×41	1.84	530
Coarse: 21×21, fine: 21×21, equiv.: 41×41	9.988	530

Table 5.4: Calculation time for LDC algorithm and equivalent uniform grid

Chapter 6

Boundary conditions for turbulent flows

6.1 Introduction

Numerical simulations necessitate a truncation of the computational domain by artificial boundaries. The artificial boundaries for the rectangular computational domain used for the present application are the inflow, outflow, lower and upper boundaries as sketched in Figure 6.1. Since these boundaries are close to the plate, the flow outside the computational domain is different from an undisturbed, uniform flow; it is affected by disturbances inside the domain. Moreover, the presence of the cooling jet makes the flow picture even more complicated. On the other hand, in order to calculate the interior flow, we need some external flow information. This mutual requirement of information makes the conditions along the artificial boundaries unknown in most flow applications and therefore should be approximated. Mathematically the boundary conditions of a system of equations are subject to a certain requirement in order for the problem to be well-posed. The number of physical boundary conditions for the *well-posedness* requirement for the Euler and Navier-Stokes equations is stated in Table 6.1 for two- and three-dimensional cases [69]. Depending on the type of flow and the dimension of the problem, a different number of "physical" boundary conditions is required, so in the first column of Table 6.1 we list different types of inflow and outflow situations.

The boundary conditions listed in Table 6.1 are provided by some information about the external flow adjacent to the boundaries. We call these boundary conditions physical boundary conditions. In some cases, however, no accurate external flow information is available, such as at the outflow boundary. Although mathematically only a certain number of boundary conditions is required, depending on the local flow condition, numerically, we should specify all the dependent variables. We call the conditions when

Figure 6.1: Sketch of the computational domain

Dimension	Two-dimensional		Three-dimensional	
Flow type	Euler	Navier-Stokes	Euler	Navier-Stokes
Supersonic inflow	4	4	5	5
Subsonic inflow	3	4	4	5
Supersonic outflow	0	3	0	4
Subsonic outflow	1	3	1	4

Table 6.1: Number of physical boundary conditions required for well-posedness of two- and three-dimensional flows.

completing the specification of the dependent variables the numerical boundary conditions. From numerical calculations, we observe [55] that fulfilling the number of physical boundary conditions according to the theory for a hyperbolic system (Euler equations) is sufficient. Satisfying the number of conditions for the Navier-Stokes equations by adding some viscous conditions (see [55]) results only in small differences. Therefore we follow the inviscid approach. The equivalence of the results can be understood from the fact that in the present application the viscous effects are restricted to the region near the wall.

Extrapolation method

Along an artificial boundary, some dependent variables are unspecified by the physical boundary conditions. To complete the specification of the variables, we simply extrapolate dependent variables from interior points to the boundary. The choice of dependent variables for the physical and numerical boundary conditions is not unique. The following are appropriate combinations based on our experience for two-dimensional problems.

First we consider a subsonic inflow boundary. According to Table 6.1, we should prescribe three physical boundary conditions. In the case of an isothermal wall, the appropriate physical boundary conditions at the inflow boundary are provided by the (two) velocity components and the temperature. The fourth variable can then be specified by extrapolating the pressure or the density from the interior points, which represents the numerical boundary condition. In the case of a subsonic outflow boundary, we need one physical and three numerical boundary conditions. The physical boundary condition can be satisfied by prescribing one dependent variable, for instance pressure, although this will cause reflections of existing disturbance waves at the outflow boundary. The numerical conditions are satisfied by extrapolating the velocity components and the temperature from the interior points.

We consider now a subsonic upper boundary. Depending on the sign of the velocity normal to the boundary, we regard it as an inflow upper boundary (negative sign) or an outflow upper boundary (positive sign). From this point, we can proceed in the same way as in the previous cases. The pressure can be used as a physical boundary condition, for instance by prescribing a pressure jump in the case we want to invoke a steady shock from the upper boundary. The procedure for a supersonic boundary follows accordingly. We prescribe all the dependent variables at the inflow boundary (no numerical boundary conditions) and extrapolate them from the interior points at the outflow boundary (no physical boundary conditions).

Buffer/sponger zone method

As an alternative for the extrapolation method, a buffer zone method was developed. The idea is to add to the original computational domain a special ”buffer” zone at the

outflow boundary. The purpose of this zone is to avoid reflections from the outflow boundary. The governing equations in the buffer domain are modified by increasing diffusion in a certain direction. Alternatively, a sponge layer approach has been developed and used in [71]. The sponger layer is put outside the outflow or far-field boundary, where a damping function is used to depress the flow fluctuations. Both buffer or sponge zones are capable of absorbing the outward moving waves and have been used in many LES/DNS computations.

But these methods have some shortages. First, extra sponge or buffer areas have to be added to the original domain, which will increase the number of grid points and computational cost. Second, the sponge approach can only be applied when the equations of perturbation are considered, which may not be applicable when a so-called ”base flow” does not exist or cannot be defined. With ”base flow” we mean presence of some basic flow stability, and therefore ability to describe the physical components by their mean value and fluctuating part. Consider as an example turbulent duct flow - there is a vivid directional anisotropy, since the streamwise components are dominant. We can consider these streamwise components as ”base” flow and try to split flow variables into the mean components (those present in the ”base” flow) and a fluctuating part.

Characteristic method

The *characteristic method* described here was originally proposed by Hedstrom [27] for one-dimensional cases. Thompson [74] performed an extension to multi-dimensional problems and non-rectangular coordinate systems. The advantage of the characteristic method is that it can provide numerical as well as alternative physical boundary conditions in case the external flow is unknown. In the latter case, the so-called non-reflecting boundary condition replaces the required physical boundary condition. This boundary condition is extracted by invoking the non-reflecting principle (Hedstrom) in the characteristic form of the Euler equations. Some authors combine the extrapolation technique and the non-reflecting property of the one-dimensional characteristic method for their boundary conditions, for example [63]. Others use a quasi multi-dimensional characteristic method [55, 74]. In our work we use the method of Poinsot and Lele [55], which is briefly described below. This approach was extended to curvilinear coordinates in [62].

It is well known that hyperbolic systems of equations represent the propagation of waves [74]. At the boundary, some waves are propagating into and others out of the computational domain. The behavior of the outward propagating waves is defined entirely by the solution within the domain. In contrast, the incoming waves bring the information from the outside of the computational domain and therefore require boundary conditions to complete specification of their behavior. Although the Navier–Stokes equations are not hyperbolic [74], they do propagate waves like systems described by Euler equations do. In order to estimate these waves it seems natural to deal only with the hyperbolic part of the Navier-Stokes equations. These quantities combined with viscous stresses and dissipation are then used to specify the boundary values of all variables not prescribed by the physical boundary conditions.

Since we shall use characteristic boundary conditions for our calculations, we present a short introductional example and the basic idea for the simple one-dimensional inviscid case. In the next section this is followed by a more detailed description for viscous compressible flow.

We start with the one-dimensional Euler equations, which form a hyperbolic system. In conservative form it can be written as

$$\frac{\partial \mathbf{g}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} = 0, \tag{6.1}$$

with

$$\mathbf{g} = \begin{pmatrix} \rho \\ \rho u \\ e \end{pmatrix}, \ \mathbf{f} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (e+p)u \end{pmatrix}, \tag{6.2}$$

where u, ρ, p, e are velocity, density, pressure and total energy, respectively. Equation (6.1) describes the conservation properties of the system, that is, it relates the rate of change of the integral of a field over a small volume to the flux of that field across the volume boundaries.

We can rewrite (6.1) in so-called "primitive" form. We do so because it is easier to obtain the eigenvalues of the system when these are written in non-conservative form as a function of the primitive variables [28]. In primitive form the system (6.1) becomes

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{u}}{\partial x} = 0, \tag{6.3}$$

with

$$\mathbf{u} = \begin{pmatrix} \rho \\ u \\ s \end{pmatrix}, \ \mathbf{A} = \begin{pmatrix} u & \rho & 0 \\ c^2/\rho & u & p/(\rho s) \\ 0 & 0 & u \end{pmatrix}, \tag{6.4}$$

where $s = \frac{p}{\rho^\gamma}$, γ is ratio of specific heat at constant pressure to specific heat at constant volume (for air $\gamma = 1.4$) and $c = \sqrt{\gamma p/\rho}$. The matrix $\mathbf{A}$ has the following real eigenvalues $\lambda_1 = u - c$, $\lambda_2 = u$, $\lambda_3 = u + c$. We can diagonalize $\mathbf{A}$ using the transformation

$$\mathbf{L}\mathbf{A}\mathbf{L}^{-1} = \Lambda,$$

where Λ is a diagonal matrix $\Lambda_{ii} = \lambda_i$, $i = 1, 2, 3$ and $\mathbf{L}$ is a matrix whose rows are the left eigenvectors of the matrix $\mathbf{A}$ (under left eigenvectors we mean a row vector $\mathbf{l}_l$ satisfying $\mathbf{l}_l\mathbf{A} = \lambda_l\mathbf{l}_l$),

$$\mathbf{L} = \begin{pmatrix} \mathbf{l}_1^T \\ \mathbf{l}_2^T \\ \mathbf{l}_3^T \end{pmatrix} = \begin{pmatrix} -c & \rho & -\frac{p}{sc} \\ 0 & 0 & 1 \\ c & \rho & \frac{p}{sc} \end{pmatrix}. \tag{6.5}$$

Multiplying equation (6.3) by L gives

$$\mathbf{L}\frac{\partial \mathbf{u}}{\partial t} + \Lambda\mathbf{L}\frac{\partial \mathbf{u}}{\partial x} = \mathbf{0}, \tag{6.6}$$

or

$$\begin{pmatrix} -c & \rho & -\frac{p}{sc} \\ 0 & 0 & 1 \\ c & \rho & \frac{p}{sc} \end{pmatrix} \frac{\partial}{\partial t} \begin{pmatrix} \rho \\ u \\ s \end{pmatrix} + \begin{pmatrix} u-c & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & u+c \end{pmatrix} \begin{pmatrix} -c & \rho & -\frac{p}{sc} \\ 0 & 0 & 1 \\ c & \rho & \frac{p}{sc} \end{pmatrix} \frac{\partial}{\partial x} \begin{pmatrix} \rho \\ u \\ s \end{pmatrix} = 0. \tag{6.7}$$

We can define a new function $\mathbf{w}$ by

$$dw_i = \mathbf{l}_i^T d\mathbf{u}, \tag{6.8}$$

or

$$\frac{\partial \mathbf{w}}{\partial x} = \mathbf{L}\frac{\partial \mathbf{u}}{\partial x} = \begin{pmatrix} \mathbf{l}_1^T \\ \mathbf{l}_2^T \\ \mathbf{l}_3^T \end{pmatrix} \frac{\partial \mathbf{u}}{\partial x} = \begin{pmatrix} \mathbf{l}_1^T \frac{\partial \mathbf{u}}{\partial x} \\ \mathbf{l}_2^T \frac{\partial \mathbf{u}}{\partial x} \\ \mathbf{l}_3^T \frac{\partial \mathbf{u}}{\partial x} \end{pmatrix} \tag{6.9}$$

and rewrite equation (6.3) as follows

$$\frac{\partial \mathbf{w}}{\partial t} + \Lambda \frac{\partial \mathbf{w}}{\partial x} = \mathbf{0}, \tag{6.10}$$

or

$$\frac{\partial w_i}{\partial t} + \lambda_i \frac{\partial w_i}{\partial x} = 0, \tag{6.11}$$

which gives us a set of wave equations with characteristic velocities λ_i. Each wave amplitude w_i is constant along the curve in the xt-plane, defined by $dx/dt = \lambda_i$. The next step is to use those equations to develop boundary conditions. Now we have to make some assumption about the incoming waves. Using the so-called "non-reflecting" approach, proposed in [27] and further developed by [74], we demand that the amplitudes of the incoming waves at the boundaries (in case of a one-dimensional problem those boundaries are just points a, b) are constant in time, which leads to the relation

$$\frac{\partial w_i}{\partial t}|_{x=a,b} = L_i|_{x=a,b} = \left(\mathbf{l}_i^T \frac{\partial \mathbf{u}}{\partial t}\right)|_{x=a,b} = 0, \tag{6.12}$$

for those waves whose characteristic velocities are directed inward at the boundary. Equations (6.12) for the incoming waves and (6.7) for the outgoing waves completely determine the solution at the boundaries.

We can write a general expression for the equations at the boundary

$$\mathbf{l}_i^T \frac{\partial \mathbf{u}}{\partial t} + L_i = 0, \tag{6.13}$$

where

$$L_i = \begin{cases} \lambda_i \mathbf{l}_i^T \frac{\partial \mathbf{u}}{\partial x}, & \text{for outgoing waves} \\ 0 & \text{for incoming waves.} \end{cases} \tag{6.14}$$

In the next section we extend this approach to a more complicated case, that is, a three-dimensional viscous compressible flow. But the approach stays the same: we determine characteristic waves that enter the computational domain and then, with the use of certain assumptions (for instance it can be something like we just used - that the amplitudes of the incoming waves at the boundaries are constant in time) we try to estimate those amplitudes.

6.2 Boundary conditions based on the characteristic method

The fluid dynamics equations (for viscous compressible fluid), in Cartesian coordinates, are

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u} = 0, \tag{6.15}$$

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}, \tag{6.16}$$

$$\rho \frac{De}{Dt} = -p \nabla \cdot \mathbf{u} + \nabla \cdot k \nabla T + \Phi, \tag{6.17}$$

$$p = \rho R T, \text{ and } e = c_v T, \tag{6.18}$$

For simplicity we introduce the following notation in this section: index i stands for the direction and in the three-dimensional case $i = 1, 2, 3$, so $x_1 = x$, $x_2 = y$, $x_3 = z$ and $u_1 = u$, $u_2 = v$, $u_3 = w$. Let us now consider a boundary, located at $x_1 = X$ (Figure 6.2). The major idea of the things presented below is to identify waves crossing the boundary, and their normal and tangential components, and then use some specific technique presented below for estimating the incoming waves. So first we rewrite our system of equations (6.15)-(6.18) to find out which waves are crossing boundaries, and in which direction. Using the characteristic analysis [74] to modify the hyperbolic terms of equations (6.15)-(6.17) corresponding to the waves propagating in the x_i direction, we can recast this system as

$$\frac{\partial \rho}{\partial t} + d_1 + \frac{\partial}{\partial x_2}(m_2) + \frac{\partial}{\partial x_3}(m_3) = 0, \tag{6.19}$$

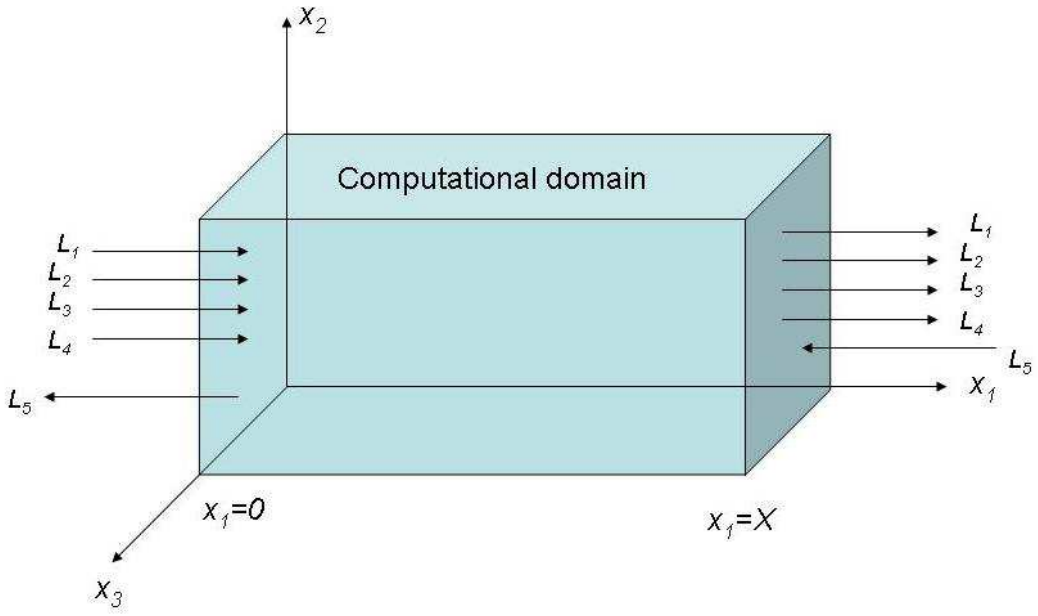


Figure 6.2: Waves leaving and entering the computational domain through an inlet plane ($x_1 = 0$) and an outlet plane ($x_1 = X$) for a subsonic flow.

$$\frac{\partial \rho E}{\partial t} + \frac{1}{2} d_1 + \frac{d_2}{\gamma - 1} + m_1 d_3 + m_2 d_4 + m_3 d_5 + \frac{\partial}{\partial x_2}\left[(\rho E + p) u_2\right] + \frac{\partial}{\partial x_3}\left[(\rho E + p) u_3\right] = \sum_{i=1}^{3}\sum_{j=1}^{3} \frac{\partial}{\partial x_i}(u_j \tau_{ij}) - \sum_{i=1}^{3} \frac{\partial q_i}{\partial x_i}, \tag{6.20}$$

$$\frac{\partial m_1}{\partial t} + u_1 d_1 + \rho d_3 + \frac{\partial}{\partial x_2}(m_1 u_2) + \frac{\partial}{\partial x_3}(m_1 u_3) = \sum_{j=1}^{3} \frac{\partial \tau_{1j}}{\partial x_j}, \tag{6.21}$$

$$\frac{\partial m_2}{\partial t} + u_2 d_1 + \rho d_4 + \frac{\partial}{\partial x_2}(m_2 u_2) + \frac{\partial}{\partial x_3}(m_2 u_3) = \sum_{j=1}^{3} \frac{\partial \tau_{2j}}{\partial x_j}, \tag{6.22}$$

$$\frac{\partial m_3}{\partial t} + u_3 d_1 + \rho d_5 + \frac{\partial}{\partial x_2}(m_3 u_2) + \frac{\partial}{\partial x_3}(m_3 u_3) = \sum_{j=1}^{3} \frac{\partial \tau_{3j}}{\partial x_j}, \tag{6.23}$$

where $m_i = \rho u_i$ is the x_i-direction momentum density (flow rate).

The various terms in the system (6.19) - (6.23) contain derivatives normal to the x_1 boundary (d_1 to d_6), derivatives parallel to the x_1 boundary like $(\partial/\partial x_2)(m_2 u_2)$ and local viscous terms. The vector **d**, which is written in terms of amplitudes of waves, can be expressed as

$$\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} = \begin{pmatrix} \frac{1}{c^2}\left[L_2 + \frac{1}{2}(L_5 + L_1)\right] \\ \frac{1}{2}(L_5 + L_1) \\ \frac{1}{2\rho c}(L_5 - L_1) \\ L_3 \\ L_4 \end{pmatrix} = \begin{pmatrix} \frac{\partial m_1}{\partial x_1} \\ \frac{\partial (c^2 m_1)}{\partial x_1} + (1-\gamma)\mu\frac{\partial p}{\partial x_1} \\ u_1 \frac{\partial u_1}{\partial x_1} + \frac{1}{\rho}\frac{\partial p}{\partial x_1} \\ u_1 \frac{\partial u_2}{\partial x_1} \\ u_1 \frac{\partial u_3}{\partial x_1} \end{pmatrix}, \quad (6.24)$$

where the L_i's are the amplitudes of waves associated with each characteristic velocity λ_i. In the following we call those waves "characteristic waves". These velocities are given by

$$\lambda_1 = u_1 - c, \quad (6.25)$$

$$\lambda_2 = \lambda_3 = \lambda_4 = u_1, \quad (6.26)$$

$$\lambda_5 = u_1 + c, \quad (6.27)$$

where c is the speed of sound

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

The L_i's are given by

$$L_1 = \lambda_1 \left(\frac{\partial p}{\partial x_1} - \rho c \frac{\partial u_1}{\partial x_1} \right), \quad (6.28)$$

$$L_2 = \lambda_2 \left(c^2 \frac{\partial \rho}{\partial x_1} - \frac{\partial p}{\partial x_1} \right), \quad (6.29)$$

$$L_3 = \lambda_3 \frac{\partial u_2}{\partial x_1}, \quad (6.30)$$

$$L_4 = \lambda_4 \frac{\partial u_3}{\partial x_1}, \quad (6.31)$$

$$L_5 = \lambda_1 \left(\frac{\partial p}{\partial x_1} + \rho c \frac{\partial u_1}{\partial x_1} \right). \quad (6.32)$$

A simple physical interpretation of the L_i's can be given by looking, for example, at the linearized equations for one-dimensional inviscid acoustic waves. Let us consider the

upstream-propagating wave associated with the velocity $\lambda_1 = u - c$. If p' and u' are the pressure and velocity perturbations, the wave amplitude $A = p' - \rho c u'$ is conserved along the characteristic line $x + \lambda_1 t = c$, where c is a constant, so that

$$\frac{\partial A}{\partial t} + \lambda_1 \frac{\partial A}{\partial x} = 0 \text{ or } \frac{\partial A}{\partial t} + L_1 = 0.$$

At a given location $-L_1$ represents the time variation of the wave amplitude A. By analogy we will call L_i's the amplitude variations of the characteristic waves crossing the boundary.

We see that system of equations (6.19) - (6.23) can be used to give values of variables on the boundary at the following time step if we can estimate the amplitude variation L_i of the waves propagating into the domain.

We have now to distinguish two different types of problem: (1) those where some information is known about the outside domain so that L_i's of the incoming waves can be determined and (2) those where such information is not available:

1. For local refinement if we consider the fine grid problem, we have all the needed information concerning the incoming and outgoing waves from the corresponding coarse grid solution. So the first approach is just to impose those quantities for the waves which enter the domain as Dirichlet boundary conditions.

2. In most cases no information about the amplitude of the incoming waves is available and the approach described in [55] can be used. This is based on inferry values of the wave amplitude variations in the viscous multidimensional case by examining a local associated one-dimensional inviscid problem. This approach is described below.

6.2.1 The local one-dimensional inviscid (LODI) relations

At each point on the boundary we consider the system of equations (6.19) - (6.23) and neglect transverse and viscous terms. The resulting equations allow us to infer values for wave amplitude variations by considering the flow locally as inviscid and one-dimensional. The relations obtained by this method are not "physical" conditions but should be viewed as compatibility relations between choices made for the physical boundary conditions and the amplitudes of waves crossing the boundary.

One of forms for the LODI system is [74]

$$\frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left[L_2 + \frac{1}{2}(L_5 + L_1)\right] = 0 \tag{6.33}$$

$$\frac{\partial p}{\partial t} + \frac{1}{2}(L_5 + L_1) = 0 \tag{6.34}$$

$$\frac{\partial u_1}{\partial t} + \frac{1}{2\rho c}(L_5 - L_1) = 0 \tag{6.35}$$

$$\frac{\partial u_2}{\partial t} + L_3 = 0 \tag{6.36}$$

$$\frac{\partial u_3}{\partial t} + L_4 = 0 \tag{6.37}$$

These equations may be combined to express the time derivatives of all other quantities of interest. For example we can express time derivative of the temperature T as follows [74]

$$\frac{\partial T}{\partial t} + \frac{T}{\rho c^2}\left[-L_2 + \frac{1}{2}(\gamma - 1)(L_5 + L_1)\right] = 0 \tag{6.38}$$

Most physical boundary conditions have a counterpart LODI relation. For example, imposing a constant entropy on some boundary requires setting $L_2 = 0$, imposing a constant inlet pressure should be accompanied (from (6.34)) by setting $L_5 = -L_1$ to fix the amplitude variation of the wave L_5 entering the domain.

Values obtained for the wave amplitude variations through LODI relations will be approximate because the complete Navier-Stokes equations involve viscous and parallel terms. However, boundary variables will be time advanced using the system of equations (6.19) - (6.23) and viscous and parallel terms will effectively be taken into account at this stage. The LODI relations are used only to estimate the incoming wave amplitude variations.

Here we describe two case, namely for Euler and Navier-Stokes equations. The procedure for Euler equations involves three steps. As an example to illustrate the method we use a subsonic outlet boundary where pressure is specified.

1. For each inviscid boundary condition imposed on this boundary, eliminate the corresponding conservation equation from the system (6.19) - (6.23). For the example of a constant outlet pressure, p is specified and there is no need to use the energy equation. The choice of the conservation equation to eliminate in most practical cases one can find in Table 6.2 [55].
2. For each inviscid boundary condition use the corresponding LODI relation to express the unknown L_i's (corresponding to incoming waves) as a function of the known L_i's (corresponding to outgoing waves). For for a constant outlet pressure

Table 6.2: Conservation equations to eliminate for a given inviscid boundary condition (examples)

Inviscid condition	Equation to eliminate
u_1 velocity imposed	x_1 momentum equation (6.20)
u_2 velocity imposed	x_2 momentum equation (6.21)
u_3 velocity imposed	x_3 momentum equation (6.22)
m_1 flowrate imposed	x_1 momentum equation (6.20)
Pressure imposed	energy equation (6.23)
Density imposed	continuity equation (6.19)
Enthalpy imposed	energy equation (6.23)
Entropy imposed	energy equation (6.23)

the only incoming wave is L_1 (see Figure 6.2) and LODI relation (6.34) suggests that

$$L_1 = -L_5. \tag{6.39}$$

3. Use the remaining conservation equations of the system of equations (6.19) - (6.23) combined with the values of the L_i's obtained from Step 2 to compute all variables which were not given by the inviscid boundary conditions. For a constant pressure outlet the density and the velocities will be obtained through the corresponding conservation equations (6.19), (6.21)-(6.23), where equation (6.39) has been used to evaluate the incoming wave amplitude variation L_1.

Step 2 is the key part of the algorithm. Using the conservation equations written on the boundary as well as some reasonable information on the amplitude of incoming waves (suggested by the LODI relations) removes the ambiguity of having to choose some arbitrary "numerical" conditions. Time advancement of Step 3 of the algorithm includes parallel terms to obtain the solution at the next time step. We use the LODI relations only to estimate the amplitudes of the incoming waves. Since the LODI relations are a simplification of the original system (we neglect some terms while deriving them), the solution of the complete set of equations (6.19) - (6.23) with LODI conditions would not satisfy the physical boundary conditions we have imposed [55]. Step 1 is necessary to discard equations in the system (6.19) - (6.23), which are replaced by inviscid boundary conditions.

The Navier-Stokes equations require more boundary conditions than the Euler equations do. In the Navier-Stokes characteristics boundary conditions (NSCBC) method complete Navier-Stokes boundary conditions are obtained by using Euler inviscid boundary conditions and supplementing them with viscous conditions. In the NSCBC procedure viscous conditions are applied only during Step 3. They are only used to modify the conservation equations applied in Step 3 to compute boundary variables which have not been specified by inviscid conditions. Steps 1 and 2 are the same for the Euler and Navier-Stokes equations.

In the NSCBC method, the number and the choices of physical boundary conditions

(inviscous and viscous) were guided using theoretical studies. However, the agreement between these studies and the results present in [55] is not complete.

6.2.2 Boundary conditions used in calculations

Below we list boundary conditions from [55] which are used for the calculations presented in Chapter 7. We have a subsonic flow, so the most important boundary conditions are subsonic inflow and outflow. In what follows we give some insight how these conditions might be obtained using the technique presented above.

For subsonic inflow all components of velocity $\mathbf{u} = (u_1, u_2, u_3)$ as well as temperature T are imposed.

$$u_1(0, x_2, x_3, t) = U_b(x_2, x_3, t),$$

$$u_2(0, x_2, x_3, t) = V_b(x_2, x_3, t),$$

$$u_3(0, x_2, x_3, t) = W_b(x_2, x_3, t),$$

$$T(0, x_2, x_3, t) = T_b(x_2, x_3, t).$$

For a subsonic three-dimensional flow, four characteristic waves are entering the domain (see Figure 6.2), while one of them (namely L_1) is leaving the domain with speed $\lambda_1 = u_1 - c$. Therefore the density ρ (or pressure p) has to be determined by the flow itself. We have four physical boundary conditions (for u_1, u_2, u_3 and T) and one numerical (for ρ). No viscous relation is needed in this case. To advance the solution in time at the boundary, we need to determine the amplitudes L_i $(i = 2, 3, 4, 5)$ of the different waves crossing the boundary. Only one of them (L_1) may be computed from the interior points.

1. The inlet velocities u_1, u_2 and u_3 are imposed, so equations (6.21), (6.22) and (6.23) are not needed. The inlet temperature T is imposed, so (6.20) is also not needed.

2. As the inlet velocity u_1 is imposed, the LODI relation (6.35) suggests the following expression for L_5

   $$L_5 = L_1 - 2\rho c \frac{dU_b}{dt}.$$

   As the inlet temperature is imposed, the LODI relation (6.38) gives

   $$L_2 = \frac{1}{2}(\gamma - 1)(L_5 + L_1) + \frac{\rho c^2}{T_b}\frac{dT_b}{dt}.$$

   LODI relations (6.36) and (6.37) show that $L_2 = -dV_b/dt$ and $L_4 = -dW_b/dt$.

3. The density ρ can now be obtained by using equation (6.19).

For subsonic non-reflecting outflow we have the following. Consider a subsonic outlet where we want to implement our non-reflecting boundary condition. We see that four characteristic waves L_2, L_3, L_4 and L_5 leave the domain while one of them, L_1, is entering the domain with speed $\lambda_1 = u_1 - c$. Specifying one inviscid boundary condition for the primitive variables would generate reflected waves. For example, imposing the static pressure at the outlet $p = p_\infty$ leads to a well-posed problem, but will create acoustic wave reflection. To avoid such a situation we need to impose a "soft" boundary condition. But at the same time we want to add some physical information about the mean static pressure p_∞ to our set of boundary conditions so that our problem remains well-posed. Below we describe a procedure which allows to do it.

1. We have only one boundary condition: pressure p_∞ at infinity is imposed. This condition does not fix any of the dependent variables at the boundary and we keep all conservation equations in the system (6.19)-(6.23).

2. The condition of constant pressure at infinity is used to obtain the amplitude variation of the incoming wave L_1, the simplest way to insure well-posedness is to set [55]

$$L_1 = K(p - p_\infty), \tag{6.40}$$

   where K is a constant: $K = \sigma(1 - M^2)c/L$. M is the maximum Mach number in the flow, L is the characteristic size of the domain, c is the speed of sound, σ is a constant, which controls the level of reflection.

   If we consider a viscous flow, the viscous conditions require that the tangential stresses τ_{12} and τ_{13} and the normal heat flux q_1 have zero spatial derivatives along x_1. These conditions may be implemented in the system (6.19)-(6.23) directly.

3. All the L_i with $i \neq 1$ may be estimated from the interior points. L_1 is given by the equation in step 2 and the system of equations (6.19)-(6.23) may be used to advance the solution in time at the boundary.

In [55] a number of test cases is mentioned, in particular non-reacting ducted shear layer, non-reacting free shear layer, reacting free shear layer, acoustic wave propagation, vortex convection and Poiseuille flow. Four different sets of outflow boundary conditions were used for numerical comparison for different flow situations, namely

1. The method used in [74]. It is based on partial use of extrapolation and Riemann invariants. In this method values of velocities and density are extrapolated, while for the pressure a condition similar to the NSCBC condition (6.40) is used. In [55] numerical results obtained with this method were used as a reference for comparison.

2. The NSCBC formulation corresponding to perfectly non-reflecting boundary conditions ($\sigma = 0$).

3. The Corrected non-reflecting NSCBC formulation ($\sigma = 0.25$).

4. The reflecting outlet maintained at a constant static pressure p_∞.

The results of the tests show that in certain situations, for example in case of Poisseuille flow, the corrected non-reflecting condition gives better results than the perfectly non-reflecting one.

6.3 Boundary conditions for artificial internal boundaries

As mentioned in Chapter 3, we want to use local grid refinement techniques for calculations, so we have to provide some boundary conditions for the local grid problem. Since we have information from the coarse grid, we can use it to estimate the incoming and outgoing waves from the corresponding coarse grid solution and impose those quantities for the waves which enter the domain as Dirichlet boundary conditions. There are two possible situations concerning Dirichlet boundary conditions for the fine grid: first, we can operate with "physical" variables like velocities, pressure, temperature, etc.; second, we can use the estimates of the corresponding L_i and impose those as the boundary conditions. Below we consider a numerical example, which has the same nature of wave propagation as the problem of interest.

6.3.1 Numerical example: spreading of an acoustic pulse

Acoustic phenomena are typical for all types of compressible flows. One of the simplest examples of such flows could be described by linearized Euler equations. Those equations are derived from the nonlinear Euler equations describing flow of inviscid gas. The idea is to linearize the flow around a time-invariant reference flow. All variables are expanded as a sum of a mean part and a fluctuating part. Consider small amplitude disturbances superimposed on a uniform mean flow with density ρ_0, pressure p_0 and velocity u_0 in the x-direction. The linearized Euler equations for the disturbances in two dimensions are

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{e}}{\partial x} + \frac{\partial \mathbf{f}}{\partial y} = \mathbf{h}, \tag{6.41}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ u \\ v \\ p \end{bmatrix}, \qquad \mathbf{e} = \begin{bmatrix} \rho_0 u + \rho u_0 \\ u_0 u + p/\rho_0 \\ u_0 v \\ u_0 p + \gamma p_0 u \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} \rho_0 v \\ 0 \\ p/\rho_0 \\ \gamma p_0 v \end{bmatrix}.$$

The nonhomogeneous term $\mathbf{h}$ on the right side of (6.41) represents disturbed unsteady sources.

For our test problem we choose the time advancement of an initial acoustic pulse in a rectangular domain (see Figure 6.3). If we look at the pressure distribution, initially it

has the form of a Gaussian pulse, but with the time this pulse spreads out in the domain and is moved by the main flow. For the computations presented below we use the domain $\Omega = \{0 \leq x \leq 50, 0 \leq y \leq 50\}$ (see Figure 6.4).

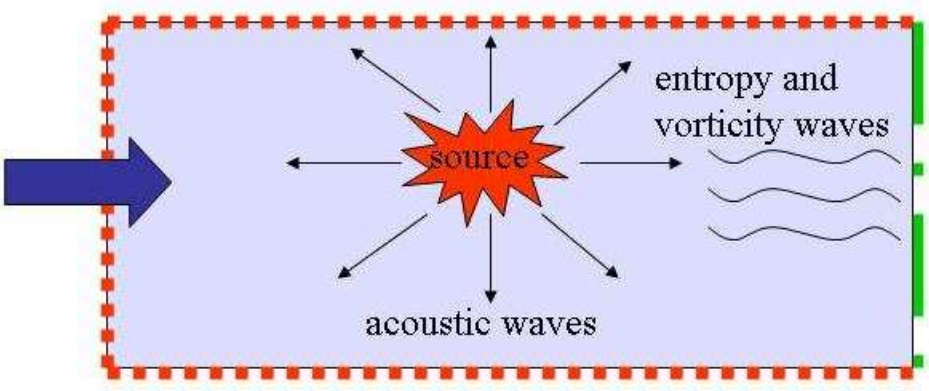


Figure 6.3: Acoustic pulse problem. We mark with dots and slash-dots parts of the boundary where radiation and outflow boundary conditions are used, respectively.

The crucial parts in the acoustical simulation are the boundary conditions prescribed. As mentioned in the previous section there are several main approaches for construction of the non-reflecting boundary conditions

- the extrapolation method;
- usage of damping zone;
- the method of characteristics;
- Fourier-Laplace transform and expansion at infinity. This method is specific to acoustic problems and therefor was not introduced in the previous section and is shortly presented in the next section.

According to [30] the best performance in aeroacoustic problems is shown by the last approach.

The major goal of this section is to test the possibility of Dirichlet boundary conditions for the fine grid. So for the coarse grid solution we just choose the best boundary conditions known from the literature and for the linearized Euler equations of aeroacoustics these are the boundary conditions developed in [71]. We state them below:

The radiation boundary conditions are given by

$$\left(\frac{1}{V(\Theta)}\frac{\partial}{\partial t}+\frac{\partial}{\partial r}+\frac{\partial}{2r}\right)\begin{bmatrix}\rho\\ u\\ v\\ p\end{bmatrix}=0, \tag{6.42}$$

where $V(\Theta)=a_0\left[M\cos\Theta+\left(1-M^2\sin^2\Theta\right)^{1/2}\right]$, a_0 is the speed of sound of the main flow and r, Θ are polar coordinates with the origin in the center of the computational domain (in the present case in the point with coordinates $(0,0)$).

The outflow boundary conditions are given

$$\frac{\partial\rho}{\partial t}+u_0\frac{\partial\rho}{\partial x} = \frac{1}{a_0^2}\left(\frac{\partial p}{\partial t}+u_0\frac{\partial p}{\partial x}\right), \tag{6.43}$$

$$\frac{\partial u}{\partial t}+u_0\frac{\partial u}{\partial x} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x}, \tag{6.44}$$

$$\frac{\partial v}{\partial t}+u_0\frac{\partial v}{\partial x} = -\frac{1}{\rho_0}\frac{\partial p}{\partial y}, \tag{6.45}$$

$$\frac{1}{V(\Theta)}\frac{\partial p}{\partial t}+\cos\Theta\frac{\partial p}{\partial x} + \sin\Theta\frac{\partial p}{\partial y}+\frac{p}{2r}=0. \tag{6.46}$$

The movement of the gas is described (6.41). At the outflow boundary (downstream the main flow ($x=50$)) we prescribe outflow boundary conditions (6.43) - (6.46) and the rest of boundaries ($x=0$, $y=0$, $y=50$) are of radiation type (6.42). The initial conditions are

$$p(x_1,x_2) = 1/\gamma, \tag{6.47}$$

$$\rho(x_1,x_2) = 1, \tag{6.48}$$

$$u_1(x_1,x_2) = M+\epsilon x_2\exp\left(-\ln(2)\left(x_1^2+x_2^2\right)/b^2\right), \tag{6.49}$$

$$u_2(x_1,x_2) = \epsilon x_1\exp\left(-\ln(2)\left(x_1^2+x_2^2\right)/b^2\right), \tag{6.50}$$

where the flow Mach number is $M=0.5$, the Gaussian half-width $b=5$ and the amplitude $\epsilon=0.03$ and $\gamma=1.4$.

The analytical solution of this initial value problem is given by [71]

$$p(x_1,x_2,t)=\frac{1}{\gamma}+\frac{\epsilon}{2\alpha}\int_0^\infty \xi\exp\left[-\xi^2/(4\alpha)\right]\cos(c_0t\xi)J_0(\eta\xi)d\xi, \tag{6.51}$$

where $\eta=\sqrt{(x_1-Mt)^2+x_2^2}$, c_0 is the speed of sound in the main flow, $\alpha=(\ln 2)/b^2$ and $J_0(z)$ is the Bessel function of the first kind and order zero.

We first discretize our initial value problem in space and time. Consider the approximation of the first derivative $\partial f/\partial x$ at node i of a uniform one-dimensional grid. Suppose

	Case $M = N = 3$	Case $N = 0, M = 6$	Case $N = 1, M = 5$	Case $N = 2, M = 4$
-3	0.02651995	-	-	-
-2	-0.18941314	-	-	0.0346861
-1	0.79926643	-	-0.173186	-0.408117
0	0	-2.3947	-2.39476	-0.563042
1	-0.79926643	5.6686	2.40222	1.30628
2	0.18941314	-6.6715	-1.53629	-0.479708
3	-0.02651995	5.5620	0.73555	0.125217
4	-	-2.9215	-0.210886	-0.0153139
5	-	0.8686	0.0268144	-
6	-	-0.11143	-	-

Table 6.3: Coefficients a_j for some different stencils

M values of f to the right and N values of f to the left of this point are used to form the finite difference approximation

$$\left(\frac{\partial f}{\partial x}\right)_i \simeq \frac{1}{\Delta x} \sum_{j=-N}^{M} a_j f_{i+j}. \tag{6.52}$$

The special case of $M = N$ is of particular interest. It can be shown that if M and N are not equal that such an asymmetric stencil causes spatially growing wave solutions when it is used over a large region. Asymmetric stencils may, however, be employed in limited regions (such as boundary regions of the computational domain) without leading to accumulated numerical instability [70]. The coefficients a_j for some stencils are presented in Table 6.3. In the example presented in this section we use the same time integration scheme as in the original paper [70]. It is an explicit time marching scheme, although we can also use implicit ones. The linearized Euler equations (6.41) provide the time derivatives of $\mathbf{u}$. Suppose the solution is known up to a time level $t = n\Delta t$. To advance to the next time step a four-level scheme is used in form

$$\mathbf{u}^{n+1} - \mathbf{u}^n \simeq \Delta t \sum_{j=0}^{3} b_j \left(\frac{d\mathbf{u}}{dt}\right)^{n-j}. \tag{6.53}$$

To ensure that the scheme is consistent, three of the four coefficients b_j are chosen so that (6.53) is satisfied to third-order in time. The remaining coefficient b_0 is determined by requiring that the Laplace transform of scheme (6.53) is a good approximation of that of the partial derivative. This gives the following coefficients [70]:

$$b_0 = 2.30255809,\ b_1 = -2.49100760,\ b_2 = 1.57434093,\ b_3 = -0.38589142.$$

For a symmetric space stencil ($M = N = 3$) the complete discretized problem looks like

$$\mathbf{u}(x, y, t + \Delta t) = \mathbf{u}(x, y, t) + \Delta t \sum_{j=0}^{3} b_j \mathbf{k}(x, y, t - j\Delta t), \tag{6.54}$$

$$\mathbf{k}(x,y,t) = -\frac{1}{\Delta x}\sum_{j=-3}^{3} a_j \mathbf{e}(x+j\Delta x, y, t) - \frac{1}{\Delta x}\sum_{j=-3}^{3} a_j \mathbf{f}(x, y+j\Delta y, t) + \mathbf{h}(x,y,t), \quad (6.55)$$

$$\mathbf{u}(x,y,t) = \begin{cases} \mathbf{u}_{\text{initial}}(x,y), & 0 \le t \le \Delta t \\ 0, & t < 0. \end{cases} \quad (6.56)$$

As one can see from (6.54), we use an explicit time integration scheme. This leads to a time step restriction, which in the case of problem (6.54) looks like [71]

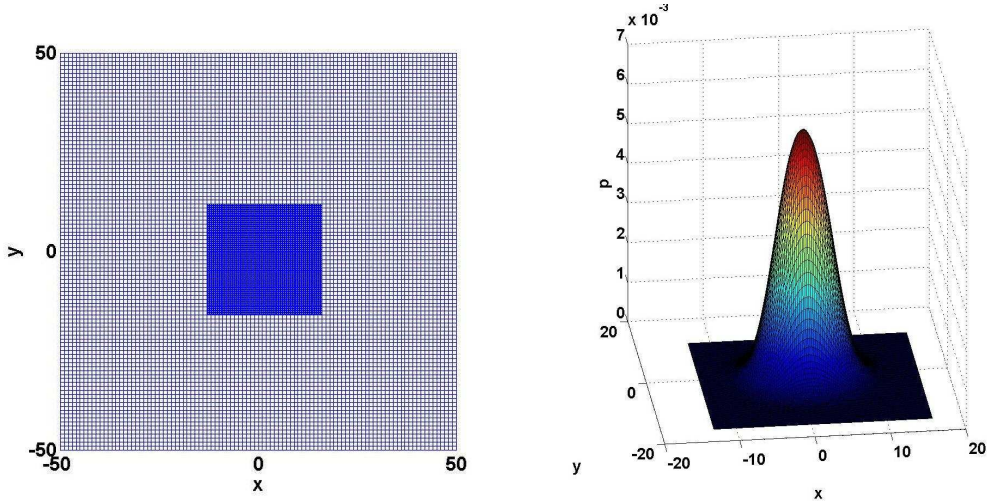
$$\Delta t_{\max} = \frac{0.4}{1.75\left[M + (1 + (\Delta x/\Delta y)^2)^{0.5}\right]}\frac{\Delta x}{a_0}. \quad (6.57)$$

Due to the stability criterion (6.57) time step sizes of the fine and coarse grid solutions have to be different in case of local grid refinement. In fact, since the stability criterion (6.57) is linear in Δx, we should set the time refinement factor τ to 4 when our space refinement factor σ is equal to 4.

To solve problem (6.54) and to test artificial boundary conditions, we consider the following grid refinement strategy for our model problem:

1. Coarse grid solution. On the coarse grid we solve equation (6.41) with boundary conditions (6.42)-(6.46). In the interior points we use the symmetric 7-point stencil and scheme described by (6.54). In the boundary region we use the asymmetric 7-point stencil and discretized version of the boundary conditions (6.42)-(6.46).
2. Fine grid solution. First we determine the position of the high activity region. Several options exist; in the present calculations we compute the gradient of the coarse grid solution to determine where the high activity is. Then we construct a fine grid which covers that region. The next step is that we use interpolation (linear or cubic) to get initial conditions for the fine grid problem.
3. We advance our solution in time on the fine grid using the symmetric 7-point stencil and scheme described by (6.54). As for the boundary conditions, several situations are possible. The first one is when the region of refinement lies completely in the coarse domain. In this case we prescribe Dirichlet boundary conditions at all boundaries. Another situation occurs when part of the fine grid boundary corresponds to the coarse grid boundary. In this case at the common part of the boundary we prescribe the original boundary condition (6.42) or (6.43)-(6.46) and Dirichlet boundary conditions at the rest of the fine grid boundaries.
4. Once we reached the same time point for both fine and coarse grid solutions, we perform an interpolation from the fine grid to the coarse one, and replace values in the coarse grid solution with those interpolated from the fine grid.
5. We repeat steps 1-4 until we reach the final time.

The algorithm for the linearized 2D Euler equations was implemented and uses the 7-point stable centered high order finite difference scheme for the space derivatives and time scheme (6.54)-(6.55) for the time derivative. Some of the results one can find in Figure 6.4. The choice of time for Figure 6.4 is quite arbitrary since we use this picture only to show the solution's behavior. As one can see from Figure 6.4 we have a Gaussian type of pressure fluctuation distribution initially. Our fine grid is concentrated near the origin and is quite small (see Figure 6.4 (a)). With time the area occupied by the pressure fluctuations is growing and moving. We see that in certain points pressure fluctuations get negative values (see Figure 6.5). Since the initial pulse spreads through the domain, our fine grid also increases in size and moves (see Figure 6.5(a)).

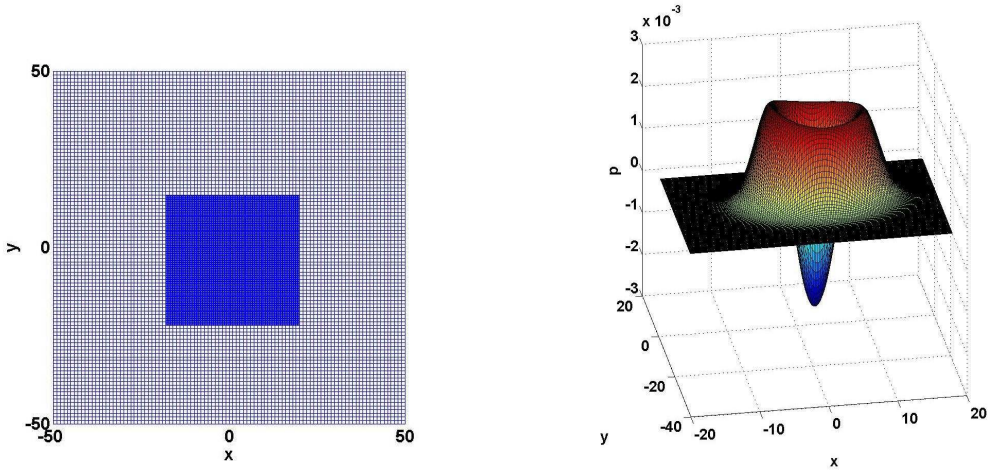


(a) Coarse and fine grids

(b) Pressure distribution on fine grid

Figure 6.4: $t = 8.35$

As one can see from Figures 6.4-6.6, Dirichlet boundary conditions coming from the coarse grid used on the fine grid do not pose any reflections on the boundaries. As one can see from Figure 6.6, the outflow boundary conditions (6.43)-(6.46) work well. Here the pulse leaves the computational domain. In Chapter 7 we present some more examples of the use of Dirichlet boundary conditions in actual calculations.



(a) Coarse and fine grids

(b) Pressure distribution on fine grid

Figure 6.5: t = 11.1

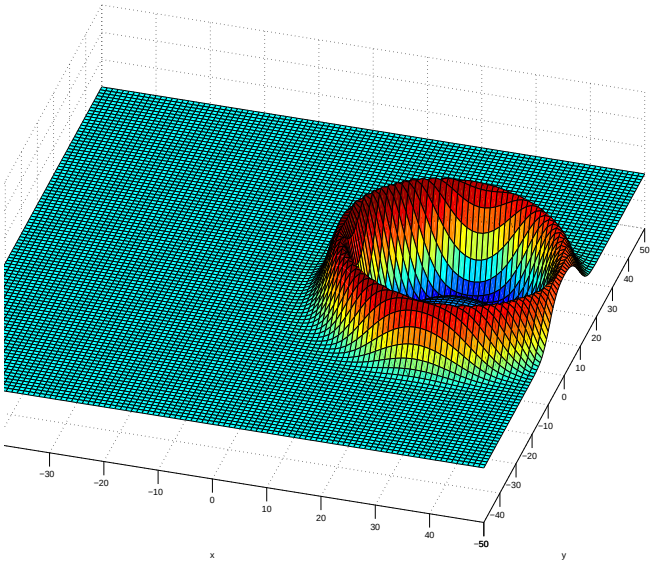


Figure 6.6: Time at which the pulse crosses the boundary of the coarse grid (non-reflecting boundary conditions are used)

Chapter 7

Numerical simulation of air film cooling

Numerical methods for the direct simulation of turbulent flows are required to accurately reproduce its evolution over a wide range of length and time scales. This chapter discusses some of the issues involved. We start with a recap of the mathematical model. Spatial discretization and the time advancement scheme are considered in Section 7.2 which is followed in Section 7.3 by a discussion of domain decomposition and parallelization. This chapter is concluded by numerical results in Section 7.4.

7.1 Mathematical description of the film cooling problem

Below we briefly repeat the mathematical model, described in Chapter 2. Flow of a compressible ideal gas is described by the following system of equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{7.1}$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = -\nabla p - \nabla \cdot \Upsilon, \tag{7.2}$$

$$\rho T \left(\frac{\partial s}{\partial t} + \mathbf{u} \cdot \nabla s \right) = -\nabla \cdot \mathbf{q} - \Upsilon \otimes \nabla \mathbf{u}, \tag{7.3}$$

where ρ, t, $\mathbf{u}$, Υ, s and T denote density, time, velocity vector, stress tensor, entropy and temperature respectively. The heat flux vector $\mathbf{q}$ can be expressed as a function of

Figure 7.1: Sketch of the computational domain

temperature using Fourier's law

$$\mathbf{q} = -\lambda \nabla T, \tag{7.4}$$

where

$$\lambda = \frac{\mu c_p}{Pr},$$

is the thermal conductivity, Pr is the Prandtl number and c_p is specific heat at constant pressure. For our calculations we take $Pr = 0.72$. The dynamic viscosity μ is a function of the temperature T and is given by Sutherland's law

$$\mu = 1.458 * 10^{-6} \frac{T^{1.5}}{T + 110.4}.$$

The set of equations is closed by constitutive relations for the temperature and pressure

$$T = \frac{p}{\rho R}, \; p = p_\infty \left(\frac{\rho}{\rho_\infty} \right)^\gamma \exp\left(\frac{s - s_\infty}{c_v} \right), \tag{7.5}$$

where the subscript ∞ indicates a reference condition ($p_\infty = 1.03 * 10^5$ Pa, $s_\infty = 0$ J$*$K^{-1}, $\rho_\infty = 1.2$ kg $*$ m^{-3})and $\gamma = 1.4$ (γ is the ratio of specific heat at constant pressure c_p to specific heat at constant volume c_v). If we introduce R as the ideal gas constant, then we have the following relations $\gamma = \frac{c_p}{c_v}$, $c_p = c_v + R$, $c_v = \frac{R}{\gamma - 1}$.

In Figure 7.1 we have sketched the computational domain. It is given in Cartesian coordinates by $\Omega = \{(x, y, z) \in \mathbb{R}^3 | 0 \leq x \leq X, 0 \leq y \leq Y, 0 \leq z \leq Z\}$.

At the inlet plane ($x = 0$) we prescribe boundary layer profile and temperature.

$$u(0, y, z, t) = U_b(y, z), \tag{7.6}$$

$$v(0, y, z, t) = V_b(y, z), \tag{7.7}$$

$$w(0, y, z, t) = W_b(y, z), \tag{7.8}$$

$$T(0, y, z, t) = T_\infty(y, z), \tag{7.9}$$

where the subscript b stands for Blasius because those values come from numerically solving a Blasius problem with a boundary layer on flat plate and T_∞ is the inflow temperature (for more details on Blasius profile see Chapter 2).

As mentioned in Section 2.3 we apply periodic boundary conditions at $z = 0$ and $z = Z$.

$$\begin{aligned} u(x, y, 0, t) &= u(x, y, Z, t), \\ v(x, y, 0, t) &= v(x, y, Z, t), \\ w(x, y, 0, t) &= w(x, y, Z, t), \\ s(x, y, 0, t) &= s(x, y, Z, t). \end{aligned} \tag{7.10}$$

At the bottom plane ($y = 0$) we assume adiabatic wall conditions except for the cooling nozzle, where the profile and temperature of the cooling jet are prescribed. So everywhere at $y = 0$, except at the hole, we impose

$$u(x, 0, z, t) = v(x, 0, z, t) = w(x, 0, z, t) = 0, \tag{7.11}$$

$$\frac{\partial T(x, 0, z, t)}{\partial n_w} = 0, \tag{7.12}$$

where n_w is the normal to the wall (for our problem of interest $n_w = y$). At the exit of the film cooling hole we prescribe the velocity profile and temperature distribution, viz.

$$u(x, 0, z, t) = u_j(x, z, t), \tag{7.13}$$

$$v(x, 0, z, t) = v_j(x, z, t), \tag{7.14}$$

$$w(x, 0, z, t) = w_j(x, z, t), \tag{7.15}$$

$$T(x, 0, z, t) = T_j(x, z, t), \tag{7.16}$$

where the subscript j stands for jet. We have several options: the simple approach is just to prescribe an undisturbed Poiseuille type profile; a more sophisticated (and therefor more detailed) one includes the solution of a supplemental problem. We describe how to get those values in Section 7.3 in detail.

At the outflow boundary ($x = X$) we assume stress free outflow in the tangential direction. Furthermore we require the derivative of the heat fluxes leaving the domain in streamwise direction to be zero

$$\frac{\partial \tau_{xy}}{\partial x} = 0, \ \frac{\partial \tau_{xz}}{\partial x} = 0, \ \frac{\partial q}{\partial x} = 0. \tag{7.17}$$

The pressure at the outlet boundary is forced towards its reference value by imposing

$$\frac{\partial p}{\partial x} = \frac{p_\infty - p}{X}. \tag{7.18}$$

The top boundary is treated analogously to the outflow one by forcing the pressure towards some reference value and taking special care not to get reflections (see Chapter 6 for details).

7.2 Discretization in space and time

7.2.1 Spatial discretization

We choose in the spatial domain (see Figure 7.1) a Cartesian structured grid. The spatial derivatives in the governing equations (7.1)-(7.5) are approximated using finite differences. Quite a number of different compact and non-compact upwind and central difference schemes have been developed for turbulence simulations. An overview can be found in Section 5.1. We choose the following schemes for our calculations based on the experiments in [16].

A compact finite difference sixth-order scheme proposed by Lele [38] is applied to the non-convective fluxes

$$\frac{1}{3}\left(f'_{i-1} + f'_{i+1}\right) + f'_i = \frac{1}{36h}\left(f_{i+2} + 28f_{i+1} - 28f_{i-1} - f_{i-2}\right), \tag{7.19}$$

where h is the space step size.

A fourth-order explicit biased upwind scheme is available for calculating the convective fluxes [84]:

$$f'_i = \frac{s}{12h}\left(-f_{i-3s} + 6f_{i-2s} - 18f_{i-s} + 10f_i + 3f_{i+s}\right), \tag{7.20}$$

where s accounts for the upwind direction, i.e. $s = 1$ for a positive and $s = -1$ for a negative velocity. In addition a fifth-order compact upwind scheme is available [84]

$$\frac{1}{6}\left(f'_{i-s} + f'_{i+s}\right) + f'_i = \frac{s}{18h}\left(10f_{i+s} + 9f_i - 18f_{i-s} - f_{i-2s}\right). \tag{7.21}$$

As mentioned in Section 2.3 we apply periodic boundary conditions at $z = 0$ and $z = Z$. Near all other boundaries (in- and outflow, top boundary, wall) we have to propose closing schemes, since we cannot use interior schemes (7.19)-(7.21) in those areas. Below we present differential schemes for all non-periodic boundary points as well as differential schemes for points next to non-periodic boundaries.

The first derivative at the boundary $i = 1$ can be obtained from a third-order compact scheme [38]

$$f'_1 + 2f'_2 = \frac{1}{2h}\left(4f_2 - 5f_1 + f_3\right). \tag{7.22}$$

	Central schemes	Upwind schemes
Interior points	compact sixth-order, symmetric (7.19)	compact fifth-order (7.21)
Points next to boundary	compact fourth-order, symmetric (7.23)	compact third-order (7.24)
Boundary points	compact fifth-order (7.22)	compact fifth-order (7.22)

Table 7.1: Spatial discretization scheme applied.

Instead of the sixth-order compact central scheme (7.19) at the points next to the boundary the following scheme can be applied [38]

$$\frac{1}{4}\left(f'_{i-1} + f'_{i+1}\right) + f'_i = \frac{3}{4h}\left(f_{i+1} - f_{i-1}\right). \tag{7.23}$$

Instead of the compact-fifth order upwind scheme next to the boundary a third-order compact scheme can be applied

$$\frac{1}{16}\left(7f'_{i-s} + f'_{i+s}\right) + f'_i = \frac{3s}{8h}\left(-3f_{i-s} + 2f_i + f_{i+s}\right). \tag{7.24}$$

To summarize, the spatial discretization schemes used for the simulation of the film cooling problem are listed in Table 7.1.

7.2.2 Time discretization

Since the flows in the film cooling problem are subsonic with Reynolds number Re larger than one, the acoustic fluxes will evolve faster than the convective ones and the convective fluxes will evolve faster than the diffusive ones. In explicit time-integration this means that stability imposes a severe time step restriction. This gives a possibility [16] to employ a split-time integration in order to reduce computational times, which uses separate time stepping for the different fluxes.

The proposed numerical time integration [16] starts from the splitting of the Navier-Stokes equations (7.1)-(7.5) in three separate operators:

$$\frac{\partial \mathbf{f}}{\partial t} = A(\mathbf{f}) + C(\mathbf{f}) + D(\mathbf{f}), \tag{7.25}$$

where $\mathbf{f}$ denotes the vector of unknown primitive variables and $A(\mathbf{f})$, $C(\mathbf{f})$ and $D(\mathbf{f})$ denote the acoustic, convective and diffusive contributions, respectively. The time-split integration uses separate time steps for the three different fluxes (see Figure 7.2). For

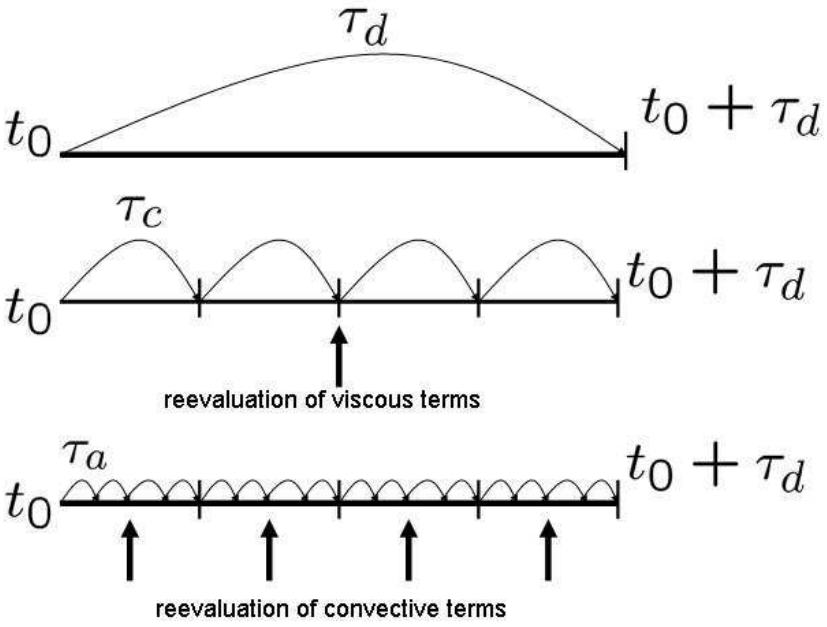


Figure 7.2: Time-split integration. We assume here that per diffusive time step we perform four convective time steps ($\tau_c = 0.25\tau_d$) and per convective time step we perform four acoustic time steps ($\tau_a = 0.25\tau_c$).

the diffusive fluxes a second-order Runge-Kutta scheme is used:

$$\begin{aligned}
\mathbf{d}_0 &= \mathbf{f}(t_d), \\
\mathbf{d}_1 &= \mathbf{d}_0 + \int_{t_d}^{t_d+0.5\tau_d} \left(A\left(\mathbf{f}(t)\right) + C\left(\mathbf{f}(t)\right) + D\left(\mathbf{d}_0\right)\right) dt, \\
\mathbf{d}_2 &= \mathbf{d}_1 + 0.5\tau_d \left(D(\mathbf{d}_1) - D(\mathbf{d}_0)\right), \\
\mathbf{f}(t_d+\tau_d) &= \mathbf{d}_2 + \int_{t_d+0.5\tau_d}^{t_d+\tau_d} \left(A\left(\mathbf{f}(t)\right) + C\left(\mathbf{f}(t)\right) + D\left(\mathbf{d}_1\right)\right) dt,
\end{aligned} \tag{7.26}$$

where $D\left(\mathbf{f}(t_d)\right)$ represents the stress and diffusive contributions in $\frac{\partial \mathbf{f}}{\partial t}$ evaluated in solution vector $\mathbf{f}$ at time t_d. The diffusive time step size τ_d must be chosen in agreement with the stability criterion for the diffusive fluxes (see (7.29)). For the internal integration in steps 2 and 4 the same procedure is followed to separate acoustics from convection:

$$\begin{aligned}
\mathbf{c}_0 &= \mathbf{f}(t_c), \\
\mathbf{c}_1 &= \mathbf{c}_0 + \int_{t_c}^{t_c+0.5\tau_c} \left(A\left(\mathbf{f}(t)\right) + C\left(\mathbf{c}_0\right) + D\left(\mathbf{d}_0\right)\right) dt, \\
\mathbf{c}_2 &= \mathbf{c}_1 + 0.5\tau_c \left(C(\mathbf{c}_1) - C(\mathbf{c}_0)\right), \\
\mathbf{f}(t_c+\tau_c) &= \mathbf{c}_2 + \int_{t_c+0.5\tau_c}^{t_c+\tau_c} \left(A\left(\mathbf{f}(t)\right) + C\left(\mathbf{c}_1\right) + D\left(\mathbf{d}_0\right)\right) dt,
\end{aligned} \tag{7.27}$$

where τ_c is the convective time step size. Again, $C\left(\mathbf{f}(t_c)\right)$ represents the convective contributions in $\frac{\partial \mathbf{f}}{\partial t}$ evaluated in solution vector $\mathbf{f}$ at time t_c. After half of the convective time steps have been performed during one diffusive time step, the diffusion terms are reevaluated and the values of these terms at t_d are replaced by those at $t_d + 0.5\tau_d$.

For the time integration of the acoustic terms, a third-order Runge-Kutta scheme is used

$$\begin{aligned}
\mathbf{a}_0 &= \mathbf{f}(t_a) + \frac{1}{4}\tau_a \left(A\left(\mathbf{f}(t_a)\right) + C\left(\mathbf{c}_0\right) + D\left(\mathbf{d}_0\right)\right), \\
\mathbf{a}_1 &= \mathbf{f}(t_a) + \frac{8}{15}\tau_a \left(A\left(\mathbf{f}(t_a)\right) + C\left(\mathbf{c}_0\right) + D\left(\mathbf{d}_0\right)\right), \\
\mathbf{a}_2 &= \mathbf{a}_0 + \frac{5}{12}\tau_a \left(A\left(\mathbf{a}_1\right) + C\left(\mathbf{c}_0\right) + D\left(\mathbf{d}_0\right)\right), \\
\mathbf{f}(t_a+\tau_a) &= \mathbf{a}_0 + \frac{3}{4}\tau_a \left(A\left(\mathbf{a}_2\right) + C\left(\mathbf{c}_0\right) + D\left(\mathbf{d}_0\right)\right),
\end{aligned} \tag{7.28}$$

where τ_a is the acoustic time step size. $A\left(\mathbf{f}(t_a)\right)$ represents the acoustic contributions in $\frac{\partial \mathbf{f}}{\partial t}$ evaluated in solution vector $\mathbf{f}$ at time t_a. Again, like with the diffusive terms, after half of the acoustic time steps have been performed during one convective time step, the convective terms are reevaluated and the values of these terms at t_c are replaced by those at $t_c + 0.5\tau_c$.

As we use explicit time integration, *stability restrictions* (like CFL criteria) on the maximum time step size have to be considered. For the acoustic, convective and two diffu-

sive (viscous contribution to the momentum equation and heat dissipation in the energy equation) terms the following critical time step sizes hold [16]

$$\begin{aligned}
\frac{1}{T_a} &= \max\left(\frac{|u|+c}{h_x}, \frac{|v|+c}{h_y}, \frac{|w|+c}{h_z}\right), \\
\frac{1}{T_c} &= \max\left(\frac{|u|}{h_x}, \frac{|v|}{h_y}, \frac{|w|}{h_z}\right), \\
\frac{1}{T_{d,visc}} &= \max\left(\frac{\mu}{\rho(h_x)^2}, \frac{\mu}{\rho(h_y)^2}, \frac{\mu}{\rho(h_z)^2}\right), \\
\frac{1}{T_{d,heat}} &= \max\left(\frac{\lambda}{\rho c_p(h_x)^2}, \frac{\lambda}{\rho c_p(h_y)^2}, \frac{\lambda}{\rho c_p(h_z)^2}\right),
\end{aligned} \tag{7.29}$$

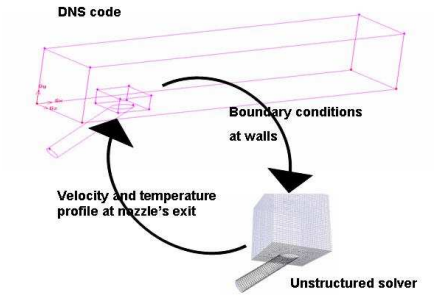
where c is the speed of sound and h_x, h_y and h_z are space step sizes in x, y and z directions respectively.

7.3 Domain decomposition and parallelization

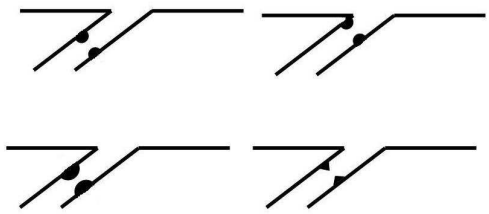
7.3.1 Domain decomposition

In the DNS code the cooling jet is modeled via the boundary conditions at the bottom of the computational domain. It is not clear from the experiments which velocity and temperature profiles one should prescribe at the cooling nozzle's exit. Because of the nature of the developed DNS code it is not possible to incorporate the underlying geometry of the cooling nozzle. Hence we propose to use *domain decomposition*. The essence of the method is that we first assume some profile and solve the DNS problem. Then, using a commercial unstructured solver, we solve a supplemental problem modelling the cooling nozzle with boundary conditions obtained from the DNS results. The result of this simulation gives a new profile for the nozzle's exit. We can once again solve the DNS problem with the updated profile. Several geometries of the underlying nozzle will be treated, mainly those used in the experiments. It should be noted that such an approach allows us to incorporate almost any inaccuracies of the cooling nozzle.

A lot of parameters influence the flow (and therefore the velocity profile) in the cooling nozzle. From experiments [32] we know that three main geometrical parameters have major influence. These are the position, the size and the shape of the inaccuracy (see Figure 7.3 (b)). In the top left figure we have an inaccuracy positioned far away from the nozzle's exit, while in the top right figure the inaccuracy of the same size is moved towards the exit. In the bottom left figure we have the situation with a larger inaccuracy, while the bottom right figure represents the influence of the shape. Experiments in a water channel show that among these three, the last parameter has a minor influence, while first two have major effect. For practical reasons we can also eliminate the influence of the size. The reason for this is that according to manufacturing specifications



(a) Domain decomposition strategy.



(b) Inaccuracies

Figure 7.3: Domain decomposition

provided by the end user, the geometrical parameters should be within prescribed tolerance. Because of those requirements nozzles with large inaccuracies are not accepted during the final control stages, so we can say that the size difference lies within a narrow region. Although large inaccuracies are not interesting from a practical point of view, we still make some calculations in this case as well as for more realistic geometries. So from three groups of parameters from practical point of view we are mostly interested in the influence of the position of the inaccuracy on the flow parameters. We discuss results of the calculations in Section 7.4. The algorithm calculation of proper boundary conditions for the cooling jet can be summarized as follows (see Figure 7.3 (a)):

1. Solve the main flow problem with the use of the DNS code. As for boundary condition for the cooling jet use either parabolic profile or experimental data if available.

2. For a specified region get artificial boundary conditions for unstructured grid solver from the DNS solution.

3. Solve the complementary flow problem with the use of unstructured solver. Extract the inflow profile for the cooling jet.

4. Solve the main flow problem with the use of the DNS code. As for boundary condition for the cooling jet use the profile from the unstructured grid solution.

5. Repeat steps 2-4.

We should note that the approach described above is limited to quasi-stationary situations, although we can also use this approach for developing flow assuming the inflow profile does not change that much, so that we are able to catch these changes.

7.3.2 Parallelization

Parallel computing is concerned with producing the same results using multiple processors. The problem to be solved is divided up between a number of processors. Ideally, if a program is running on P processors, we would like it to go P times faster than on one processor. In practice this is extremely difficult to achieve due to overheads such as communication time and purely sequential parts of the code e.g., file accesses. In designing a multiple processor computer, an important question needs to be addressed: How do processors coordinate to solve a problem? Processors must have the ability to communicate with each other in order to cooperatively complete a task. One parallel computing architecture uses a single address space. Systems based on this concept, otherwise known as *shared-memory* multiprocessors, allow processor communication through variables stored in a shared address space. Another major architecture for parallel computers employs a scheme by which each processor has its own memory module. Such a *distributed-memory* multiprocessor is constructed by connecting each component with a high-speed communications network. Processors communicate to each other over the network.

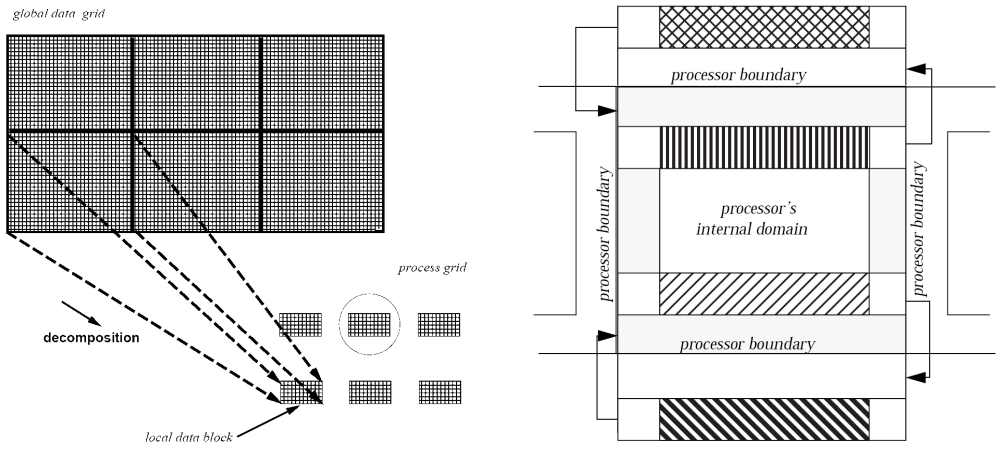
The architectural differences between shared-memory multiprocessors and distributed-memory multiprocessors have implications on how each is programmed. With a shared-memory multiprocessor, different processors can access the same variables. This makes referencing data stored in memory similar to traditional single-processor programs, but adds the complexity of shared data integrity. A distributed-memory system introduces a different problem: how to distribute a computational task to multiple processors with distinct memory spaces and reassemble the results from each processor into one solution.

For the calculations presented in this chapter we use the following distributed-memory computer (Beowulf cluster)

1. Master node
   - Hardware: 2x 2.6 GHz Intel Xeon 512kB cache; 1 GB memory (ECC); 2x 200 GB HDD (in RAID1, i.e. mirrored); 2x Gbit ethernet (1 local, 1 WAN)
   - Software: ClusterVisionOS (modified RedHat) Vanilla kernel (easy updating)
2. Slave nodes (23 in total)
   - Hardware: 2x 2.4 GHz Intel Xeon 512kB cache; 2 GB memory; (ECC) 40 GB HDD (in RAID1, i.e. mirrored); Gbit ethernet
   - Software ClusterVisionOS (modified RedHat) Vanilla kernel (easy updating)
3. In total
   - 48 Xenon CPUs
   - 49 GB internal memory
   - 1320 GB total harddisk space

- Software (at present)
  - MPI (MPICH, LAM)
  - Compilers (gcc, intel)
  - Queuing software
  - Math libs: Atlas (BLAS, LAPACK), PetSc, SLEPc
  - MatLab

Many problems are based on taking a very large set of data, arranged in a regular grid structure, and applying transformations to the data elements. When the data can be split up into regular subgrids, and distributed over a set of processes, then the transformations can be applied in parallel, allowing the problem to be solved in a smaller time scale, or allowing much larger problems to be solved than could normally be attempted. The regular domain decomposition method is to take a large grid of data elements, split it up into regular subgrids, and distribute these subgrids to separate processes where they can be operated on. The global data set is decomposed into separate sections, and each section is placed under the control of a separate process as shown in Figure 7.4 (a).



(a) Decomposition of the global data set

(b) Boundary swapping

Figure 7.4: Parallel solution.

The degree of decomposition depends on the number of processes available. The aim is to ensure that the data is distributed as evenly as possible amongst all of the processes. Each process is assigned its own section of the data - its data block. Before the data grid can be distributed amongst the processes, the processes themselves must be arranged as a logical process grid. This grid of processes must allow the data set to be distributed evenly, and therefore the size of each dimension of the logical process grid will reflect the magnitude of the corresponding dimension of the global dataset. For example, if a two-dimensional data grid is to be distributed over 6 processes, the processes could be arranged as in Figure 7.4 (a).

It is often necessary for data generated within one process to be made available to another process to allow it to perform an update on its own data. When a single data element is updated, the new value can depend on the original value of that element, and also on the values of a number of neighboring elements. If one of these neighboring elements is actually contained within the data grid belonging to a different process, the value of that element must be copied over from that process so that it can be used. In general, whenever a process has successfully updated all of its data elements, it must arrange to send off copies of any data which might be needed by other processes, to those processes. At the same time, it must be prepared to receive data from other processes, which it can then copy into its own overlap areas, to be used in the next iteration. Each iteration can be considered as performing the following within each process:

- Send off copies of any data elements that neighboring processes might require to carry out the next update.
- Receive copies of data elements that neighboring processes have sent to this process, and place them in the appropriate overlap areas. Update every element in the local data grid.

The sending off of copies of data to other processes, and the receiving of copies of data from other processes, is known as boundary swapping (see Figure 7.4).

In order to ensure communication between different processes, we use the Message Passing Interface (*MPI*). MPI is a standard for inter-process communication on distributed-memory multiprocessors. The standard has been developed by a committee of vendors, government labs, and universities. Implementation of the standard is usually left up to the designers of the systems on which MPI runs, but a public domain implementation, MPICH, is available. MPI is a set of library routines for C/C++ and FORTRAN. MPI is a standard interface, so code written for one system can easily be ported to another system with those libraries.

The execution model of a program written with MPI looks as follows: when an MPI program starts, the program spawns into the number of processes as specified by the user. Each process runs and communicates with other instances of the program, possibly running on the same processor or different processors. The greatest computational speedup will occur when processes are distributed among processors. Basic communication consists of sending and receiving data from one process to another. This communication takes place over a high-speed network which connects the processors in the distributed-memory system.

In Figure 7.5 we plot timings for one time step for a parallelized version of the code for different problem sizes. Note that we use parallelization only in one (vertical) direction. We plot two results - for the problem of the size $128 \times 64 \times 64$ (line marked with squares) and for the problem of the size $128 \times 64 \times 128$. With the dashed line we mark the theoretical limit (that is when calculation time scales with the number of processors available). As one can see from Figure 7.5, our problem of interest scales quite good, although we still are quite far from the theoretical limit.

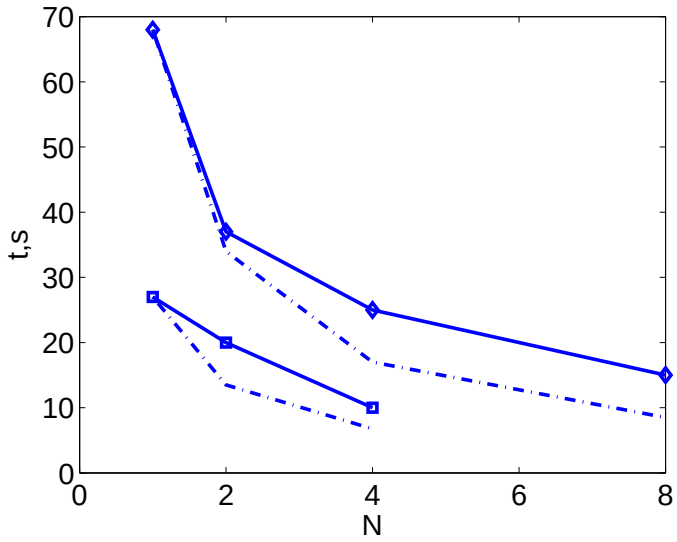


Figure 7.5: Calculation time per time step with the use of MPI. Dashed line represents theoretical limit.

7.3.3 Grid refinement

Following our theoretical discussion of the local grid refinement in Chapters 3-5 and, partly, 6, below we present results of the computation based on the *local uniform grid refinement technique* . We can combine local uniform grid refinement with parallel solution of the underlying coarse and fine grid problems. This allows to decrease computational time and to increase the size (in the sense of the number of the grid points) of the problem of interest. In this case Algorithm 3.3 looks like the one sketched in Figure 7.6

In Figure 7.7 we plot wall clock calculation time with and without local grid refinement. We take just one processor and would like to get a certain resolution in the area of interest. Within the developed framework this can be achieved with two different techniques, that is, with or without local grid refinement. In the first case we just use a global grid which coves the whole computational domain and has a predefined space step size. In the second approach we use two grids, the coarse and the fine, such that the space fine grid size is equal to the space step size in the first approach. We perform just one step in time, say from t^n to t^{n+1} with time step $\Delta t = t^{n+1} - t^n$. This is the time step used both in case of no grid refinement and in case of grid refinement. In real calculations we have different time steps for the coarse and fine grids, which leads to further time savings compared to the results presented below, but here we keep it the same for simplicity. In summary, for this comparison we have to solve:

- In case of no refinement, solve the problem with predefined time step Δt and space

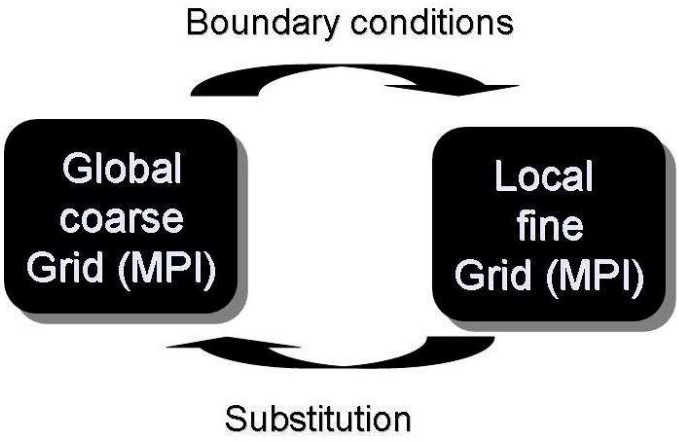


Figure 7.6: Local grid refinement and parallelization strategy.

step size (assuming that space step sizes in all three directions are equal) h.

- In case of local uniform grid refinement in one direction: one coarse solution with time step Δt and space step size $H = \sigma h$, where $\sigma > 1$ is the refinement factor (in the case presented in Figure 7.7, $\sigma = 4$) and one fine grid solution (with refinement in just one direction) space and time steps equal to Δt and h, respectively.

In Figure 7.7 we see two lines, the shorter one represents timings for local grid refinement in x-direction and the longer one represents timings for the equivalent uniform grid. If we look at the point marked by "A" in Figure 7.7, this represents the calculation time for local grid refinement with two $64 \times 64 \times 32$ grids. With a refinement factor $\sigma = 4$ this corresponds to an equivalent uniform grid of size $256 \times 64 \times 32$, calculation time for which one can find marked by "C". The same holds for point marked by "B" and "D" - these are calculation times for two $128 \times 64 \times 32$ for local grid refinement versus one $512 \times 64 \times 32$ equivalent uniform grid respectively. As one can see from Figure 7.7 we get substantial time savings while using local grid refinement.

At the next stage we address the accuracy of the local uniform grid refinement method versus the equivalent uniform grid. First we consider possible reflections from the boundaries in case of local grid refinement. From the numerical experiments we have, we conclude that in case of first order interpolation technique for transferring boundary values from coarse grid to fine grid and with restrictions on fine space and time steps coming from (7.29) we do not see any problems of this type. Some typical results one can find in Figure 7.8. In this figure we have already a developed flow after approximately 2000 time steps. We used the same time step both for coarse and fine grids.

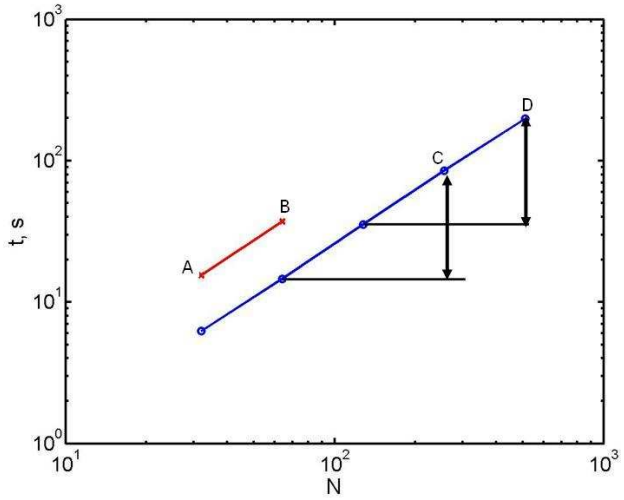


Figure 7.7: Wall clock calculation time with and without local grid refinement.

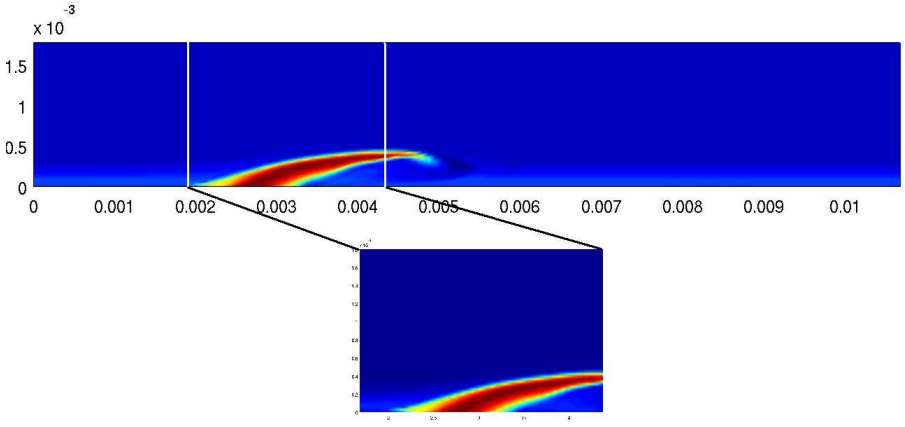
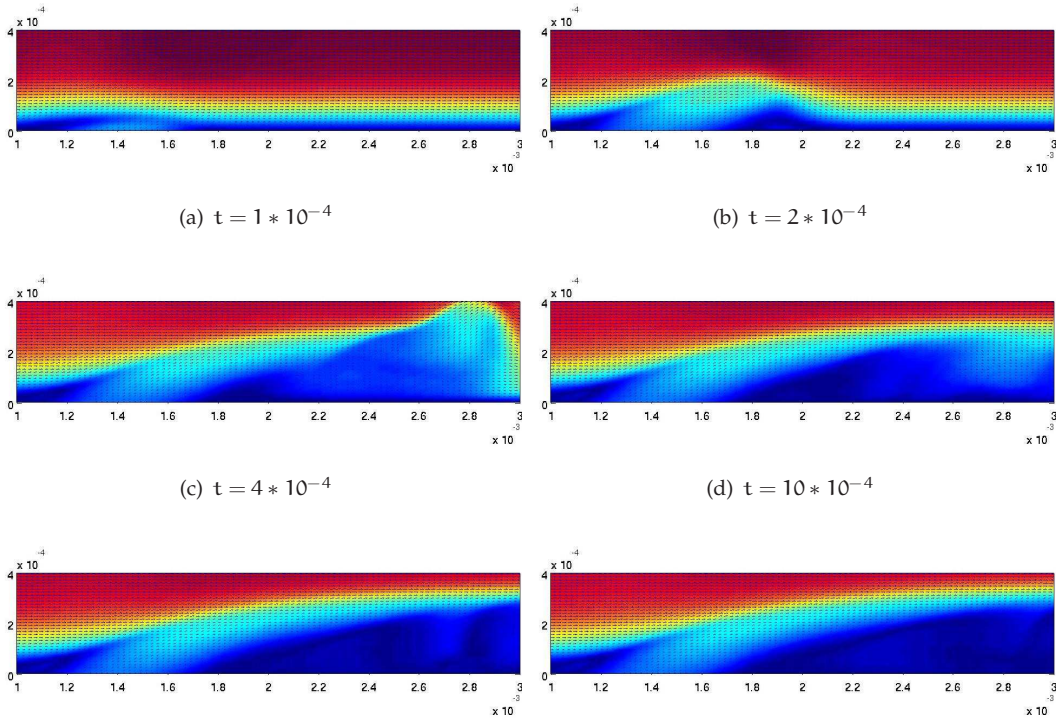


Figure 7.8: Combination of coarse and fine grid calculations.

In Figure 7.9 one can see fine grid solution at different times. We plot velocity vectors in the center plane and mark with color velocity magnitude.



(a) $t = 1 * 10^{-4}$ (b) $t = 2 * 10^{-4}$

(c) $t = 4 * 10^{-4}$ (d) $t = 10 * 10^{-4}$

(e) $t = 15 * 10^{-4}$ (f) $t = 20 * 10^{-4}$

Figure 7.9: Fine grid solution on different times. We plot velocity vectors in the center plane. Color represents velocity magnitude.

A small comparison of accuracy results for uniform grid refinement method and equivalent uniform grid one can find in Table 7.2. We compared velocity values in one calculation point, which is situated down the stream behind the cooling nozzle. ($x = 2.5 \cdot 10^{-3}$, $z = 3 \cdot 10^{-4}$).

7.4 Numerical results

Here we present some numerical results from calculations of the film cooling process. The experimental part and quite extensive description of the physics of the process one can find in [32].

We typically use grids of the size $128 \times 64 \times 32$, although experiments with much finer grids (up to $512 \times 256 \times 128$) were also performed. We normally use parallelization

Value	GR $64 \times 64 \times 32$	UG $256 \times 64 \times 32$
V_x	130.56	129.23
V_z	0.1	0.15

Table 7.2: Comparison between equivalent uniform grid and local grid refinement results. "UG" stands for equivalent uniform grid and "GR" stands for local grid refinement.

for calculations, but in the present state it is limited to one direction only (we split the domain in vertical direction). Typical size of the domain is $0.02 \times 0.01 \times 0.01$ m and time steps can be derived from (7.29).

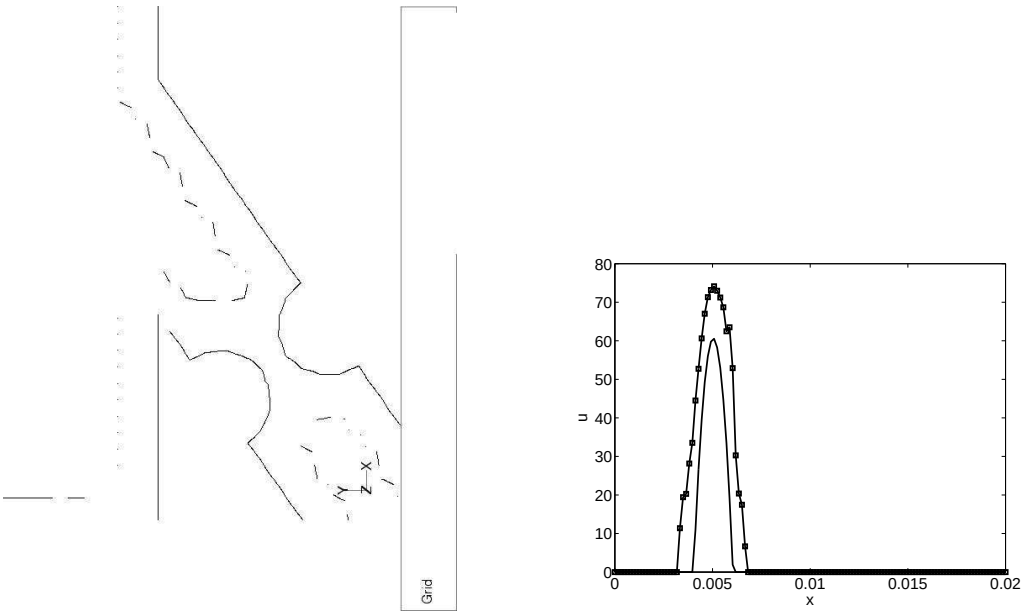
First we present some results from calculations for different geometries. We compare the so-called simple approach (namely just a prescribed parabolic profile; it is marked by a solid line in the pictures) with results obtained by simulation with the unstructured grid solver. In the simple approach we impose the prescribed velocity profile just at the plane where the cooling nozzle intersects the bottom plane of the computational domain. In the realistic approach we use the domain decomposition technique described in Section 7.3 to get the profile of the cooling jet. In fact the simple approach assumes that the main flow has no influence on the flow in the cooling nozzle and therefore we can use just some kind of inclined Poiseuille profile.

It is important to note that for all geometries under consideration we see a relatively strong presence of the third (z) velocity component, which in most cases is symmetric around the center plane. This velocity component causes circulation of the flow inside the cooling jet at the inflow plane and is caused by the interaction with the main flow. This was also observed experimentally in [32].

The second thing we observe while comparing results in Figures 7.10-7.13 is that with the simple approach we underestimate the velocity in x and y directions. This is due to the fact that we have a compressible flow and presence of inaccuracies may turn a straight tube in some analogous to a Laval nozzle (a Laval nozzle is a converging-diverging nozzle, which under certain conditions allows to achieve supersonic flow) due to inaccuracies flow in the cooling nozzle may accelerate.

Concerning the position of the inaccuracy, during the numerical experiments we observe that the deeper the inaccuracy is within tube, the more close the outflow profile is to the theoretical one. The closer the inaccuracy is to the nozzle's exit, the more influence it has on the flow. This influence is not always negative. In some cases it might have some positive effect. Each of these cases should be studied separately, so if we consider the predictability of final results, the main conclusion is that we should keep the inaccuracy deep inside the cooling nozzle. The influence of the position can be neglected for low velocity ratios (less than 0.25) [32].

The next point of interest from a practical point of view is to investigate the influence of the inaccuracy shape on the flow characteristics. From this perspective we studied



(a) Sketch of the domain

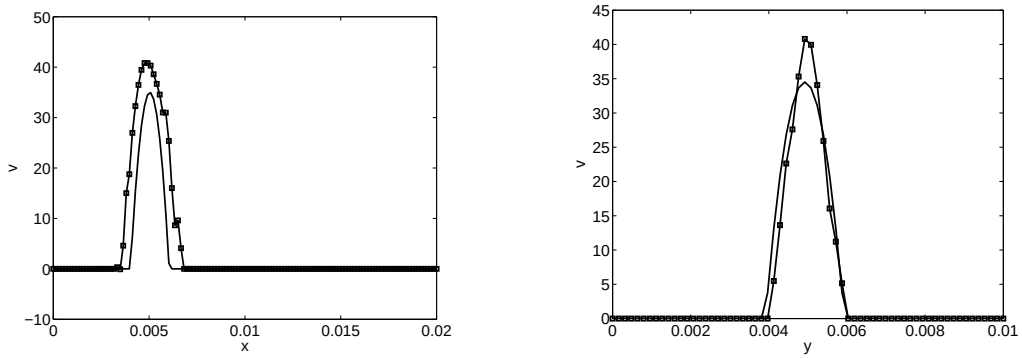
(b) u along x-direction

Figure 7.10: Sketch of the computational domain and example of an input profile. Velocity ratio is equal to 0.5. Single line is the simple approach, squares denote the unstructured grid solver results.

three types of inaccuracies, which have a cross sectional shape of a half circle, triangle and parallelepiped (see Figures 7.10, 7.12). For our numerical experiments we fix the position of the inaccuracy within the tube as well as the so-called *blockage area* (we define blockage as the ratio between the area of cross cut of nozzle with and without inaccuracy) and we only change the shape. Based on the comparison of results for different shapes of inaccuracies, we can conclude that the shape has a minor influence on the flow characteristics.

The final point of interest is the influence of the size of the imperfection. As it was mentioned before, this is mostly a question of theoretical interest due to the fact that in real production all the nozzles are checked to be within prescribed tolerance and large inaccuracies are impossible. From a practical point of view we can speak of blockages up to 30% at most. It was shown in [32] experimentally and our numerical results agree with it, that blockage has a very strong influence on the film cooling at small and moderate velocity ratios (around 0.5).

Next we analyze the flow structure itself via numerical experiments. Here we just show how numerical simulations may enrich the experimental part. We start with a series of computations of the "start up" phase of a flow. From an application point of view this is quite an interesting phase, with not that many information available. Most of the researches concentrate on the quasi-stationary part, when the flow is fully developed.

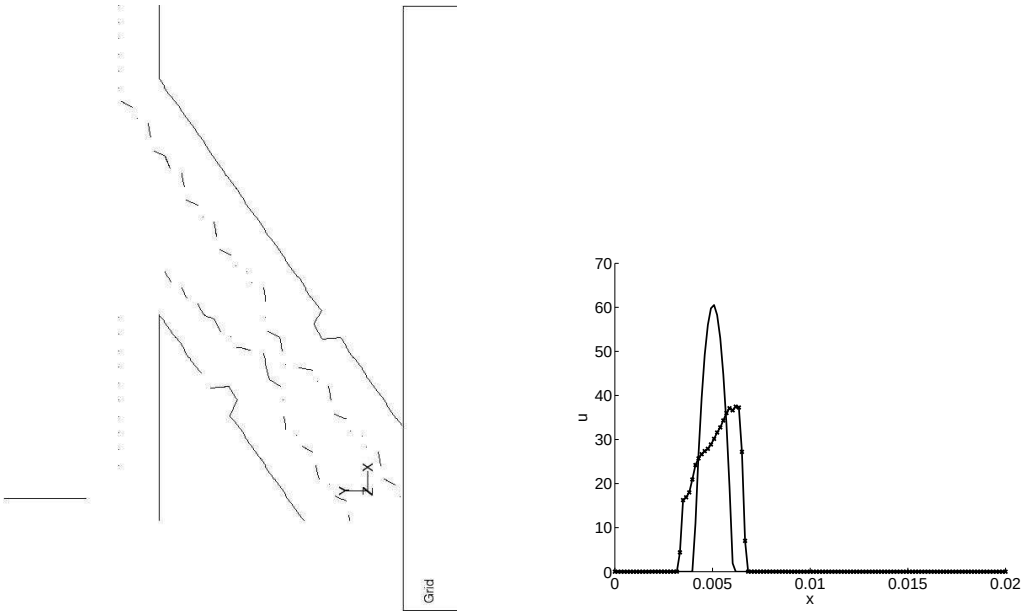


(a) v along x-direction (b) v along y-direction

Figure 7.11: Example of an input profile. Velocity ratio is equal to 0.5. Single line is the simple approach, squares denote the unstructured grid solver results

However, during the start up phase we may see the biggest changes in the flow parameters, this in turn could lead to higher loads (both thermal and mechanical) on the blade. In Figures 7.14 one can see the evolution of the flow at the "start-up" phase.

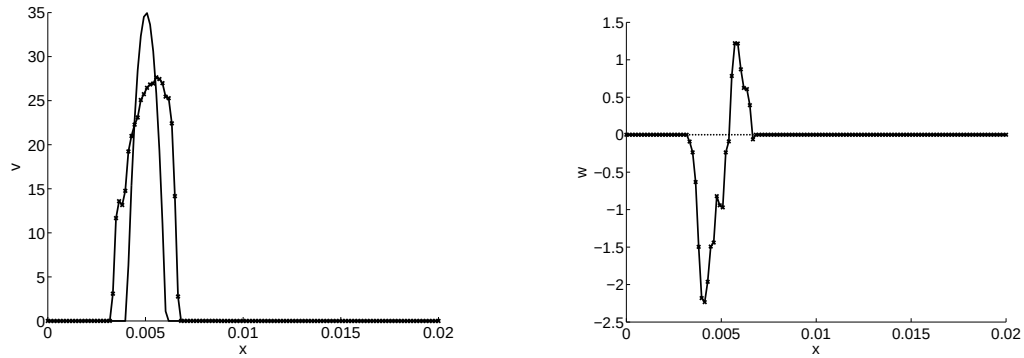
Once we reach fully developed flow, it is interesting to take a look at the flow structure. If we fix the moment in time, then we can just go through different planes in the computational domain in order to visualize the flow structures. In Figure 7.15 one can see a sequence of cross cuts through the plane, perpendicular to the main flow direction. One can see that the flow has quite a difficult structure, with vortices acting in all directions. A detailed experimental exploration one can find in [32], here we just show typical computational results. The main advantage we have with numerical computations compared to experiments is that the numerical approach gives much more information and flexibility compared to the experimental one. We have our data in the whole domain of interest, whereas in experiments we get data only in a small part of the domain (see Figure 7.15).



(a) Sketch of the domain

(b) u along x-direction

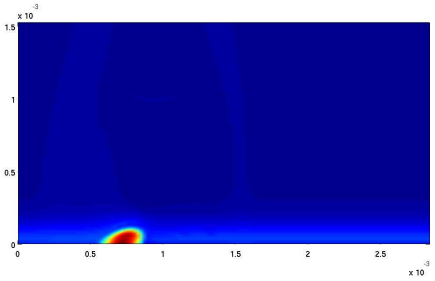
Figure 7.12: Sketch of the computational domain and example of an input profile. Velocity ratio is equal to 0.5. Single line is the simple approach, squares denote the unstructured grid solver results



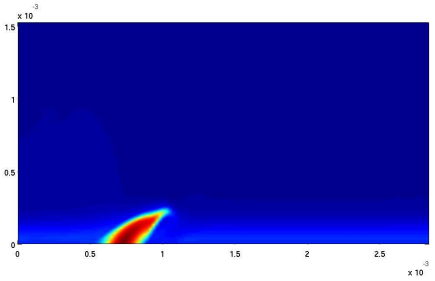
(a) v along x-direction

(b) w along x-direction

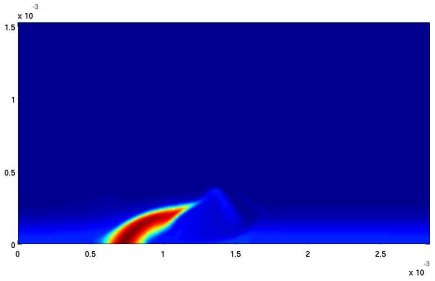
Figure 7.13: Example of an input profile. Velocity ratio is equal to 0.5. Single line is the simple approach, squares denote the unstructured grid solver results



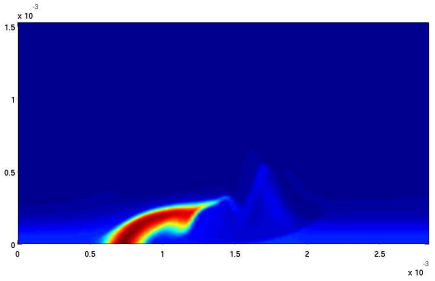
(a)



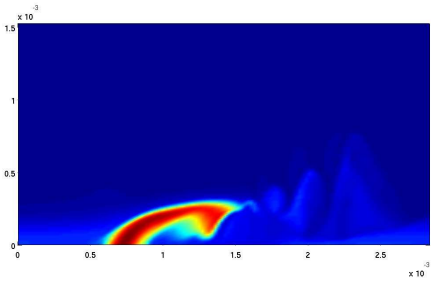
(b)



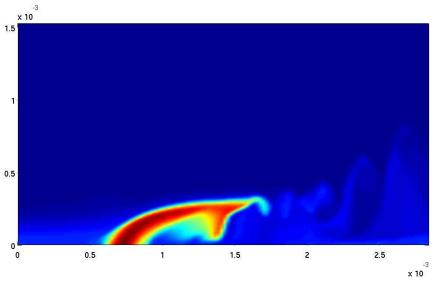
(c)



(d)

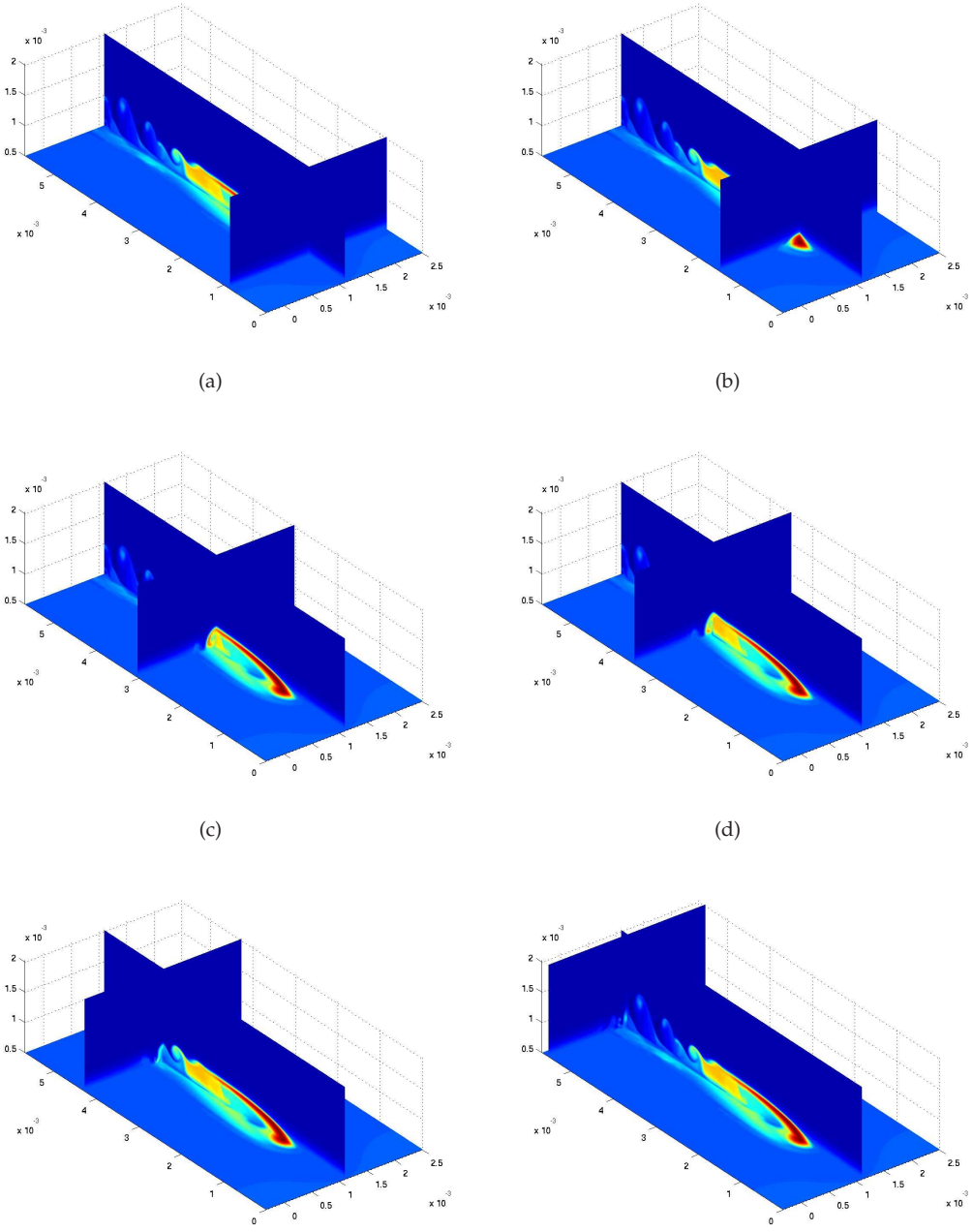


(e)



(f)

Figure 7.14: Time evolution of temperature distribution in the center plane of the computational domain.



(a) (b) (c) (d) (e) (f)

Figure 7.15: Temperature distribution in the computational domain.

Chapter 8

Conclusions and recommendations

In this thesis we have studied local grid refinement methods and their application to flow problems. Two methods were investigated: local uniform grid refinement (LUGR) and local defect correction (LDC). LDC is an iterative method for solving pure boundary value or initial-boundary value problems on composite grids based on using simple data structures and simple discretization stencils on uniform or tensor-product grids. Fast solution techniques exist for solving the system of equations resulting from discretization on a structured grid. The discretization on the composite grid is implicitly given by the LDC iteration. Numerical experiments and analytical derivations illustrate the fast convergence of the method. The standard method has been combined with high order finite difference schemes. The domain decomposition algorithm provides a natural way for parallelization and enables the usage of many small tensor-product grids rather than a single large unstructured grid. It has been shown that this may strongly reduce memory usage.

We have analyzed the convergence behavior of the LDC method for the convection-diffusion equation. We are able to theoretically estimate the convergence rate for different combination of fine and coarse grid discretization schemes. The following were analyzed: central differences on both grids, upwind on both grids, upwind for the coarse grid and central differences for the fine grid. We managed to show that the convergence speed is a function of the coarse grid step size H and in case of the upwind scheme we have $O(H)$ convergence, while in case of central difference it increases to $O(H^2)$. Numerical experiments support the theoretical estimation. We also proposed a way of transforming a pure convection problem into a boundary value problem by introduction of a small diffusive term and an artificial boundary condition. Another finding is that in case of a high Péclet number and an unstable scheme (like central differences) we still can get convergence of the composite solution.

We have combined the standard LDC method with high order finite differences. There is a natural way to combine high order schemes and LDC for one-dimensional problems, which after certain modifications is applicable for two- and three-dimensional problems. Numerical results prove high accuracy and fast convergence of the proposed method.

We made a review of boundary conditions for compressible flows in Chapter 6. Since we would like to use local grid refinement for such type of problems, we studied the model problem of an acoustic pulse spreading in Chapter 6. For this model problem we introduced local grid refinement and made a series of tests in order to see if artificial boundary conditions introduced for the fine grid cause any reflections of the acoustic waves.

We numerically studied film cooling problem and proposed a way to have a domain decomposition in order to use a proper boundary condition for the cooling jet. This domain decomposition allows to combine a structured DNS flow solver for the problem of interest with an unstructured solver for the flow in the cooling nozzle. On the other side, we implemented local grid refinement for the flow problem, which gives some time saving. So far grid refinement was used only in stream-wise direction. In this case an existing DNS solver on tensor-product grids was used as a black box solver. However, the proposed approach is solver independent and may be combined with different solvers. The existing tensor-product grid solver has been modified in order to run on a cluster, although so far we made parallelization only in one (vertical) direction.

In this thesis we just described a numerical approach to solve the film cooling problem without giving a detailed analysis of the flow. In other words we were more interested in designing a proper tool than in extensive and comprehensive numerical experiments with the help of the designed tool. This is partly due to the lack of time. On the next stage the approach described in this work should be used for a detailed series of calculations, which in turn should be compared with experimental results available. Another interesting area of research is to study the interaction of the cooling jets from several nozzles. In the present work we assumed periodic boundary conditions, which means an infinite row of cooling nozzles, which is not the case in reality. Also we concentrated on the flow pattern in the neighborhood of each individual nozzle and we were not interested in things far down the stream, where in reality jets from different nozzles start to interact.

For optimization of the solution method the following things could be interesting

- Parallelization in all three directions. This will lead (especially in stream-wise direction) to substantial time savings and/or the possibility to increase the size of the problem (number of grid points). However, there is a difficulty of tracking the position of the cooling hole, since if we implement parallelization in all three directions, because of different boundary conditions we have to trace different computational blocks independently.

- Grid refinement in all three directions. So far we used grid refinement only in one

direction.

- Nesting grids. This could give further insight into flow structure. At the moment we cannot use large refinement factors, however use of nested grid will allow to go to almost any level of detail of the flow structure.

Bibliography

[1] N. Adam and K. Shariff. A high-resolution hybrid compact-ENO scheme for shock-turbulence interaction problems. *J. Comput. Phys*, 127, 1996.

[2] N. A. Adams and L. Kleiser. Subharmonic transition to turbulence in a flat-plate boundary layer at Mach number 4.5. *J. Fluid Mech.*, 317:301, 1996.

[3] M. J. H. Anthonissen. *Local Defect Correction Techniques: Analysis and Application to Combustion*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2001.

[4] M. J. H. Anthonissen, D. Hömberg, and W. Weiss. Real-time control of surface remelting. In A. di. Bucchianico, R. M. M. Mattheij, and M. A. Peletier, editors, *Progress in Industrial Mathematics at ECMI 2004*, pages 356–360, Berlin, 2006. Springer.

[5] M. J. H. Anthonissen, R. M. M. Mattheij, and J. H. M. ten Thije Boonkkamp. Convergence analysis of the local defect correction method for diffusion equations. *Numerische Mathematik*, 95(3):401–425, 2003.

[6] M. J. H. Anthonissen, B. van 't Hof, and A. A. Reusken. A finite volume scheme for solving elliptic boundary value problems on composite grids. *Computing*, 61:285–305, 1998.

[7] A. Azzi and D. Lakehal. Perspectives in modeling film cooling of turbine blades by transcending conventional two-equation turbulence models. *ASME J. Turbomach.*, 124:472–484, 2002.

[8] B. A. V. Bennett and M. D. Smooke. Local rectangular refinement with application to nonreacting and reacting fluid flow problems. *J. Comput. Phys.*, 151:684–727, 1999.

[9] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[10] G. Blaisdell, N. Mansour, and W. Reynolds. Compressibility effects on the growth and structure of homogeneous turbulent shear flow. *J. Fluid Mech*, 256:443–485, 1993.

[11] D. E. Bohn, V. Becker, K. Kusterer, S. Ardey, and L. Fottner. The influence of slot injection and shower-head injection on the 3D flow field of a film-cooled turbine blade under consideration of side-wall effects. *AIAA Paper*, 97-7162, 1997.

[12] M. Carpenter, D. Gottlieb, and S. Abarbanel. Stability of numerical boundary treatment for compact high-order finite-difference schemes. *J. Comput. Phys*, 108:272–295, 1993.

[13] I. Chrisite. Upwind compact finite difference schemes. *J. Comput. Phys*, 59:353–368, 1985.

[14] G. Coleman, J. Kim, and R. Moser. A numerical study of turbulent supersonic isothermal-wall channel flow. *J. Fluid Mech*, 305:159–183, 1995.

[15] L. Collatz. *The Numerical Treatment of Differential Equations*. Springer, Berlin, 1961.

[16] H. C. de Lange. Split time-integration for low Mach number compressible flows. *Communications in Numerical Methods in Engineering*, 20:501–509, 2004.

[17] G. Erlebacher, M. Hussaini, H. Kreiss, and S. Sarkar. The analysis and simulation of compressible turbulence. *Theor. Comput. Fluid Dyn*, 2:73–95, 1990.

[18] D. J. Ferguson, K. D. Walters, and J. H. Leylek. Performance of turbulence models and near-wall treatments in discrete jet film cooling simulations. *ASME Paper*, 98-GT-438, 1998.

[19] P. J. J. Ferket. *Solving Boundary Value Problems on Composite Grids with an Application to Combustion*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1996.

[20] P. J. J. Ferket and A. A. Reusken. Further analysis of the local defect correction method. *Computing*, 56:117–139, 1996.

[21] P. J. J. Ferket and A. A. Reusken. A finite difference discretization method on composite grids. *Computing*, 56:343–369, 1996.

[22] V. K. Garg and R. S. Abhari. Comparison of predicted and experimental Nusselt number for a film cooled rotating blade. *Int. J. Heat Fluid Flow*, 18:452–460, 1998.

[23] V. K. Garg and A. A. Ameri. Comparison of two-equation turbulence models for prediction of heat transfer on film-cooled turbine blades,. *Numer. Heat Transfer, Part A*, 31:347–371, 1997.

[24] M. Graziadei, R. M. M. Mattheij, and J. H. M. ten Thije Boonkkamp. Local defect correction with slanting grids. *Numerical Methods for Partial Differential Equations*, 20:1–17, 2001.

[25] W. Hackbusch. Local defect correction and domain decomposition techniques. In K. Böhmer and H. J. Stetter, editors, *Defect Correction Methods. Theory and Applications, Computing, Suppl. 5*, pages 89–113, Wien, New York, 1984. Springer.

[26] W. Hackbush. *Elliptic Differential Equations. Theory and Numerical Treatment*. Springer, Berlin, 1992.

[27] G. W. Hedstrom. Nonreflecting boundary conditions for nonlinear hyperbolic systems. *J. Comput. Phys*, 30:222–237, 1979.

[28] C. Hirch. *Numerical Computation of Internal and External Flows. Vol. 2. Computational Methods for Inviscid and Viscous Flows.* Wiley, New York, 1990.

[29] R. S. Hirsch. Higher order accurate difference solutions of fluid mechanics problems by a compact differencing scheme. *J. Comput. Phys*, 19:20–109, 1975.

[30] R. Hixon, S.-H. Shih, and R. R. Mankbadi. Evoluation of boundary conditions for computational aeroacoustics. *AIAA Journal*, 33:2006–2012, 1995.

[31] S. Irmisch. Simulation of film-cooling aerodynamics with a 2-D Navier-Stokes solver using unstructured grids. *ASME Paper*, 95-GT-024, 1995.

[32] M. B. Jovanović. *Film Cooling through Imperfect Holes*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2006.

[33] Z. Kopal. *Numerical Analysis*. John Wiley, New York, 1961.

[34] E. Krause. Mehrstellen Verfahren zur Integration der Grenzschichtgleichungen. *DLR Mitteilungen*, 71:109–40, 1971.

[35] D. Lakehal, G. Theodoridis, and W. Rodi. Computation of film cooling of a flat plate by lateral injection from a row of holes. *Int. J. Heat Fluid Flow*, 19:418–430, 1998.

[36] D. Lakehal, G. Theodoridis, and W. Rodi. Three dimensional flow and heat transfer calculations of film cooling at the leading edge of a symmetrical turbine blade model. *Int. J. Heat Fluid Flow,*, 22:113–122, 2001.

[37] S. Lee, S. K. Lele, and P. Moin. Eddy-shocklets in decaying compressible turbulence. *Phys. Fluids A*, 3:657–664, 1991.

[38] S. Lele. Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys*, 103:16–42, 1991.

[39] S. Lele. Computational aeroacoustics: a review. *AIAA Paper*, 970018, 1997.

[40] M. Lesieur and O. Metais. New trends in large eddy simulations of turbulence. *Annu. Rev. Fluid Mech.*, 28:45–82, 1996.

[41] J. H. Leylek and R. D. Zirkle. Discrete-jet film cooling: A comparison of computational results with experimentss. *ASME J. Turbomach.*, 116:358–368, 1994.

[42] K. Mahesh. A family of high order difference schemes with good spectral resolution. *J. Comput. Phys*, 145:332–358, 1998.

[43] S. McCormick. Fast adaptive composite grid (FAC) methods: Theory for the variational case. In K. Böhmer and H. J. Stetter, editors, *Defect Correction Methods. Theory and Applications, Computing, Suppl. 5*, pages 115–121, Wien, New York, 1984. Springer.

[44] R. Minero. *Local Defect Correction for Time-Dependent Problems*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2006.

[45] R. Minero, M. J. H. Anthonissen, and R. M. M. Mattheij. A local defect correction technique for time-dependent problems. *Numerical Methods for Partial Differential Equations*, 22:128–144, 2006.

[46] R. Minero, M. J. H. Anthonissen, and R. M. M. Mattheij. Solving parabolic problems using local defect correction in combination with the finite volume method. *Numerical Methods for Partial Differential Equations*, 22:1149 – 1172, 2006.

[47] P. Moin. Progress in large eddy simulation of turbulent flows. *AIAA Paper*, 970749, 1997.

[48] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *Annu. Rev. Fluid Mech.*, 30:539–578, 1998.

[49] V. Nefedov. *Numerical Analysis of Viscous Flow Using Composite Grids with Application to Glass Furnaces*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2001.

[50] V. Nefedov and R. M. M. Mattheij. Local defect correction with different grid types. *Numerical Methods for Partial Differential Equations*, 18:454–468, 2002.

[51] M. J. Noot. *Numerical analysis of turbine blade cooling ducts*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1997.

[52] J. Oliger. Approximate methods for atmosperic and oceanographic circulation problems. In *Lecture Notes in Physics, 91*, pages 171–184, Wien, New York, 1979. Springer.

[53] A. T. Patera. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *J. Comput. Phys*, 54:468, 1984.

[54] R. Peyret. Introduction to high-order approximation methods for computational fluid dynamics. In R. Peyret, editor, *In Advanced Turbulent flow calculations*, pages 1–45, Berlin, 2000. Springer.

[55] T. J. Poinsot and S. K. Lele. Boundary conditions for direct simulations of compressible viscous flows. *J. Comput. Phys*, 101:104–129, 1992.

[56] C. D. Pruett, T. A. Zang, C. L. Chang, and M. H. Carpenter. Spatial direct numerical simulation of high-speed boundary-layer flows. Part I. Algorithmic considerations and validation. *Theoret. Comput. Fluid Dynam.*, 49:301, 1995.

[57] M. Rai, T. Gatski, and G. Erlebacher. Direct simulation of spatially evolving compressible turbulent boundary layers. *AIAA Paper*, 95:0583, 1995.

[58] M. M. Rai and P. Moin. Direct numerical simulation of transition and turbulence in a spatially evolving boundary layer. *J. Comput. Phys*, 109:169, 1993.

[59] U. Rist and H. Fasel. Direct numerical simulation of controlled transition in flat-plate boundary layer. *J. Fluid Mech.*, 298:211, 1995.

[60] W. Rodi. Experience with two-layer models combining the $k - \epsilon$ model with a one-equation model near the wall. *AIAA Paper*, 91-0216, 1991.

[61] S. G. Rubin and R. A. Graves. Viscous flow solutions with a cubic soline approximation. *Computer & Fluids*, 3:1–36, 1975.

[62] S. G. Rubin and P. K. Khosla. Polynomial interpolation method for viscous flow calculations. *J. Comput. Phys*, 24:217–46, 1976.

[63] D. Rudy and J. Strikwerda. A nonreflecting outflow boundary condition for subsonic Navier–Stokes equations. *J. Comput. Phys*, 36:55–70, 1980.

[64] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, 1996.

[65] S. Sarkar, G. Erlebacher, and M. Hussaini. Direct simulation of compressible turbulence in a shear flow. *Theor. Comput. Fluid Dyn.*, 2:291–305, 1991.

[66] B. Sjögreen. High order centered difference methods for the compressible Navier-Stokes equations. *J. Comput. Phys.*, 117:67–75, 1995.

[67] C. Speziale. Analytical methods for the development of Reynolds-stress closures in turbulence. *Annu. Rev. Fluid Mech.*, 23:107–157, 1991.

[68] G. Strang. *Linear Algebra and Its Applications*. Harcourt Brace Jovanovich, San Diego, 1988.

[69] J. C. Strikwerda. Initial boundary value problem for incompletely parabolic systems. *Communications on Pure and Applied Mathematics*, 30:797–822, 1977.

[70] C. Tam. Computational aeroacoustics: issues and methods. *AIAA Paper*, 950677, 1995.

[71] C. Tam and J. Webb. Dispersion-relation-preserving finite differences schemes for computational acoustics. *J. Comput. Phys*, 107:262–281, 1993.

[72] G. Theodoridis, D. Lakehal, and W. Rodi. 3D calculations of the flow field around a turbine blade with film cooling injection near the leading edge. *Flow, Turbul. Combust.*, 66:57–83, 2001.

[73] G. Theodoridis and W. Rodi. Calculation of the flow around a high-pressure turbine blade with cooling-jet injection from slots at the leading edge. *Flow, Turbul. Combust.*, 62:89–110, 1999.

[74] K. W. Thompson. Time dependent boundary conditions for hyperbolic systems. *J. Comput. Phys*, 68:1–24, 1987.

[75] A. Tolstykh and M. Lipavskii. On performance of methods with third- and fifth-order compact upwind differencing. *J. Comput. Phys*, 140:205–232, 1998.

[76] R. A. Trompert and J. G. Verwer. Analysis of the implicit Euler local uniform grid refinement. Technical Report NM-R9011, CWI, Amsterdam, 1990.

[77] R. A. Trompert and J. G. Verwer. Runge-Kutta methods and local uniform grid refinement. Technical Report NM-R9022, CWI, Amsterdam, 1990.

[78] A. van der Sluis. Equilibration and pivoting in linear algebraic systems. In *Proceedings of the IFIP Congress*, pages 127–129, 1968.

[79] B. van 't Hof. *Numerical Aspects of Laminar Flame Simulation*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1998.

[80] J. C. J. Verhoeven. *Modelling laser percussion drilling*. PhD thesis, Eindhoven University of Technology, Eindhoven, 2004.

[81] J. G. Verwer and R. A. Trompert. Analysis of local uniform grid refinement. Technical Report NM-R9211, CWI, Amsterdam, 1992.

[82] A. Vreman, N. Sandham, and K. Luo. Compressible mixing layer growth rate and turbulence characteristics. *J. Fluid Mech*, 320:235–258, 1996.

[83] K. D. Walters and J. H. Leylek. A detailed analysis of film-cooling physics, part 1: Streamwise injection with cylindrical holes. *ASME Paper*, 97-GT-269, 1994.

[84] X. Zhong. High-order finite-difference schemes for numerical simulation of hypersonic boundary-layer transition. *J. Comput. Phys*, 144:662–709, 1998.

[85] D. W. Zingg, H. Lomax, and H. Jurgens. High-accuracy finite-difference schemes for linear wave propagation. *SIAM J. Sci. Comput.*, 17:328, 1995.

Index

Summary

One of the major problems in enhancing the specific work output and efficiency in gas turbines is the maximum possible value of the turbine inlet temperature due to blade material properties. To increase this maximum, turbine blades need to be cooled (internal or external), which is usually done by compressor air. Based on its high cooling efficiency, film cooling is one of the major cooling techniques used, especially for the hottest blades. In film cooling cold air is injected into the boundary layer through small nozzles in the blade surface. Impingement of the jets into the (laminar) boundary layer flow is essentially three-dimensional. The collision of the laminar jet with the boundary layer flow produces a local turbulent shear layer and changes the local heat transfer to the blade (when poorly constructed it may even increase the local heat transfer).

In this project we have studied local grid refinement methods and their application to flow problems in general and to air film cooling in particular. Local defect correction (LDC) is an iterative method for solving pure boundary value or initial-boundary value problems on composite grids. It is based on using simple data structures and simple discretization stencils on uniform or tensor-product grids. Fast solution techniques exist for solving the system of equations resulting from discretization on a structured grid. We have combined the standard LDC method with high order finite differences by using a new strategy of defect calculation. Numerical results prove high accuracy and fast convergence of the proposed method.

We made a review of boundary conditions for compressible flows. Since we would like to use local grid refinement for such flow problems, we studied the spreading of an acoustic pulse. For this model problem we introduced local grid refinement and made a series of tests in order to see if the artificial boundary conditions introduced for the local fine grid cause any reflections of the acoustic waves.

The numerical techniques developed have been used to study film cooling. Because this problem concerns the interaction between a main flow and a jet, we also propose a domain decomposition algorithm in order to supply proper boundary conditions for the cooling jet. This domain decomposition combines a structured DNS flow solver for the problem of interest with an unstructured solver for the flow in the cooling nozzle. Additionally we implemented local grid refinement for the flow problem to save computational costs.

Samenvatting

Een van de belangrijkste problemen bij het verbeteren van het specifiek vermogen en de efficiëntie van gasturbines is hoe hoog de temperatuur bij de ingang van de turbine mag zijn met het oog op materiaaleigenschappen. Om deze waarde te verhogen moeten turbineschoepen (intern of extern) gekoeld worden. Dit wordt gewoonlijk gedaan door samengedrukte lucht. Filmkoeling is een van de belangrijkste koeltechnieken die wordt gebruikt vanwege de hoge koelefficiëntie, met name voor zeer hete schoepen. Bij filmkoeling wordt koude lucht via smalle gaten in het oppervlak van een schoep in de grenslaag geïnjecteerd. De instroom van de *jets* in de (laminaire) grenslaag is driedimensionaal. De botsing van de laminaire jet met de grenslaag produceert een lokaal turbulente afschuiflaag en verandert de lokale warmteoverdracht naar de schoep. Bij een slechte constructie kan de lokale warmteoverdracht zelfs toenemen.

In dit project hebben we een aantal methoden voor lokale roosterverfijning beschouwd en gekeken naar hun toepasbaarheid op stromingsproblemen in het algemeen en filmkoeling met lucht in het bijzonder. Lokale defectcorrectie (LDC) is een iteratieve methode voor het oplossen van randwaardeproblemen en beginrandwaardeproblemen op samengestelde roosters. De techniek berust op eenvoudige datastructuren en eenvoudige discretisatiestencils op uniforme of tensorproductroosters. Er zijn snelle technieken voor het bepalen van de oplossing van systemen van vergelijkingen die voortkomen uit de discretisatie op een gestructureerd rooster. Dankzij een nieuwe manier om het defect in de LDC-methode te berekenen zijn we er in geslaagd om LDC te combineren met zogenaamde hoge-orde eindige differentiemethoden. Numerieke resultaten tonen aan dat de ontwikkelde methode zowel een hoge nauwkeurigheid als een snelle convergentie geeft.

We hebben een overzicht gegeven van randvoorwaarden voor samendrukbare stromingen. Aangezien we roosterverfijning willen toepassen op dergelijke stromingsproblemen hebben we de verspreiding van een akoestische puls bestudeerd. Voor dit modelprobleem hebben we roosterverfijning toegepast en een serie experimenten uitgevoerd om te onderzoeken of de kunstmatige randvoorwaarden weerkaatsingen veroorzaken van de akoestische golven.

De ontwikkelde numerieke technieken zijn gebruikt om filmkoeling te analyseren. Omdat in dit probleem de interactie tussen een hoofdstroom en een jet van belang is, presen-

teren we ook een domeindecompositie algoritme om de juiste randvoorwaarden voor de (koelende) jet te kunnen opleggen. Deze domeindecompositie combineert een computerprogramma dat de stroming via DNS op een gestructureerd rooster oplost met een computerprogramma dat de stroming in het koelkanaal oplost op een ongestructureerd rooster. Tenslotte hebben we roosterverfijning geïmplementeerd voor het stromingsproblemen om rekenkosten te verlagen.

Acknowledgements

In the end of this thesis I would like to give my gratitude to all of the people that, in many different ways, have helped me to complete this work. I had doubts if to put this part at the beginning as preface or in the end as acknowledgements and finally settled with the last option. This has to do with the project flow - once you start, you know only a bit and only few people around, but as time passes you get in touch with more and more people around you, so only in the end you realize how many of those participated one way or another. First of all, I would like to thank my copromotor dr.ir. Martijn Anthonissen for all his help. I can spend lines and lines describing his contribution, but I put it short: without him this thesis would never be finished. Next I would like to thank my promotor prof.dr. Bob Mattheij, first of all, for accepting me as a master student back in 2001 and then for giving me the opportunity to continue my research as a PhD student in the CASA group. I would also like to mention all the people involved in this project: prof.dr.ir Antoon Steenhoven, dr.ir. Rick de Lange, dr.ir. Milenko Jovanovič. During my four years as a PhD student I have really enjoyed being a part of the CASA group. Therefore I would like to thank all the present and the past members of this group. Some of them, though, deserve a special word of thanks. During these years it has been a pleasure for me to share the office with Nico van der Aa and Pavel Kagan. I'm still amazed how nicely things can fit in Pavel's head and how efficient Nico can work. Thanks a lot to all of you, and also to Bas van der Linden, Paul de Haas and Enna van Dijk who have answered an enormous amount of small questions, especially during the first years of my PhD.

Fortunately being a PhD student I did not spend all of my time at the university. I was trying to spend as much time as I could with my friends, so here they are: Nazar Sushko, Anna Dabrovska, Ilia Malakhovsky and Veronika Malakhovskaya. I enjoyed a lot playing with Marfa Malakhovkaya (and was amazed by the amount of knowledge one has to have nowadays at the age of 4) and, recently, with Natalia Sushko. Although living abroad, I have not lost contacts with my friends in Russia (saying that I realize that half of them are no longer in Russia, but somewhere in Europe). I would like to take this opportunity to send my greetings to all of them, especially to Dmitry Goryntsev, Konstantin Kazin, Igor Puchkov, Vladimir and Natalia Bratov(a). During the last years of my PhD I traveled a lot mostly with the use of my old Classic Saab 900, so I would like to thank people who designed it for making such an insane car. My girlfriend Tatiana deserves special greetings for keeping me up and motivated and for just being in my

life. Finally, I would like to thank my father Anatoly and my mother Tamara: I know you have sacrificed a lot letting me go to another country and thanks a lot for your love, care and support!

Mikhail Sizov

Amsterdam, April 2007

Curriculum vitae

The author of this thesis was born April 6th, 1978, in Leningrad (now Saint Petersburg), Russia. He finished his preuniversity education at the Gymnasium 248 in Saint Petersburg in 1995. In September 1995 Mikhail started studying fluid dynamics at Baltic State Technical University, from which he graduated in 2001 with honors. His master's thesis, written under the supervision of prof.dr.ir. V.N. Emel'janov, was titled *Flow of fluid in channels of different type*. In parallel in 2000 Mikhail started to study at the Saint Petersburg Academy of Control Methods and in 2001 got a diploma of software developer.

After his studies at Baltic State Technical University, Mikhail decided to continue his studies abroad. He was selected for the Huygens Scholarship for studying in The Netherlands. From the variety of offered programs he chose Computational Science and Engineering masters's program at Technical University of Eindhoven. After first year of Computational Science and Engineering program Mikhail got an offer to continue his research at TU/e as Phd student in the Scientific Computing Group of prof.dr. R.M.M. Mattheij. In 2002 Mikhail got a master degree from TU/e. His research on air film cooling and local defect correction technique has lead to this thesis.

Since July 19, 2006, Mikhail works as a Risk Manager at Fortis Investments, being responsible for front office mathematical models validation.