Using local defect correction for laminar flame simulation

Using Local Defect Correction for Laminar Flame Simulation

Marialuce Graziadei

Printed by Eindhoven University Press

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Graziadei, Marialuce V.

Using Local Defect Correction for Laminar Flame Simulation
by Marialuce Valentina Graziadei. -
Eindhoven, Technische Universiteit Eindhoven, 2004.
Proefschrift. - ISBN 90-386-0962-0

NUR 919
Subject headings: boundary value problems; numerical methods /
elliptic differential equations; boundary value problems /
combustion; numerical methods
2000 Mathematics Subject Classification: 65N50, 65N55, 65N06, 80A25

Using Local Defect Correction for Laminar Flame Simulation

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College
voor Promoties in het openbaar te verdedigen
op donderdag 9 december 2004 om 16.00 uur

door

Marialuce Valentina Graziadei

geboren te Pagani, Italië

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. R.M.M. Mattheij
en
prof.dr. L.P.H. de Goey

Copromotor:
dr.ir. J.H.M. ten Thije Boonkkamp

Acknowledgements

Many people contributed, in different ways, to the realisation of the research work that resulted in this thesis. First, I would like to thank prof.dr. Robert Mattheij for the support, guidance and encouragement he offered me and for promoting the very nice and enjoyable atmosphere in the Scientific Computing Group. Special thanks also to dr.ir. Jan ten Thije Boonkkamp for the many fruitful discussions and advices, as well as for the thorough and time consuming revisions of the thesis manuscript that allowed to improve the quality of my work. I am also grateful to prof.dr.ir. Philip de Goey for his useful comments to the draft of this thesis.

In scientific computing, programming and coding skills are essential to the successful implementation of any numerical method. That is why the help and suggestions of dr.ir. Bas van der Linden have been extremely valuable and appreciated. Moreover, I would like to acknowledge dr.ir. Ronald Rook for the work performed on the Lamfla code.

Essential to research is also the possibility to work in a positive and lively environment, therefore I want to say thanks to all the Scientific Computing Group fellows and in particular to Bratislav Tasic, Irina Lioulina, Martijn Anthonissen, Wienand Drenth, Luiza Bondar. Special thanks are due to dr.ir. Paul de Haas and again to dr.ir. Bas van der Linden, whose room was equipped with an espresso coffee machine, an essential appliance for good working performance and for creating a very *gezellige* atmosphere.

Finally, I would like to mention the beloved ones. My parents Tina and Libero deserve all my gratitude for their love and encouragement. Some of the results of this thesis have been obtained while travelling between Eindhoven, Milan and London to reach a person that means a lot for me. The love, support and sense of humour of Maurizio made my life enjoyable and worth also in the difficult moments on the way to the end of this thesis.

Contents

CHAPTER 1

Introduction

1.1 Background

Combustion was the first source of energy for human kind. It provided early men and women with light, heat and, apart from changing radically their diet, it gave a definitive impulse to the construction of more sophisticated and effective tools. Despite the fact it witnessed and promoted the very early steps in technology, combustion still plays a very important role in our life as it supplies, both in a direct and an indirect way, 90% of the energy demand [25]. Fields of application are many, ranging from domestic appliances - combustion is primarily involved in heating and burning home devices - to industrial equipment. Electricity generating capabilities heavily depend on combustion, as do industrial energy needs. Moreover, combustion fuels are especially attractive when an energy vector is required: it is therefore not surprising that they occupy a very important position in the field of public and private transportation. Also at the very end of the product life we often find incineration, a socially and politically very controversial subject, to dispose of all sorts of waste and, sometimes, to recover energy from them. Finally, it is worth to recall that combustion is fruitfully used to flare toxic products coming from industrial applications.

The use of combustion in such a massive way implies the necessity to ensure that the processes related to it are performed in the most efficient way. Several reasons lead to consider this as a very high priority. First of all, the increase in energy demand, in conjunction with the depletion of the cheapest fossil reserves, requires strong efforts to minimise the waste of energy. Moreover, associated with combustion there is the important issue of environmental pollution. The ever more strict requirements of national and international authorities on allowed emissions are intended to put a limit on the inherent polluting products of combustion. Amongst them, we have carbon oxides, CO and CO_2, with the latter being the main responsible for the so-called *greenhouse effect*. Moreover, combustion occurring in air inevitably leads to the formation of nitrogen oxides, NO and NO_2, also generally indicated with NO_x. They constitute one of the main ingredients involved in the formation of *ground-level ozone* and, together with sulphur oxides,

Figure 1.1: Bunsen burners

contribute to formation of acid rains. In addition, partially burned or unburned hydrocarbons are detrimental to human and environmental health; they are, for instance, the main cause of smog. Finally, the last problem worth to be mentioned is the safety of combustion processes that has to be guaranteed for all applications, ranging from the domestic to the industrial ones. All these issues make the study of combustion a field of big importance to meet the requirements of society. In fact, the understanding of the underlying mechanisms is essential for controlling the combustion processes and to make them cleaner, safer and more efficient.

1.2 Combustion theory in a nutshell

While the basis of combustion relies on exothermic chemical reactions, the physical processes involved are mainly energy and mass transport, such as heat diffusion, chemical species diffusion, convection and radiation. Two main components give rise to a combustion reaction: the *fuel* and the *oxidizer*, in most cases air. Flames are classified on the basis of their main features. A first distinction is made between *premixed* and *nonpremixed* flames. While the former are characterised by the mixing of the fuel and the oxidizer before combustion takes place, in the latter such components are mixed and burned at the same time. Both *premixed* and *nonpremixed* are subdivided in *laminar* and *turbulent* flames, depending on the *laminar* or *turbulent* character of the fluid flow. Examples of *nonpremixed laminar* flames are candles and wood fire, while *nonpremixed turbulent* combustion takes place in aircraft turbine engines. Spark-ignited gasoline engines are, instead, typical applications of *turbulent premixed* combustion. In this thesis we focus on the study of *premixed laminar flames* as occurring, for instance, in a *Bunsen burner*, often used for domestic or laboratory applications, see Figure 1.1.

A very important parameter characterising the behaviour of premixed flames is the *stoichiometric ratio* φ. If the proportion between fuel and oxidizer is such that they are both completely consumed after combustion, then $\varphi = 1$ and the flame is said to be

stoichiometric!flame. If there is an excess of fuel with respect to the oxidizer the system is said *fuel-rich* and $\varphi > 1$. When the opposite occurs, then $\varphi < 1$ and the system is said to be *fuel-lean*. The possibility of controlling the composition of the mixture before burning is very important for combustion efficiency. In fact, a fuel-lean system leads to complete combustion which implies low CO emissions and a relatively low temperature. It is worth to remark that the lower the combustion temperature, the smaller is the production of NO_x. Too lean flames, on the other hand, are much more sensitive to stability problems. Too lean flames, on the other hand, are much more sensitive to stability problems.

The variety of phenomena involved and the large number of relevant variables (pressure, gaseous mixture composition, flow velocity, etc.) make the study of combustion a complex and interesting field. The fluid dynamics and the chemistry used to model it give rise to highly non-linear systems of equations that do not admit analytical solutions, unless in very special and simplified cases. Moreover, since combustion is caused by phenomena that occur at different time and length scales, those systems are usually very stiff. Combustion is therefore an interdisciplinary subject, that requires the combined efforts of chemistry, physics, fluid mechanics and applied mathematics. Research on numerical flame simulation has been directed, on the one hand, to the development of reduced combustion models that, while retaining the predictive capabilities of the more complex chemical networks, are able to decrease the complexity of the numerical work. On the other hand, research deals with the development of more sophisticated solution techniques aiming at solving the combustion equations efficiently.

Both research lines have quite a long history at Eindhoven University of Technology. In [36], a theory to describe one-dimensional flames and to compute the related combustion parameters is introduced. Moreover, in [36, 37], the authors present a numerical method to solve the flow field using a stream function-vorticity description. The grid is non-staggered and grid-refinement is used in regions where a given variable has high gradients. In [56, 57] laminar flames stabilised on a premixed flat flame burner are simulated, with detailed and reduced chemical models. The authors focus their study on the development of reduction techniques and apply the steady-state and partial equilibrium assumptions to eliminate a number of species. The corresponding partial differential equations are then replaced by algebraic relations. This way the dimension of the system of equations to be solved can be significantly reduced, with great advantages not only for computational memory requirements, but also for the possibility to consider more complex burner geometries. Comparisons are carried out between results obtained with reduced and detailed models. Those are then improved via a sensitivity analysis. Reduced chemical models are also treated in [17] and [18], with a particular emphasis on laminar flames. The starting point is to divide the reaction space into two subspaces: one containing the slow and the other the fast processes. The latter are then considered to be in a steady state. This research is mainly devoted to model NO_x formation in laminar methane/air flames, with the objective to perform detailed NO computations with reduced reaction mechanisms. A new reduction method is presented in [45]. Traditional reduction techniques are based on the *steady state assumption*, i.e. a large number of species react very fast and their chemistry is dominant with respect to other processes. Actually, this holds only at high temperatures, since at lower

temperatures the number of slow chemical processes increases and also convection and diffusion play an important role. The reduction technique presented by the author does not only takes chemical reactions into account, but the contributions of convection and diffusion processes as well.

The stabilisation of steady premixed laminar flames is influenced by several physical processes related to the configuration of the burners. These phenomena are studied in [40]. The author investigates problems as *quenching*, *flash back* and *blow-off* of two-dimensional methane/air flames.

The acoustic behaviour of burner-stabilised methane/air flames is investigated in [48]. The analysis takes several different length and time scales into account. The most important two are the acoustic length scale, which characterises sound waves, and the flame length scale, which is considered to be much smaller than the previous one.

Several numerical problems arising in laminar flame simulation are explained in [30] and new numerical techniques are introduced. First of all, an improved Thiart's scheme, see [58, 59], for discretisation of convection-diffusion-reaction equations is studied. This possesses the following three properties. First, the numerical fluxes are second order accurate for all flow conditions, ranging from diffusion to convection dominated flows. Secondly, no oscillations in the proximity of high gradients are produced for convection dominated flows. Finally, only a three point coupling for each spatial direction is used, see [31]. Moreover, in [30] the author discusses iterative solution methods. In fact, since laminar combustion problems occur at low Mach numbers, the *isobaric* or *combustion approximation* can be usually applied. Traditional *pressure correction* are reformulated in [30, 32] to deal with such a situation. In particular, the correction step is no longer based on the continuity equation but on the *expansion equation*.

Combustion problems are characterised by solutions that are very steep in a small part of the computational domain. Therefore, they appear suitable to be solved with local refinement techniques. In [2] the author studies several theoretical aspects of the *Local Defect Correction* (LDC) method. First, a modified formulation for finite volume discretisation is introduced. Its key feature is the possibility to keep the discrete conservation property, which usually does not hold for composite grid approximation and which is one of the most attractive characteristics of finite volume techniques. Then the author analyses the iterative behaviour of the LDC method and introduces a new expression for the iteration matrix. An upper bound for this is derived for a model problem that consists of the Poisson equation solved on the unit square. Finally, the LDC method is combined with the concepts of *domain decomposition, regridding, multi-level refinement* and *adaptivity*; the application to the numerical simulation of a Bunsen laminar flame is presented. In the following we will refer extensively to this work.

1.3 Outline of the thesis

In this thesis we investigate some numerical techniques for laminar flame simulation. Although these are suitable to be applied to any problem having solutions that are steep in a relatively small part of the computational domain, the focus is put on combustion

applications. Moreover, we investigate some numerical techniques to solve nonlinear algebraic systems.

A mathematical model for laminar flames is presented in Chapter 2. This is derived by using the *Reynolds' Transport Theorem* to write the *energy*, *mass* and *momentum conservation equations*. The *constant pressure approximation*, together with the asymptotic analysis of the terms entering the combustion and the flow equations, allows us to neglect the second order quantities and to make the equations simpler, see i.e.[48]. The last section of Chapter 2 is dedicated to the *thermo-diffusive* model. Although it relies on a quite crude, though reasonable, assumption, i.e. the *constant density approximation*, its simplicity reveals to be extremely useful to gain insight in the structure and in the behaviour of laminar flames. A nice survey of issues related to the *thermo-diffusive* model can be found in [9]. Results about existence and uniqueness of the solution under quite mild assumptions are shown in [8, 10].

Solutions of boundary value problems (BVPs) are often characterised by the presence of regions where gradients are quite large compared to those in the rest of the domain, where the solution can be considered relatively smooth. When solving such problems numerically, this solution behaviour requires a much finer grid in such *high activity* regions than in regions where the solution is fairly smooth. This is the case, for instance, for the equations that describe laminar flames: most of the activity is concentrated in the so-called *flame front*, a narrow area where the temperature rises steeply and chemical reactions take place. Other examples of such BVPs are mathematical models of shock waves or semiconductor devices. Finite element methods can quite easily cope with this by using meshes that are fine only where this is needed, but they give rise to complicated data structures and to systems of equations whose matrices are not banded. Even if one considers finite volume methods, the unstructured grids that are a consequence of this non-uniform behaviour, are very complicated to handle. For the latter type of methods, one may still obtain a simple data structure by using tensor product grids. These should be such that they have smaller grid size in the high activity region. However, this most likely leads to too many grid points in other parts of the domain and thus to low computational efficiency. In particular, this occurs when the front is not aligned with either one of the coordinate axes and/or when its shape is far from being a straight line, which is often the case.

In Chapter 3 we therefore introduce the so-called *local defect correction* (LDC) method based on the use of tensor product grids on rectangular domains. Roughly speaking, it goes as follows. First, an approximation on a coarse grid in the entire domain is computed. Then, this is used to define a boundary value problem on a subdomain where a different grid is employed. The thus found solution on the latter subdomain is used to define a defect on the global domain, which, in turn, induces an updated global domain problem. The local defect correction method is then used in an iterative way. It converges very fast: typically one iteration suffices.

LDC was first introduced in [26]. Its convergence behaviour has been studied in [21, 2, 3] using, as a model problem, the Poisson equation discretised with the finite difference method. The main results of these studies are summarised in Section 3.3. In [4] the authors combine the LDC method with the finite volume discretisation while retaining the conservation properties characteristic of such technique. An application of LDC

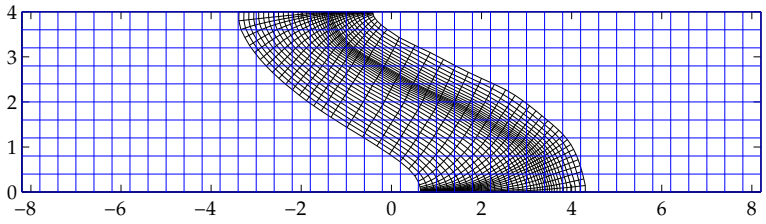


Figure 1.2: Thermo-diffusive model. Coarse and fine grids.

with multi-level adaptive grid refinement to perform laminar flame simulation can be found in [2]. In [43, 42] it is shown how different kinds of grids, namely a global rectangular and a local cylindrical, can be coupled by LDC and applied to the study of viscous flows. Finally, the application of the LDC method in combination with finite element discretisation has been studied in [64].

Since the high activity region can have an irregular shape, the boundaries of the refinement area may not consist of straight lines. In such case the local BVP problem can be solved efficiently by using curvilinear fine grids that tightly follow the shape of the solution. This method shows one drawback: when transforming the local BVP from a Cartesian to a curvilinear coordinate system, some more terms appear in the PDEs. However, the problem can be partially overcome by using an orthogonal fine grid. This requirement turns out to have also another advantage: the accuracy of numerical differencing techniques turns out to be the highest. Clearly, the generation of the fine grid is crucial for the efficiency of the method. In literature several ways to generate *body fitted* or *solution fitted* orthogonal coordinate systems can be found. A good summary is given in [61, 62]. One of the most well-known and widely used techniques is the classical *conformal mapping*, but it allows little control on the line spacing and will therefore not be considered here. The remaining methods to generate an orthogonal grid are basically of two types: *trajectory* and *field methods*. In the first approach, an existing nonorthogonal grid is transformed into an orthogonal one. One set of lines of the original nonorthogonal grid is retained; the other is displaced, through a marching process, such that the intersection between the two sets of lines is orthogonal. The second approach is based on the solution of an elliptic or a hyperbolic PDE system. In this thesis a method belonging to the first group has been chosen. The most attractive property is, for us, the possibility to retain a family of coordinate lines. In fact, if this set is aligned with the solution iso-contours, we can get a substantial reduction in the number of fine grid points that are necessary to reach a certain level of accuracy. We use a method introduced in [16], that actually turns out to be extremely simple and computationally not expensive. Just a few changes are needed to be adapted to our purposes. As an example, Figure 1.2 shows both a global coarse and a fine *solution fitted* grid used to solve the *thermo-diffusive* model.

The algorithm presented in Chapter 3 is tested with a variety of benchmark problems in Chapter 4. The first examples study the application of LDC in combination with rectangular fine grids that are slanted with an angle α over the Cartesian axes of the

global domain. A complexity analysis is then performed introducing a gain function G that compares the LDC and the tensor product grid algorithms in terms of memory requirements. Moreover, a nice feature of LDC is shown, i.e. once the desired accuracy is fixed, the number of fine grid points in the direction orthogonal to the level curves is independent of the steepness of the solution. The last problem deals with curvilinear refinement domains. An application where a rectangular coarse and a cylindrical fine grid have been combined is already presented in [43]. In this thesis such a concept is extended to general *solution fitted* local grids that are generated with the methodology presented in Chapter 3. Eventually, an asymptotic analysis is carried out to determine the overall complexity of the LDC algorithm. In particular, we show that the possibility to use the level curves of the coarse grid solution as a set of coordinate lines implies that the leading term of the formula that expresses the complexity is determined by the coarse grid problem. This certainly holds for a class of problems whose source term can be expressed as an exponential function. For both slanting and curvilinear grids a difficulty may arise if the solution iso-lines are not orthogonal to the boundary of the global domain. In this case there can be some points of the local grid that do not belong to the global domain itself. We show how to deal with such a situation.

Like many equations describing other physical phenomena, the combustion equations contain some terms that are extremely nonlinear. Once discretised, they yield algebraic systems that can be written as

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \tag{1.1}$$

and that are very hard to solve. The simplest and fastest method to deal with this kind of problem is the *Newton method*. Unfortunately, Newton iterations only converge when the initial guess is very close to the solution of this system. If this condition is not satisfied, i.e. if the initial guess does not belong to the *convergence region*, the algorithm will fail. The robustness of the *Newton method* can be improved by *embedding* the original nonlinear system into a time dependent problem making use of a (dummy) time variable. The steady state solution of the new problem will then be a zero of the original system. Early studies on this subject have been done, amongst others, in [27]. It is found that the stability of the integration process, more than its accuracy, is an important requirement to successfully obtain a root of the algebraic system. In [28] a class of iterative methods is constructed by *embedding* an algebraic system into a time dependent problem. Time integration is performed by a Runge-Kutta method. Moreover, the concept of *radius of convergence* for an iterative method is introduced. More recent studies have investigated the behaviour of the equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = -\mathbf{V}(\mathbf{x})\mathbf{F}(\mathbf{x}), \tag{1.2}$$

where $\mathbf{V}(\mathbf{x})$ is a nonsingular matrix used to improve the scaling of the system. In [34] a convergence analysis of system (1.2), integrated with a Rosenbroek method, is carried out. The extension of this analysis to Differential Algebraic Equations (DAEs) of order 1 is performed in [15]. In Chapter 5 we consider the embedded system

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = -\mathbf{J}(\mathbf{x})\mathbf{F}(\mathbf{x}), \tag{1.3}$$

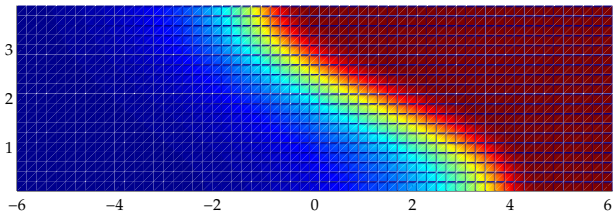


Figure 1.3: Thermo-diffusive model. Temperature

also known as the *Davidenko equation*, where $\mathbf{J}(\mathbf{x})$ is the Jacobi matrix of $\mathbf{F}(\mathbf{x})$. We start from the work carried out in [35]. There, system (1.3) is integrated with the so-called *mixed Euler method* to improve stability of the solution time history. The time step is determined trying to control the *discretisation error* over the integration interval: its estimate is given via a discrete approximation of the second derivative. Here we try to achieve a better control over the *discretisation error* in order to make the path towards the steady state solution more reliable and robust. We present two different alternatives: one is based on the use of two integration formulae having different order, the other is based on the possibility to control the time step by comparing two different solution approximations obtained with the same integration formula and with different time steps. We work out those two possibilities and derive two algorithms to implement them. A series of benchmark problems is presented to test their robustness and to compare the computational work that they require with respect to the *mixed Euler method* presented in [35].

Chapter 6 is devoted to the solution of two different 2D laminar flame models, and consists of two main sections. The first one is dedicated to the *thermo-diffusive* model introduced in Chapter 2. The computational domain is a part of an infinite channel whose walls are adiabatic and inert. Moreover, at its left side we impose a nonuniform flow that gives the flame a very peculiar shape and makes it suitable to be solved with LDC and curvilinear fine grids. The equations are discretised by finite differences; the resulting algebraic system is embedded in a time dependent problem and solved with the *Mixed Euler method*. Then we determine the temperature distribution in the channel, see Figure 1.3, as well as the velocity of the flame front. A similar problem is solved in [39] in a finite element context with an *adaptive grid* procedure.

In the second section of Chapter 6 we present the simulation of a laminar flame in a *Bunsen* burner whose chemistry consists of a five species reaction scheme. An application of LDC to laminar flames is already proposed in [2]. The author solves a problem previously presented in [6] and combines LDC with a procedure for adaptive gridding introduced in [6, 7, 63]. The equations are based on a vorticity-velocity formulation. The flame front region is covered with several overlapping rectangular patches that belong to 3 different refinement levels. Finite difference discretisation is used.

Our aim is to combine the LDC algorithm with a curvilinear orthogonal fine grid built on top of the flame front. The coarse grid problem is solved by an existing laminar

flame simulation program that is mainly developed in [30]. Its characteristic features are the finite volume discretisation technique and the multigrid method solver, see [66]. Since the program is designed for to flame computations in Cartesian geometries, we are required to rewrite the combustion and Navier-Stokes equations in terms of general coordinates. The choice of the velocity primary variables is an important issue. In fact, one has the option to use the Cartesian, the contravariant or the covariant velocity components. Several studies are available in literature. In [53] it is shown that the strong conservation form can be completely satisfied when using the Cartesian velocity components as primary variables. Moreover, we manage this way to retain the same form of the equations as it was for the Cartesian domain. This is a big advantage when building something on top of an already existing code. The effect of mesh skewness and metric coefficients evaluation is studied, for instance, in [13, 52].

Since the equations modeling combustion are very stiff, only implicit time integration methods can be used. However, they lead to algebraic systems that are very difficult to solve and that require a big computational effort. A way to overcome those difficulties is to use *pressure correction* methods. These are introduced in [33] and generalised in [30, 32] to combustion problems. When using the *combustion approximation* in combination with standard discretisation schemes, the pressure can show some spurious solutions resulting in irregular wiggles. This can happen both in a finite element and in a finite volume/finite difference context. Several methods have been developed to prevent such behaviour. Since the pressure gradients only appear in the momentum equations, a simple remedy is to approximate these on a finer grid. This approach is typical for finite volume discretisation. This remedy, although widely used, gives in some cases poor results. A detailed analysis of the problem can be found in [49, 50]. Another interesting approach is presented in [44] and is based on *filtering* the pressure field at the beginning of each time step. A standard approach to get rid of pressure wiggles when discretising with finite volumes/finite differences consists of using *staggered grids*, see [46]. This is adopted in the laminar flame program used in this thesis. Nevertheless, when translating the equations from Cartesian to curvilinear coordinates, there are several cases in which the grid staggering can lose its effectiveness. Such a situation is investigated in several papers. In [52, 51] a possible cause for this behaviour is proposed. Another approach to get rid of pressure wiggles is based on the *smoothing* of the pressure field, see i.e. [67].

Finally, in Chapter 7 conclusions are drawn and recommendations for future work are given.

CHAPTER 2

Modeling laminar flames

Since this thesis mainly focuses on computational aspects of numerical flame simulation, we omit discussing the several available mathematical models that have been developed so far to describe laminar flames. Instead, this chapter is devoted to present the combustion model introduced in [55, 56, 60]. In Section 2.1 we define the variables used to describe a reacting gas flow and we give the related thermodynamic relations. In Section 2.2 we recall the *Reynolds' Transport Theorem* that is used in Section 2.3 to derive the *conservation equations*. There, also the models for both the *stress tensor* and the *diffusion velocities* are presented. The laminar flame chemical production terms are introduced in Section 2.4, while in Section 2.5 the *constant pressure approximation* is discussed. This applies to flows characterised by a highly subsonic nature and helps to simplify the resulting system of equations that is finally worked out in the same section. Since the combustion equations represent a highly nonlinear system whose solution requires big computational effort, in Section 2.6 we introduce a simplified model, the so called *thermo-diffusive model*, derived on the basis of the *constant density* approximation.

2.1 Reacting gas flow

The way we derive the equations describing laminar flames is based on the continuum mechanics approach coupled with the laws of chemical kinetics and thermodynamics. Thus, we assume that the reacting flow is a continuum composed of N_s chemical species. This means that every control volume that is small enough to be considered punctiform still contains a big number of molecules of each distinct species. The state of such a mixture is supposed to be completely determined by the following scalar quantities: *pressure* p, *temperature* T and *density* ρ, also referred to as *state variables*. If we assume our mixture to be an ideal gas, the equation of state that couples density, pressure and temperature, reads

$$\frac{p}{\rho} = \frac{RT}{M}. \tag{2.1}$$

Here R is the universal gas constant and M the weighted average molar mass of the mixture. The latter can be expressed in terms of the molar masses M_i of the chemical

species as

$$\frac{1}{M} = \frac{1}{\rho}\sum_{i=1}^{N_s}\frac{\rho_i}{M_i}, \tag{2.2}$$

with ρ_i the mass density of the i-th species. It is customary to describe the composition of the mixture by the species *mass fractions* $Y_1, Y_2, \ldots, Y_{N_s}$, defined as

$$Y_i = \frac{\rho_i}{\rho}, \tag{2.3}$$

and it is easy to see that they satisfy the relation

$$\sum_{i=1}^{N_s} Y_i = 1. \tag{2.4}$$

Another scalar quantity reveals to be very useful for the description of a flame: the total specific energy E. It is defined as the sum of the kinetic contribution and the total internal energy e, i.e.

$$E = \frac{1}{2}|\mathbf{v}|^2 + e, \tag{2.5}$$

with

$$e = h - \frac{p}{\rho}, \tag{2.6}$$

where $\mathbf{v}$ is the velocity vector field, while h in (2.6) represents the specific enthalpy, i.e. the weighted sum of the species specific enthalpies h_i

$$h = \sum_{i=1}^{N_s} Y_i h_i. \tag{2.7}$$

Moreover, h_i is given by

$$h_i = h_i^0 + \int_{T_0}^{T} c_{p,i}(\tau)\, d\tau, \tag{2.8}$$

where h_i^0 is the enthalpy of formation of species i at a certain temperature T_0 and pressure p_0 (usually the standard conditions) and $c_{p,i}$ is the specific heat of the i-th species at constant pressure.

2.2 General conservation law

The derivation of the combustion model to be introduced in this chapter is based on a Lagrangian formulation. Therefore, each control volume $V(t)$ in the fluid is considered as an individual entity that consists of the same particles and for which the conservation laws hold. These can be stated making use of *Reynolds' Transport Theorem*. Let us consider the material volume $V(t)$ that moves with velocity $\mathbf{v}(\mathbf{x}, t)$ and a scalar or vector function $b(\mathbf{x}, t)$, defined in $V(t)$, which represents a density of a certain property

per unit mass. The variable $b(\mathbf{x}, t)$ is also called the *intensive value* of the considered property, while its *extensive value* is given by

$$B(t) = \int_{V(t)} \rho b \, dV. \tag{2.9}$$

According to the transport theorem, if b is a scalar field the material derivative of $B(t)$ can be computed as follows [60]

$$\frac{dB(t)}{dt} = \int_{V(t)} \frac{\partial \rho b}{\partial t} \, dV + \oint_{\partial V(t)} \rho b \mathbf{v} \cdot \mathbf{n} \, dS, \tag{2.10}$$

where $\partial V(t)$ is the surface that encloses $V(t)$ and $\mathbf{n}$ is the outward unit vector normal to $\partial V(t)$. There are two ways for $B(t)$ to change within $V(t)$; viz. transport across the boundary $\partial V(t)$ and production mechanisms inside the control volume itself. If we name $\mathbf{f}(\mathbf{x}, t)$ the flux and $s(\mathbf{x}, t)$ the production term per unit volume and per unit time of the property $b(\mathbf{x}, t)$, respectively, we can write

$$\frac{dB(t)}{dt} = \int_{V(t)} s \, dV - \oint_{\partial V(t)} \mathbf{f} \cdot \mathbf{n} \, dS. \tag{2.11}$$

The right-hand sides of equations (2.10) and (2.11) must be equal. By applying the Gauss theorem to the boundary integrals in both (2.10) and (2.11), we obtain

$$\int_{V(t)} \left(\frac{\partial \rho b}{\partial t} + \boldsymbol{\nabla} \cdot (\rho b \mathbf{v}) \right) dV = \int_{V(t)} \left(s - \boldsymbol{\nabla} \cdot \mathbf{f} \right) dV. \tag{2.12}$$

Relation (2.12) must hold for an arbitrary control volume $V(t)$. This leads to the general conservation law

$$\frac{\partial \rho b}{\partial t} + \boldsymbol{\nabla} \cdot (\rho b \mathbf{v}) = s - \boldsymbol{\nabla} \cdot \mathbf{f}. \tag{2.13}$$

If $\mathbf{b}$ is a vector field, applying (2.13) componentwise we get

$$\frac{\partial \rho \mathbf{b}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{b} \otimes \mathbf{v}) = \mathbf{s} - \boldsymbol{\nabla} \cdot \mathbf{f}, \tag{2.14}$$

with $(\mathbf{b} \otimes \mathbf{v})_{ij} = b_i v_j$, $[\boldsymbol{\nabla} \cdot (\mathbf{b} \otimes \mathbf{v})]_i = \boldsymbol{\nabla} \cdot (b_i \mathbf{v})$ and $\mathbf{f}$ a tensor.

2.3 The conservation equations

In this section we use (2.13) and (2.14) to derive the conservation equation of mass for the individual species and for mass, momentum and specific energy of the mixture.

- **The continuity equation**

It is also called *mass conservation* and states that, within the fluid, mass can neither be created nor destroyed. If $B(t)$ represents the mass contained in a control volume $V(t)$, the application of (2.13) with $b = 1$, $\mathbf{f} = \mathbf{0}$ and $s = 0$ reads

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v}) = 0. \tag{2.15}$$

- **The species conservation equations**

The velocity vector field $\mathbf{v}$ that appears in (2.5) is a bulk mass-weighted quantity, related to the velocities $\mathbf{v}_i$ of the individual species by the relation

$$\mathbf{v} := \sum_{i=1}^{N_s} Y_i \mathbf{v}_i. \tag{2.16}$$

It is customary to describe the motion of the N_s species using the *molecular diffusion velocities* $\mathbf{V}_i$. They are defined as

$$\mathbf{V}_i = \mathbf{v}_i - \mathbf{v}. \tag{2.17}$$

Making use of (2.16), we see that, after weighting with the mass fractions, the molecular diffusion velocities add to zero

$$\sum_{i=1}^{N_s} Y_i \mathbf{V}_i = \sum_{i=1}^{N_s} Y_i \mathbf{v}_i - \mathbf{v} = 0. \tag{2.18}$$

The conservation law (2.13) applied to the mass fraction Y_i contained in the control volume $V(t)$ reads

$$\frac{\partial \rho Y_i}{\partial t} + \nabla \cdot \left(\rho \mathbf{v} Y_i\right) = s_i - \boldsymbol{\nabla} \cdot \mathbf{f}_i, \qquad i = 1, \cdots, N_s, \tag{2.19}$$

with

$$\mathbf{f}_i = \rho_i \mathbf{V}_i = Y_i \rho \mathbf{V}_i. \tag{2.20}$$

An expression for the chemical production term s_i is introduced in the next section. Here, let us consider the flux of the i-th chemical species through $\partial V(t)$. A quite accurate transfer model makes use of the generalized Fick's law and reads

$$\mathbf{V}_i = -D_{im} \frac{\boldsymbol{\nabla} Y_i}{Y_i}. \tag{2.21}$$

In (2.21) the mixture-averaged gas diffusion coefficient D_{im} is a measure of the diffusivity of the i-th species in the gas mixture and it is given by

$$D_{im} = \frac{(1 - Y_i)}{\sum\limits_{j=1,\, j \neq i}^{N_s} \frac{X_j}{D_{ij}}}, \tag{2.22}$$

where X_j is the mole fraction of the j-th species and D_{ij} are the binary diffusion coefficients. The model expressed by (2.22) is also called the *trace species approximation* because, in a multi-component flow, it applies strictly only to the trace species. Since in an atmospheric fuel mixture nitrogen, with its 80% in weight, constitutes by far the most abundant species, we can consider this model quite reliable when used to describe the behaviour of the other $(N_s - 1)$ species in fuel/air flames. This implies that only the first $(N_s - 1)$ equations (2.19) must be solved, since the nitrogen mass fraction can be determined by using relation (2.4).

- **The momentum equations**

Now let $\mathbf{B}(t)$ be the momentum of the fluid contained in $V(t)$. According to the *Newton's second law*, in an inertial system the following holds

$$\frac{d\mathbf{B}(t)}{dt} = \mathbf{F}, \tag{2.23}$$

where $\mathbf{F}$ is the resultant of all forces applied to the system. This vector can be expressed as the sum of two contributions: the body force $\rho\mathbf{g}$, that corresponds to the volumetric contribution in (2.12), with $\mathbf{g}$ the gravitational acceleration, and the divergence of the surface forces, represented by the stress tensor $\mathbf{P}$. The resulting equations are

$$\frac{\partial \rho\mathbf{v}}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\mathbf{v}\otimes\mathbf{v}) = \rho\mathbf{g} - \boldsymbol{\nabla}\cdot\mathbf{P}. \tag{2.24}$$

Surface forces depend on the nature of the fluid and on the assumptions made about the relation between deformation and stresses within the fluid. If this is assumed to have an isotropic Newtonian behaviour, the tensor $\mathbf{P}$ has the following form

$$\mathbf{P} = \Big(p + \frac{2}{3}\mu(\boldsymbol{\nabla}\cdot\mathbf{v})\Big)\mathbf{I} - \mu\Big((\boldsymbol{\nabla}\otimes\mathbf{v}) + (\boldsymbol{\nabla}\otimes\mathbf{v})^{T}\Big) = p\mathbf{I} + \boldsymbol{\tau}, \tag{2.25}$$

with p the hydrostatic pressure, $\mathbf{I}$ the unit tensor, $\boldsymbol{\tau}$ the viscous stress tensor and μ the dynamic viscosity.

- **The energy equation**

This is obtained by applying (2.13) with $b = E$. The rate of change of the internal total energy density is due to the heat flux and to the work done on the system by body and surface forces. The resulting conservation law is

$$\frac{\partial \rho E}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\mathbf{v}E) = -\boldsymbol{\nabla}\cdot\mathbf{q} - \boldsymbol{\nabla}\cdot(\mathbf{P}\,\mathbf{v}) + \rho\mathbf{v}\cdot\mathbf{g}. \tag{2.26}$$

An alternative formulation for the energy equation can be obtained by subtracting from (2.26) the inner product between the momentum equation (2.24) and $\mathbf{v}$. This way we get

$$\frac{\partial \rho e}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\mathbf{v}e) = -\boldsymbol{\nabla}\cdot\mathbf{q} - \mathbf{P} : \nabla\mathbf{v}, \tag{2.27}$$

with

$$\mathbf{P} : \nabla\mathbf{v} = \boldsymbol{\nabla}\cdot(\mathbf{P}\mathbf{v}) - (\boldsymbol{\nabla}\cdot\mathbf{P})\cdot\mathbf{v}. \tag{2.28}$$

The energy conservation equation can also be formulated in terms of the enthalpy h. Substituting (2.6) into (2.27), we obtain

$$\frac{\partial \rho h}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\mathbf{v}h) = -\boldsymbol{\nabla}\cdot\mathbf{q} + \frac{Dp}{Dt} + \boldsymbol{\tau} : \nabla\mathbf{v}. \tag{2.29}$$

The stress tensor has already been introduced, so, in order to complete the expression in the right-hand side of (2.27), we need a model for the heat flux vector. We assume that the only two mechanisms of importance to our purpose are the conductive heat transfer

and the enthalpy transport due to species , leaving out both the radiative contribution and the heat transfer due to mass diffusion (*Dufour effect*). By introducing the *thermal conductivity* of the gas mixture λ, we get

$$\mathbf{q} = -\lambda \nabla T + \rho \sum_{i=1}^{N_s} h_i Y_i \mathbf{V}_i. \tag{2.30}$$

Sometimes it is useful to write the energy equation in terms of temperature. Let us introduce the specific heat at constant pressure of the mixture

$$c_p = \sum_{i=1}^{N_s} c_{p,i} Y_i. \tag{2.31}$$

If we assume c_p to be constant and all $c_{p,i} (i = 1, \dots N_s)$ to be equal, then (2.29) can be transformed, see [2], into

$$c_p \frac{\partial \rho T}{\partial t} + c_p \boldsymbol{\nabla} \cdot (\rho \mathbf{v} T) = \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) + \frac{Dp}{Dt} + \boldsymbol{\tau} : \nabla \mathbf{v} - \sum_{i=1}^{N_s - 1} h_i^* s_i, \tag{2.32}$$

where $h_i^* = h_i - h_{N_s}$.

2.4 Chemistry in laminar flames

In Section 2.3 the species conservation equations are introduced. In order to complete the model, we need to specify the form of the chemical production term s_i. We focus on pure methane-air flames, whose combustion mechanisms are described in detail in [19] as consisting of 210 reactions involving 36 chemical species. If $\nu'_{i,j}$ and $\nu''_{i,j}$ are the stoichiometric coefficients for the reactant and the product species $\mathcal{M}_i$ appearing in the j-th reaction, respectively, a chemical mechanism can be described by the following

$$\sum_{i=1}^{N_s} \nu'_{i,j} \mathcal{M}_i \rightleftharpoons \sum_{i=1}^{N_s} \nu''_{i,j} \mathcal{M}_i. \tag{2.33}$$

The rate at which the concentration c_i of the species i changes because of the reaction j is proportional to the product of the concentration of reactants $c_k = \rho Y_k / M_k$ with $k = 1, \dots, N_s$, see [12]. Therefore, the chemical production term s_i is given by

$$s_i = M_i \sum_{j=1}^{M} (\nu''_{i,j} - \nu'_{i,j}) \left(k_{f,j} \prod_{k=1}^{N_s} c_k^{\nu'_{k,j}} - k_{b,j} \prod_{k=1}^{N_s} c_k^{\nu''_{k,j}} \right). \tag{2.34}$$

In (2.34), $k_{f,j}$ and $k_{b,j}$ are the specific reaction rate coefficients for the forward and backward path in the j-th reaction, respectively. The specific reaction rate coefficients $k_{f,j}$ depend on the temperature T and can be represented with an Arrhenius-like relation

$$k_{f,j} = A_{f,j} \exp\left(- \frac{E_{f,j}}{RT} \right), \tag{2.35}$$

where $E_{f,j}$ is the activation energy for the j-th forward reaction. The frequency factors $A_{f,j}$ are given by

$$A_{f,j} = B_{f,j} T^{\alpha_{f,j}}. \tag{2.36}$$

The coefficients $k_{f,j}$ are strongly dependent on the absolute temperature T and can be considered to be zero when T is below a certain threshold value. The backward reaction equilibrium constant, see [68], is given by

$$k_{b,j} = \frac{k_{f,j}}{K_{c,j}}, \tag{2.37}$$

with $K_{c,j}$ the equilibrium constant of the j-th reaction.

The solution of a detailed chemical model, as the one described in [19], would require a big computational effort. Therefore, the study of reduction strategies, aimed at systematically reducing the number of chemical species and reactions, plays a crucial role. The *skeletal mechanism*, consisting of 25 reactions and 15 chemical species, is suitable to predict the most important features of lean methane/air laminar flames. Those are combustion processes that occur when the oxidizer, in this case oxygen, is in excess with respect to the fuel, see [65]. The temperature, the species mass fractions and the burning velocity of a lean laminar flame can be predicted in an even cheaper way with the *one step* model, where $N_s = 5$ and $M = 1$. The latter is used for the computations presented in this thesis. Furthermore, we also introduce the *thermo-diffusive* model for laminar flames.

2.5 The isobaric approximation

Laminar flame propagation is a highly subsonic phenomenon. In fact, the characteristic flow speeds do usually not exceed $3\,m/s$ and, since the speed of sound at atmospheric pressure is about $300\,m/s$, the representative *Mach numbers* Ma are the order of 10^{-2}. This leads to the so-called *combustion approximation* [14]. By formally expanding the pressure as a series of Ma, it can be shown that

$$p(\mathbf{x}, t) = P(t) + \tilde{p}(\mathbf{x}, t), \qquad \text{with} \qquad \frac{\tilde{p}(\mathbf{x}, t)}{P(t)} = \mathcal{O}(Ma^2), \tag{2.38}$$

i.e. the pressure is constant in space up to the second order, its leading term depending only on time. This implies that the pressure gradient can be neglected in the energy equation, see [48]. If combustion takes place at atmospheric pressure, $P(t) = P_0$ is constant and the ideal gas law becomes

$$\rho = \frac{P_0 M}{RT}. \tag{2.39}$$

It has to be noticed that, although the pressure terms have disappeared from the energy equation, their contribution cannot be ignored in the momentum equations, where they constitute the flow driving forces. The subsonic behaviour of laminar flames helps to further simplify (2.29). In fact, the viscous heating, represented by the last term in the

right-hand side, can be shown to be proportional to Ma^2 [48]. This implies that it is small and can be neglected as well.

In this thesis we consider 2D geometries. The model so far developed consists finally of the following $N_s + 3$ independent equations

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v}) = 0, \tag{2.40a}$$

$$\frac{\partial \rho Y_i}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} Y_i) = \boldsymbol{\nabla} \cdot (\rho D_{im} \nabla Y_i) + s_i, \qquad i = 1, \cdots, N_s - 1, \tag{2.40b}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} \otimes \mathbf{v}) = -\boldsymbol{\nabla} \cdot \mathbf{P} - \rho \mathbf{g}, \tag{2.40c}$$

$$\frac{\partial \rho h}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} h) = \boldsymbol{\nabla} \cdot (\lambda \nabla T) + \boldsymbol{\nabla} \cdot \Big(\rho \sum_{i=1}^{N_s} h_i D_{im} \nabla Y_i \Big). \tag{2.40d}$$

The independent variables are here ρ, p, T, h, $\mathbf{v}$ and $Y_i (i = 1, \cdots, N_s - 1)$, thus $N_s + 5$ in total. This means that the problem must be closed by adding two equations. These are the equation of state (2.39) and the thermodynamic identities (2.7) and (2.8).

2.6 The thermo-diffusive model

Finding a numerical solution for the system that describes laminar flame propagation is not an easy task. Therefore, some simplified models have been developed. These, although with a loss of information details, help to gain a good understanding of the flame behaviour with little numerical effort. Let us consider a single one-step chemical model of a two component gaseous mixture

$$\mathcal{M}_1 \rightarrow \mathcal{M}_2. \tag{2.41}$$

It has been shown in [38] that, for a one-dimensional isobaric model, the introduction of a mass-weighted Lagrangian coordinate allows to decouple the combustion variables Y, the mass fraction of the reactant, and T from the flow variables $\mathbf{v}$ and p. Unfortunately, this does not hold for two-dimensional geometries, where the same result can only be achieved by considering the *constant density approximation*

$$\rho = \text{Constant}. \tag{2.42}$$

Assuming that the flow is inviscid, the application of (2.42) to the equations describing a two-components mixture leads to the two decoupled systems

$$\boldsymbol{\nabla} \cdot \mathbf{v} = 0, \tag{2.43a}$$

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} \otimes \mathbf{v}) = -\boldsymbol{\nabla} p, \tag{2.43b}$$

and

$$\rho \frac{\partial Y}{\partial t} + \rho \boldsymbol{\nabla} \cdot (\mathbf{v} Y) = \rho \boldsymbol{\nabla} \cdot (D \boldsymbol{\nabla} Y) - M\omega(\rho Y, T), \tag{2.44a}$$

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p \boldsymbol{\nabla} \cdot (\mathbf{v} T) = \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) + QM\omega(\rho Y, T), \tag{2.44b}$$

with

$$\omega(\rho Y, T) = \frac{\rho Y}{M} B \exp\left(-\frac{E}{RT}\right). \tag{2.45}$$

In (2.43) the contribution of gravity has been omitted and the energy equation (2.44b) has been written in terms of the temperature as independent variable. Moreover, Q is the heat release per unit mass of fuel consumption. When making use of the thermo-diffusive model, we consider also a slightly simpler model for the chemical production term. In particular, the dependency of the frequency factor on the temperature is suppressed.

We see that the *combustion variables* T and Y do not appear in (2.43) anymore: this system describes only the hydrodynamic variables behaviour, i.e. $\mathbf{v}$ and p. On the other hand, (2.42) is the new equation of state. System (2.44) can then be solved for the *combustion variables* if the velocity field is supposed to be known and to satisfy (2.43). It has to be remarked that, while the isobaric assumption finds its roots in the physics of the combustion phenomena, the *constant density approximation* is in some way more difficult to justify. In fact, it is based on qualitative considerations, see [54], that show how hydrodynamic effects are less important than conduction and diffusion mechanisms in determining the thermo-diffusive instabilities of laminar flames. This implies that the first can be neglected such that thermal and hydrodynamic perturbations do not influence each other. The way to achieve such a separation is to discard the density fluctuations from the model.

CHAPTER 3

LDC method: algorithm, properties and local grid refinement

The discretisation of systems describing laminar flames requires a very high resolution in only a small part of the domain. In fact, the activity is almost completely concentrated in the *reaction layer*, a narrow band where gradients are relatively high. Apart from combustion processes, there are numerous physical phenomena described by PDE equations that exhibit this behaviour, like shock waves or semiconductor devices. For this kind of problems, *composite grids* turn out to be very attractive. The principle relies on the possibility to cover the global domain with a relatively coarse grid and the areas of interest with grids or patches of grids that are finer than the coarse one. This chapter is devoted to explain the basics of the LDC method. Moreover, we show how this can be used in combination with *solution fitted* grids. In the first section we consider a general boundary value problem on a domain Ω with boundary conditions that can be either of Dirichlet or Neumann type. Although both Ω and the refinement region Ω^l are represented as being rectangular, the notation and the LDC algorithm presented there apply to more general configurations, whichever the shape of those domains is. In the second section, besides the very important *safety region* concept, we present some interesting convergence and accuracy property results that apply strictly only to Dirichlet boundary value problems defined on a rectangular domain. Section 3 is dedicated to broaden the LDC concepts: we use this technique coupling different grid types and different discrete operators. The reason for this is to be found in the fact that the *high activity* regions of the processes we want to describe have very often a shape that can be better followed by a curvilinear grid, which allows to compute the solution with a greater accuracy and much less grid points. Obviously, the system of equations has to be mapped into the new computational domain. Since we aim to keep this mapping as easy as possible, we consider only *curvilinear orthogonal* grids. Although several more recent techniques are available to build such a *solution fitted* grid, we present one of the easiest and computationally less expensive that we have found in literature. Since it has

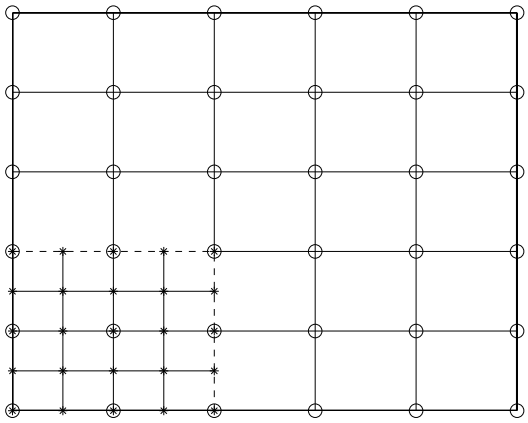


Figure 3.1: Global coarse and local fine grids. The circles represent the nodes of the coarse grid Ω_H; the stars the nodes of the fine grid Ω_h^l. The dashed line is the interface Γ between the global and the local grid.

originally been developed as a *body fitted* grid generation method, we adapt it to our purposes to make it a useful tool to be used in combination with LDC.

3.1 Formulation of the LDC method

In this section the general formulation of the LDC method is introduced. Let us consider the BVP

$$\mathcal{L}[u] = f, \qquad \mathbf{x} \in \Omega, \tag{3.1a}$$

$$\mathcal{B}[u] = g, \qquad \mathbf{x} \in \partial\Omega, \tag{3.1b}$$

where $\Omega \subset R^2$ is a simply connected domain, $\mathcal{L}$ a linear elliptic operator, f a source term and g the value of u at the boundary $\partial\Omega$. The boundary conditions, expressed by the operator $\mathcal{B}$, are either of Dirichlet or Neumann type. Let us define a discretisation of (3.1) on a uniform (coarse) grid of size H covering Ω, viz.

$$\mathcal{L}_H[u_H] = f_H, \tag{3.2}$$

where the right-hand side f_H contains both the source term and the contribution of the boundary conditions. In the iteration procedure below, we denote the solution of (3.2) by u_H^0.

Suppose now that u changes very rapidly in a small subdomain $\Omega^l \subset \Omega$. In this high activity region the grid size H is most likely too large to capture the behaviour of u, so we formulate a new discrete BVP in Ω^l by covering it with a grid Ω_h^l, see Figure 3.1. For the latter, h is either the actual grid size, if the fine grid is uniform, or a characteristic grid size, otherwise. In order to define a discrete BVP on Ω^l, let us introduce some notation.

We split $\partial\Omega^l$ in two parts: the interface Γ between Ω and Ω^l, i.e. $\Gamma = \partial\Omega^l \backslash (\partial\Omega^l \cap \partial\Omega)$, and $\partial\Omega^l \cap \partial\Omega$. This last subset can also be empty. Furthermore, we define $\Gamma_h := \Gamma \cap \Omega_h^l$, $\Omega_H^l := \Omega_H \cap \Omega^l$ and introduce the vector space of grid functions on Ω_H, $\mathcal{F}(\Omega_H)$, and the vector space of grid functions on Γ_h, $\mathcal{F}(\Gamma_h)$. We set $\mathcal{B}[u] = g$ on $\partial\Omega^l \cap \partial\Omega$ and we interpolate from u_H^0 the boundary conditions on Γ using an interpolation operator that maps grid functions on Ω_H onto grid functions on Γ_h

$$\mathcal{P}^{h,H} : \mathcal{F}(\Omega_H) \to \mathcal{F}(\Gamma_h). \tag{3.3}$$

Note that when the global and local grids are chosen like in Figure 3.1, we can have, as a special case of (3.3), $\mathcal{P}^{h,H} : \mathcal{F}(\Gamma_H) \to \mathcal{F}(\Gamma_h)$, with $\Gamma_H := \Gamma \cap \Omega_H$.

The discrete problem on Ω^l now reads

$$\mathcal{L}_h[u_h] = f_h - \mathcal{B}_\Gamma^h \mathcal{P}^{h,H}[u^H]. \tag{3.4}$$

Here the term f_h contains the contribution of both the right-hand side f and the boundary conditions on $\partial\Omega^l \cap \partial\Omega$, while $\mathcal{B}_\Gamma^h$ is the part of $\mathcal{L}_h$ operating on Γ_h. By solving (3.4), we obtain a fine grid solution u_h^0. This is used to estimate the local discretisation error of the coarse grid approximation, defined as

$$d_H := \mathcal{L}_H[u] - f_H, \tag{3.5}$$

i.e. the residual found by substituting the exact solution of (3.1) into the coarse grid scheme (3.2). To this end, we introduce the grid function w_H^0

$$w_H^0(\mathbf{x}) := \begin{cases} u_H^0(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_H \backslash \Omega_H^l, \\ \mathcal{R}^{H,h}[u_h^0](\mathbf{x}), & \text{if } \mathbf{x} \in \Omega_H^l. \end{cases} \tag{3.6}$$

Here

$$\mathcal{R}^{H,h} : \mathcal{F}(\Omega_h^l) \to \mathcal{F}(\Omega_H^l), \tag{3.7}$$

is the restriction operator that maps grid functions on Ω_h^l onto grid functions on Ω_H^l. Let $S_H(\mathbf{x})$ be the set of coarse grid points belonging to the stencil of $\mathcal{L}_H$ at the grid point $\mathbf{x} \in \Omega_H^l$; then we define the subset $\Omega_H^S \subset \Omega_H^l$ by the relation

$$\mathbf{x} \in \Omega_H^S \iff S_H(\mathbf{x}) \subset \Omega_H^l. \tag{3.8}$$

The approximation of d_H is then given by

$$d_H(\mathbf{x}) \doteq d_H^0(\mathbf{x}) := \chi_{\Omega_H^S}\big(\big(\mathcal{L}_H[w_H^0] - f_H\big)(\mathbf{x})\big), \tag{3.9}$$

with $\chi_{\Omega_H^S}$ the characteristic function of Ω_H^S, i.e.

$$\chi_{\Omega_H^S}(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_H^S, \\ 0 & \text{if } \mathbf{x} \notin \Omega_H^S. \end{cases} \tag{3.10}$$

Once d_H^0 has been calculated, it is possible to add it to the right-hand side of (3.2), resulting in the equation

$$\mathcal{L}_H[u_H^1] = f_H + d_H^0. \tag{3.11}$$

By solving (3.11), we get a new approximation u_H^1. Again, u_H^1 can be used to approximate the interface condition for the fine grid problem, giving rise to the following algorithm

LDC Algorithm

- Initialisation
  - Compute u_H^0 from the basic coarse grid problem (3.2);
  - Build a fine grid enclosing the high activity region;
  - Define the fine grid boundary value problem, interpolating the boundary conditions on Γ_h;
  - Compute u_h^0 from the fine grid BVP (3.4).
- Iteration $i = 1, 2, ...$
  - Compute the grid functions w_H^{i-1} using (3.6);
  - Estimate the local discretisation error d_H^{i-1} using (3.9);
  - Solve the new coarse grid problem

$$\mathcal{L}_H[u_H^i] = f_H + d_H^{i-1};$$

  - Define the new BVP on the fine grid, interpolating boundary conditions on Γ_h;
  - Compute u_h^i from the fine grid BVP (3.4).

The *i*-th approximation of the solution of BVP (3.1) is the composite grid solution

$$u_{H,h}^i(\mathbf{x}) := \begin{cases} u_H^i(\mathbf{x}) & \mathbf{x} \in \Omega_H \backslash \Omega_H^l, \\ u_h^i(\mathbf{x}) & \mathbf{x} \in \Omega_h^l, \end{cases} \tag{3.12}$$

i.e. $u_{H,h}^i$ is the fine grid solution inside Ω^l and the coarse grid solution outside Ω^l.

The following features of the LDC method are interesting to remark. The effectiveness of the algorithm relies on a proper definition of the high activity area. In fact, the accuracy of the composite grid solution cannot be bigger than the accuracy of the coarse grid solution in the not refined area. Then, on the one hand Ω^l has to be chosen broad enough to cover the region where gradients are high, on the other hand too many fine grid points would spoil the efficiency of the computations. A compromise between these two conflicting requirements has to be reached. There are several algorithms devoted to adaptive grid refinement. In general, they are based on the evaluation of the discrete derivatives on the coarse grid. In [2], for instance, some weight functions that depend on the values of the coarse grid solution and of its first derivatives are introduced. The refinement is then performed where the values of such function exceed a certain threshold. An even easier criterion is based on the knowledge of the asymptotic values of either the solution $u_{-\infty}$ and u_∞ or of the source term $s_{-\infty}$ and s_∞. In this case Ω^l can be defined such that in $\Omega \backslash \Omega^l$ we either have $|u_{-\infty} - \epsilon| < u < |u_\infty + \epsilon|$ or $|s_{-\infty} - \epsilon| < s < |s_\infty + \epsilon|$, respectively.

The situation of Figure (3.1), where Ω and Ω^l are both square and only one local domain is represented, is not the most general. In the following sections we will discuss the application of the LDC method in combination with several grid types that can be differently shaped and also protrude the global domain.

The elliptic discrete operators $\mathcal{L}_H$ and $\mathcal{L}_h$ have not been specified. In fact, they can be different from each other and arbitrarily chosen, provided they are invertible. We can also use different PDE on different local grids. This is the case, for instance, when the set of equation have to be transformed in order to be adapted to a different coordinate system.

3.2 Some convergence and accuracy properties

The following lemma and theorem have been introduced in [2]. They show some nice properties relevant to the *fixed point* of LDC iteration. It has to be pointed out that the lemma and theorems of this chapter refer to Dirichlet problems defined on a rectangular domain, as in Figure 3.1. To present them, we need some more notation. The coarse grid points can be partitioned as follows: $\Omega_H := \Omega_H^l \cup \Gamma_H \cup \Omega_H^c$ with $\Gamma_H := \Omega_H \cap \Gamma$ and $\Omega_H^c := \Omega_H \backslash (\Omega_H^l \cup \Gamma_H)$. This induces the following corresponding partition of u_H, viz.

$$u_H := \begin{pmatrix} u_H^l \\ u_H^\Gamma \\ u_H^c \end{pmatrix}, \tag{3.13}$$

as well as for all other grid functions on $\mathcal{F}(\Omega_H)$. Assuming a 5-points stencil at each grid point, i.e. there is no coupling between Ω_H^l and Ω_H^c, the discrete operator $\mathcal{L}_H$ can be symbolically written as

$$\mathcal{L}_H = \begin{pmatrix} \mathcal{L}_H^l & \mathcal{B}_H^{l,\Gamma} & 0 \\ \mathcal{B}_H^{\Gamma,l} & \mathcal{L}_H^\Gamma & \mathcal{B}_H^{\Gamma,c} \\ 0 & \mathcal{B}_H^{c,\Gamma} & \mathcal{L}_H^c \end{pmatrix}, \tag{3.14}$$

where $\mathcal{B}_H^{l,\Gamma} : \mathcal{F}(\Gamma_H) \to \mathcal{F}(\Omega_H^l)$. The other operators with double index in (3.14) have an analogous meaning.

The next lemma states that the LDC algorithm has converged to a *fixed point* when the coarse grid approximation does not change on the interface.

Lemma 3.1 *If* $u_H^{\Gamma,k} = u_H^{\Gamma,k-1}$ *for a certain iteration index* k*, then the* **LDC Algorithm** *converges and it has reached a fixed point, i.e.,*

$$u_H^i = u_H^k, \qquad u_{h,l}^i = u_{h,l}^k \tag{3.15}$$

for all $i = k, k+1, \ldots$

The concept of *safety region* has now to be introduced. Numerical experiments presented in [3, 20, 26] have shown that the convergence behaviour of the LDC algorithm is improved if the correction term (3.9) is applied to only a subset of Ω_H^S, excluding the coarse

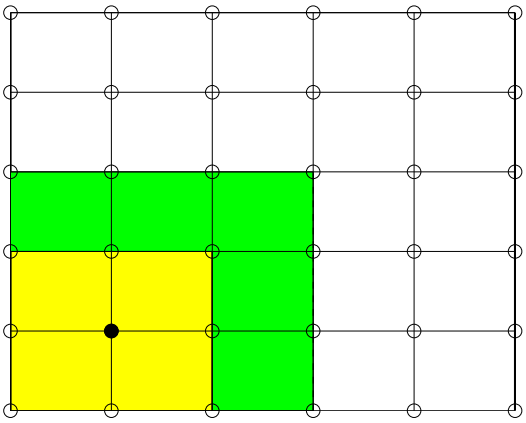


Figure 3.2: Global and local domains. The circles represent the coarse grid Ω_H; Ω^l is the union of the dark and the light grey regions. The dark grey area is the safety region; the light grey area the part of Ω_H^l where the defect correction is applied. In the figure it reduces to the grid point marked with the filled circle.

grid points adjacent to Γ, as shown in Figure 3.2. According to this, the approximation of the defect reads

$$d_H(\mathbf{x}) \doteq d_H^0(\mathbf{x}) := \chi_{\Omega_H^{def}}\big(\big(\mathcal{L}_H[w_H^0] - f_H\big)(\mathbf{x})\big), \tag{3.16}$$

with

$$\chi_{\Omega_H^{def}}(\mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_H^{def}, \\ 0 & \text{if } \mathbf{x} \notin \Omega_H^{def}, \end{cases} \tag{3.17}$$

and $\Omega_H^{def} \subset \Omega_H^S$. The safety region is then defined as $\Omega_H^S \backslash \Omega_H^{def}$.

We are now ready to introduce a theorem that states the properties of the fixed point $u_{H,h}$ of the LDC algorithm.

Theorem 3.1 *Consider the LDC iteration for the special case that there is no safety region, i.e.* $\Omega_H^{def} = \Omega_H^l$. *Assume that the LDC iteration converges to a fixed point* $u_{H,h}$. *Then* $u_{H,h}$ *has the following two properties*

- *the restriction of* u_h *on the local coarse grid equals* u_H^l, *viz.*

$$\mathcal{R}^{H,h} u_h = u_H^l, \tag{3.18}$$

- u_h, u_H^Γ *and* u_H^c *satisfy the system of equations*

$$\begin{pmatrix} \mathcal{L}_h & \mathcal{B}_h^{l,\Gamma}\mathcal{P}^{h,H} & 0 \\ \mathcal{B}_H^{\Gamma,l}\mathcal{R}^{H,h} & \mathcal{L}_H^\Gamma & \mathcal{B}_H^{\Gamma,c} \\ 0 & \mathcal{B}_H^{c,\Gamma} & \mathcal{L}_H^c \end{pmatrix} \begin{pmatrix} u_h \\ u_H^\Gamma \\ u_H^c \end{pmatrix} = \begin{pmatrix} f_h \\ f_H^\Gamma \\ f_H^c \end{pmatrix}. \tag{3.19}$$

The behaviour of the discretisation error on the composite grid is studied in [21]. The model problem consisting of the *Poisson equation* on the unit square has been considered, i.e.

$$\nabla^2 u = f, \qquad \text{in } \Omega = (0,1)\times(0,1), \tag{3.20a}$$
$$u = g, \qquad \text{on } \partial\Omega, \tag{3.20b}$$

with

$$\Omega^l = (0,\gamma_1)\times(0,\gamma_2), \qquad 0<\gamma_1<1, \qquad 0<\gamma_2<1. \tag{3.21}$$

The results, holding only for uniform grids where the standard finite difference discretisation is used, are summarised in the following theorem that sets an upper bound for the global discretisation error. We use the notation $|u^{(n)}(x,y)| = |\partial^n u/\partial x^n| + |\partial^n u/\partial y^n|$.

Theorem 3.2 *Consider the LDC algorithm applied to the model problem (3.20) with* $g = 0$*. Let* Ω^l *be defined as in (3.21). Assume that there is no safety region. Then the fixed point* $u_{H,h}$ *of the iteration satisfies*

$$\| u - u_{H,h} \|_\infty \leq \frac{13}{8}\max\{C_1, C_2\}h^2 + \frac{1}{8}C_3H^2 + 3D_1H^j. \tag{3.22}$$

The exponent j *in (3.22) is* 1 *or* 2 *if piecewise linear or piecewise quadratic interpolation is used on the interface, respectively. The constants* C_1*,* C_2*,* C_3 *and* D_1 *are defined by*

$$C_1 := c_1\max\{|u^{(4)}(x,y)|\,|\,(x,y)\in(0,\gamma_1-h)\times(0,\gamma_2-h)\}, \tag{3.23a}$$
$$C_2 := c_2\max\{|u^{(4)}(x,y)|\,|\,(x,y)\in\Omega^l\backslash(0,\gamma_1-2h)\times(0,\gamma_2-2h)\}, \tag{3.23b}$$
$$C_3 := c_3\max\{|u^{(4)}(x,y)|\,|\,(x,y)\in\Omega\backslash(0,\gamma_1-H)\times(0,\gamma_2-H)\}, \tag{3.23c}$$
$$D_1 := d_1\max\{|u^{(1+j)}(x,y)|\,|\,(x,y)\in\Gamma\}, \tag{3.23d}$$

in which c_1*,* c_2*,* c_3 *and* d_1 *are independent of* H*,* h *and* u*.*

A close look at (3.22) shows that the discretisation error on the composite grid consists of three different contributions: the first two terms on the right-hand side are the errors on the high and the low activity region, respectively; the third is the error due to the interpolation of the boundary conditions on the interface.

The constants C_1, C_2, on the one hand, and C_3, on the other hand, are a measure of the smoothness of the continuous function u, depending on the values of its fourth derivatives in the high activity area, close to the interface and in the low activity area, respectively. Since we have assumed u to have large gradient on Ω^l, we expect the constant C_1 to be larger than the others.

The constant D_1 depends either on the second or on the third derivatives of u. The former case occurs if interface conditions are interpolated with a piecewise linear operator; the latter if they are interpolated with a piecewise quadratic operator. Numerical experiments have been performed in [21] to study the influence of the interface conditions interpolation on the composite grid discretisation error. In particular, it has been

shown that for a non-smooth function the errors obtained by using linear and quadratic interpolation are very close. This depends on the fact that the first two terms on the right-hand side of (3.22) are usually dominant with respect to the third one and implies that, in the most of the applications, the implementation of a linear interpolation operator is suitable to satisfy the accuracy requirements. The choice of using linear interpolation for the numerical experiments presented in the following chapters is based on the former results.

A last important theorem has been presented in [3]. There, a new expression for the iteration matrix $M : \mathcal{F}(\Gamma_H) \to \mathcal{F}(\Gamma_H)$ of the LDC iteration is introduced. It reads

$$M := \begin{pmatrix} 0 & I & 0 \end{pmatrix} (L_H)^{-1} \begin{pmatrix} I \\ 0 \\ 0 \end{pmatrix} \chi_{\Omega_H^s} \left[\mathcal{B}_H^{l,\Gamma} - L_H^l \mathcal{R}^{H,h} (L_h)^{-1} \mathcal{B}_h^{l,\Gamma} \mathcal{P}^{h,H} \right], \quad (3.24)$$

and gives a measure of the extent of the variation for the interface coarse grid function values as consequence of one LDC iteration. Then an upper bound for the norm of $\mathcal{M}$ is derived. The model problem considered is the *Poisson equation* (3.20) on the unit square and with a single refinement area.

The chosen computational grid Ω_H is uniform, with grid sizes $\Delta x = \Delta y = H$ and $N := 1/H$ integer, i.e.

$$\Omega_H = \{(lH, jH) \,|\, l = 1, 2, \cdots, N-1, \quad j = 1, 2, \cdots, N-1\}, \quad (3.25)$$

as well as the local grid Ω_h^l

$$\Omega_h^l = \{(lh, jh) \,|\, l = 1, 2, \cdots, n-1, \quad j = 1, 2, \cdots, n-1\}. \quad (3.26)$$

Theorem 3.3 *Consider the LDC algorithm for the Poisson problem (3.20) with the following settings. Let $\mathcal{L}_H$ be the standard five-point discretisation of the Laplacian on a uniform grid with grid sizes $\Delta x = \Delta y = H$ as in (3.25). Let $\Omega^l = (0, \gamma) \times (0, \gamma)$ with $0 < \gamma < 1$ and γ a multiple of H. Let $\mathcal{L}_h$ be the standard five-point discretisation of the Laplacian on a uniform grid with grid sizes $\Delta x = \Delta y = h$ as in (3.26). Let $\mathcal{P}^{h,H}$ be trigonometric interpolation. Finally let*

$$\Omega_H^{def} = \{(x, y) \in \Omega_H^l | x < \gamma - \epsilon \wedge y < \gamma - \epsilon\}, \quad (3.27)$$

for some $\epsilon > 0$, ϵ independent of H. Then the following upper bound for the norm of the iteration matrix $\mathcal{M}$ of the LDC algorithm holds

$$\|\mathcal{M}\|_\infty \leq CH^2 + Dh^2, \quad (3.28)$$

with C, D independent of H and h.

We note that there are two important elements upon which the proof of this theorem relies: the use of a safety region and the choice of a trigonometric operator to interpolate boundary conditions on the interface between the coarse and the fine grid. Although the first one is essential to determine the asymptotic behaviour of the iteration matrix, numerical experiments have shown that convergence is very fast even when $\epsilon = 0$. Therefore, no safety region is considered to solve the BVP that are presented in this thesis. The trigonometric interpolation, instead, is claimed to be an important tool for the proof of the theorem, but not essential as far as practical results are concerned. This assumption is confirmed in [3] by means of numerical experiments.

3.3 LDC and curvilinear grids

The results presented in the previous section, even though very important from a theoretical point of view, refer to a standard configuration consisting of a global and a local grid that are both rectangular. In this section we broaden the use of the LDC method showing why and how to couple different grid types.

The possibility of using a local fine grid that follows the shape of the solution in the *high activity* region has already been used in [43]. There, a rectangular global grid has been combined with a local circular grid, see Figure 3.3, to capture the behaviour of a function that exhibits local rotational symmetry. This technique has several advantages. In

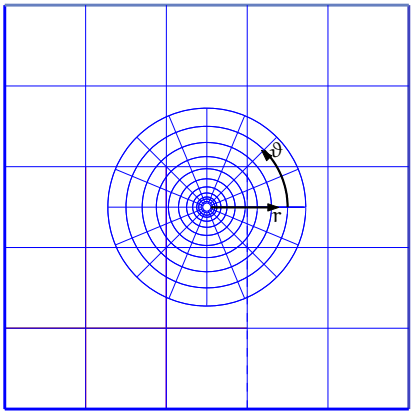


Figure 3.3: Global square and local circular grids.

particular, the concentration of grid lines can vary, being higher where the solution has steep gradients. Looking at Figure 3.3, we see that the local grid consists of two sets of curvilinear coordinates, r and ϑ, varying in the intervals $[r_1, r_2]$ and $[0, 2\pi]$, respectively. If the origin of the two coordinate systems coincides, the relation between the Cartesian and the curvilinear coordinates is given by

$$x(r, \vartheta) = r\cos\vartheta, \tag{3.29a}$$

$$y(r, \vartheta) = r\sin\vartheta. \tag{3.29b}$$

We notice that the coordinate ϑ is constant along the radii of the local domain, where r varies monotonically. The opposite situation occurs along the concentric circles. The curvilinear coordinates are thus independent of each other and can be represented as a rectangular domain. This is a crucial point, implying that all discretisation techniques used for rectangular grids can be applied also here. The coordinate system (r, ϑ) can also be normalized by introducing a new system (ξ, η), such that the local BVP is then solved in a computational domain where, unlike in the physical space, the step sizes $\Delta\xi$ and $\Delta\eta$ are unitary, see Figure 3.4.

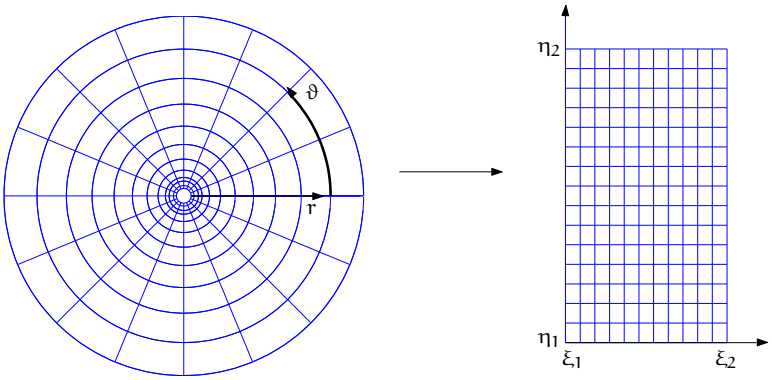


Figure 3.4: Physical and computational domain.

Usually, the shape of a BVP solution has a more complex behaviour than rotational symmetry. The general transformation between physical and computational space is expressed by

$$T : \begin{cases} x = x(\xi, \eta), \\ y = y(\xi, \eta), \end{cases} \tag{3.30}$$

where T is a one-to-one mapping. This guarantees that only one (ξ, η)-point corresponds to each (x, y)-point.

3.3.1 Generating a solution-fitted orthogonal grid

Now we consider the problem of numerically generating a local grid that fits the characteristics of an unknown function. We also require the grid to be orthogonal. This property turns out to be very beneficial for the accuracy of the solution. In fact, departure from orthogonality introduces truncation errors in the expression of the discretised differential operators. Moreover, the transformation of the BVP from the physical to the computational domain introduces some additional terms in the equations. Orthogonal coordinate systems minimize this effect. In the following we show how to construct an orthogonal fine grid, able to capture the detailed structure of the unknown function, by exploiting the coarse grid solution. We employ the method that has been presented in [16]. Let us briefly explain the main features of this technique and tailor it to our purposes.

The starting point is a non-orthogonal curvilinear coordinate system: one family of non-orthogonal coordinate lines will be kept, the other one will be transformed into a set of lines orthogonal to the first set. We will restrict ourselves to 2D domains. Consider the position vector $\mathbf{r} = \mathbf{r}(\xi, \eta) = (x(\xi, \eta), y(\xi, \eta))$ in a non-orthogonal coordinate system. Let us introduce the covariant base vectors $\partial\mathbf{r}/\partial\eta$, $\partial\mathbf{r}/\partial\xi$, tangent to the ξ- and η-lines, respectively, and the contravariant base vectors $\nabla\xi$, $\nabla\eta$ perpendicular to the ξ- and η-lines, respectively; see Figure 3.5. The relations between the covariant and the

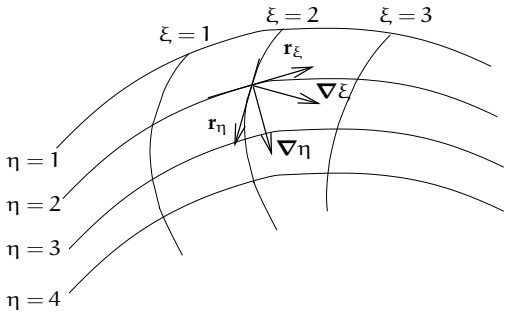


Figure 3.5: Covariant and contravariant base vectors.

contravariant base vectors are given by

$$\nabla\xi = \frac{1}{J^2}\Big[g_{\eta\eta}\frac{\partial \mathbf{r}}{\partial \xi} - g_{\xi\eta}\frac{\partial \mathbf{r}}{\partial \eta}\Big], \tag{3.31a}$$

$$\nabla\eta = \frac{1}{J^2}\Big[-g_{\xi\eta}\frac{\partial \mathbf{r}}{\partial \xi} + g_{\xi\xi}\frac{\partial \mathbf{r}}{\partial \eta}\Big], \tag{3.31b}$$

where the Jacobian, J, is defined as

$$J := \begin{vmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{vmatrix}, \tag{3.32}$$

and represents the ratio between the areas of an infinitesimal element evaluated in the physical and in the computational space. The coefficients $g_{\eta\eta}$, $g_{\xi\eta}$ and $g_{\xi\xi}$ are given by

$$g_{\eta\eta} := \Big|\frac{\partial \mathbf{r}}{\partial \eta}\Big|^2, \qquad g_{\xi\eta} := \Big(\frac{\partial \mathbf{r}}{\partial \xi}, \frac{\partial \mathbf{r}}{\partial \eta}\Big), \qquad g_{\xi\xi} := \Big|\frac{\partial \mathbf{r}}{\partial \xi}\Big|^2. \tag{3.33}$$

Suppose that we want to retain the η-lines and that we want to build a family of coordinate lines, say the ζ-lines, perpendicular to them. To do so, we introduce a function $k(\xi, \eta)$, constant along the new ζ-lines. Since the covariant base vector $\partial \mathbf{r}/\partial \xi$ is tangent to the η-lines, the orthogonality condition between the ζ-lines and the η-lines can be expressed as

$$\nabla k \times \frac{\partial \mathbf{r}}{\partial \xi} = 0, \tag{3.34}$$

where ∇k is related to the contravariant base vectors by

$$\nabla k = \frac{\partial k}{\partial \xi}\nabla\xi + \frac{\partial k}{\partial \eta}\nabla\eta. \tag{3.35}$$

Using (3.31), we can express ∇k in terms of the covariant base vectors. If we subsequently insert (3.35) into (3.34), we get the hyperbolic equation

$$\frac{\partial k}{\partial \eta} + F(\xi, \eta)\frac{\partial k}{\partial \xi} = 0, \tag{3.36a}$$

with $F(\xi,\eta)$ given by

$$F(\xi,\eta) = -\frac{\left(\frac{\partial \mathbf{r}}{\partial \xi}, \frac{\partial \mathbf{r}}{\partial \eta}\right)}{\left|\frac{\partial \mathbf{r}}{\partial \xi}\right|^2} = -\frac{g_{\xi\eta}}{g_{\xi\xi}}. \tag{3.36b}$$

The characteristics of (3.36a), i.e. the sought ζ-lines, satisfy the ODE system

$$\frac{d\xi}{d\eta} = F(\xi,\eta), \tag{3.37a}$$

$$\frac{dk}{d\eta} = 0. \tag{3.37b}$$

To integrate equation (3.36a), we use a second order accurate scheme, centred around the point $(\xi,\eta+1/2)$. This choice is based on the fact that the accuracy of the location of the fine grid points affects the value of the coefficients $\partial x/\partial\xi$, $\partial y/\partial\xi$, etc. Since such coefficients have to be used to discretise our BVP in curvilinear coordinates, too big a deviation from orthogonality would spoil the local grid solution. If I is the number of non-orthogonal ξ-lines, the derivatives of the function k are approximated as follows

$$\begin{aligned}\frac{\partial k}{\partial \xi}\left(\xi,\eta+\tfrac{1}{2}\right) \doteq \frac{1}{4}\big(&k(\xi+1,\eta+1) - k(\xi-1,\eta+1) + \\ &k(\xi+1,\eta) - k(\xi-1,\eta)\big), \qquad \xi = 2,3,\ldots,I-1,\end{aligned} \tag{3.38a}$$

$$\frac{\partial k}{\partial \eta}\left(\xi,\eta+\tfrac{1}{2}\right) \doteq k(\xi,\eta+1) - k(\xi,\eta), \qquad \xi = 1,2,3,\ldots,I. \tag{3.38b}$$

Note that (3.38) does not hold for $\xi = 1$ and $\xi = I$. In order to complete the equation set, we have to consider the kind of grid we are going to generate. In this thesis we will work with η-lines that are not closed, i.e. $\mathbf{r}(1,\eta) \neq \mathbf{r}(I,\eta)$, so the derivative $\partial k/\partial\xi$ in $\xi = 1$ and $\xi = I$ has to be discretised by a one-sided formula that is at least second order accurate, say

$$\begin{aligned}\frac{\partial k}{\partial \xi}\left(I,\eta+\tfrac{1}{2}\right) \doteq \frac{1}{4}\big(&3k(I,\eta+1) - 4k(I-1,\eta+1) + k(I-2,\eta+1) \\ &+ 3k(I,\eta) - 4k(I-1,\eta) + k(I-2,\eta)\big).\end{aligned} \tag{3.39}$$

A similar expression holds for $\xi = 1$. If we substitute (3.38) and (3.39) into (3.36a), we

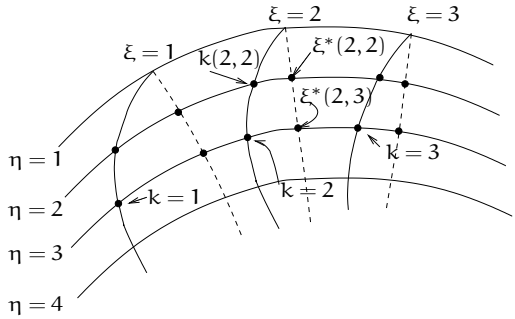


Figure 3.6: Coordinate systems (ξ, η) (solid lines) and (ζ, η) (dashed lines).

obtain

$$-\frac{F}{4}k(\xi-1,\eta+1)+k(\xi,\eta+1)+\frac{F}{4}k(\xi+1,\eta+1)= \\ \frac{F}{4}k(\xi-1,\eta)+k(\xi,\eta)-\frac{F}{4}k(\xi+1,\eta), \qquad \xi=2,3,...,I-1, \tag{3.40a}$$

$$\left(1+\frac{3F}{4}\right)k(I,\eta+1)-F\,k(I-1,\eta+1)+\frac{F}{4}k(I-2,\eta+1)= \\ \left(1-\frac{3F}{4}\right)k(I,\eta)+F\,k(I-1,\eta)-\frac{F}{4}k(I-2,\eta), \qquad \xi=I, \tag{3.40b}$$

$$\left(1-\frac{3F}{4}\right)k(1,\eta+1)+F\,k(2,\eta+1)-\frac{F}{4}k(3,\eta+1)= \\ \left(1+\frac{3F}{4}\right)k(1,\eta)-F\,k(2,\eta)+\frac{F}{4}k(3,\eta), \qquad \xi=1, \tag{3.40c}$$

where $F := F\left(\xi, \eta+\frac{1}{2}\right)$. The derivatives that appear in (3.36b) are computed numerically, using the same scheme. Let us now briefly describe the procedure to solve system (3.40).

We introduce and compute a function $\xi^*(\xi,\eta)$. It represents the new ξ-value to which the point (ξ,η) must be displaced on an η-line to get an orthogonal trajectory. Suppose we know $\xi^*(\xi,\eta)$ and we want to determine $\xi^*(\xi,\eta+1)$, see Figure 3.6, where the initial conditions are

$$\xi^*(\xi,1)=\xi, \qquad \xi=1,2,...,I. \tag{3.41}$$

We set

$$k(\xi,\eta+1)=\xi, \qquad \xi=1,2,...,I, \tag{3.42}$$

and solve system (3.40) by a backward step. This provides the values of k at all points (ξ,η). Then we know

- k on the (ξ,η)-points from system (3.40);

- $\xi^*(\xi,\eta)$ from the previous solution step.

With these elements, the values of $k(\xi^*,\eta)$ can be found by interpolation: supposing that the point (ξ^*,η) lies between (ξ',η) and $(\xi'+1,\eta)$, the following four point Lagrangian interpolation

$$\begin{aligned} k(\xi^*,\eta) = &-\frac{p(p-1)(p-2)}{6}k(\xi'-1,\eta)+\frac{(p^2-1)(p-2)}{2}k(\xi',\eta) \\ &-\frac{p(p+1)(p-2)}{2}k(\xi'+1,\eta)+\frac{p(p^2-1)}{6}k(\xi'+2,\eta), \end{aligned} \tag{3.43}$$

with $p := \xi^* - \xi'$ is used. It has to be noticed that the intervals in ξ are uniform: this guarantees the accuracy of the interpolation.

Since k is constant on the orthogonal trajectories, $k(\xi^*(\xi,\eta),\eta) = k(\xi^*(\xi,\eta+1),\eta+1)$ holds. Furthermore, because of (3.42), $k(\xi,\eta+1)=\xi$. This implies

$$\xi^*(\xi,\eta+1) = k(\xi^*(\xi,\eta+1),\eta+1) = k(\xi^*(\xi,\eta),\eta). \tag{3.44}$$

By exploiting the knowledge of the Cartesian coordinates of the $(\xi,\eta+1)$-points and the values of $k(\xi^*(\xi,\eta+1),\eta+1)$ and $k(\xi,\eta+1)$, it is possible to compute the Cartesian coordinates of the (ζ,η)-points. In fact, assuming that the point $(\xi^*(\xi,\eta+1),\eta+1)$ is between $(\xi',\eta+1)$ and $(\xi'+1,\eta+1)$, a formula similar to (3.43) leads to

$$\begin{aligned} x(\xi^*(\xi,\eta+1),\eta+1) = &-\frac{q(q-1)(q-2)}{6}x(\xi'-1,\eta+1)+\frac{(q^2-1)(q-2)}{2}x(\xi',\eta+1) \\ &-\frac{q(q+1)(q-2)}{2}x(\xi'+1,\eta+1)+\frac{q(q^2-1)}{6}x(\xi'+2,\eta+1), \end{aligned} \tag{3.45}$$

where $q := k(\xi^*(\xi,\eta+1),\eta+1) - k(\xi',\eta+1)$. The y-coordinates of the $(\zeta,\eta+1)$ points are computed in a similar way. The lines defined by the points (ξ^*,η) are the sought ζ-lines. We denote with N_η the number of η-lines. Thus, the grid generation algorithm reads

- Initialisation
  - Define two sets of coordinate lines: the η-lines, that will be kept, and the ξ-lines, that will be used as support;
  - Compute the function $F(\xi,\eta)$ as in (3.36b);
  - Set the initial conditions on ξ^* as in (3.41);
- Iteration $i = 2,3,\cdots,N_\eta$
  - Set $k(\xi,i)=\xi$ and, using system (3.40), compute $k(\xi,i-1)$;
  - Compute $k(\xi^*,i-1)$ by interpolation with (3.43);
  - Put $\xi^*(\xi,i) = k(\xi^*,i-1)$;

- Compute $x(\xi^*(\xi, \eta + 1), i)$ and $y(\xi^*(\xi, \eta + 1), i)$ by interpolation with (3.45).

The outlined algorithm can be applied to a special set of η-lines: the iso-curves of the coarse grid solution. This approach offers the possibility to decrease the complexity of the local BVP. There are two reasons for that. The first is that gradients along level curves are almost equal to zero. Secondly, gradients decrease along the lines perpendicular to the level curves when moving away from the high activity region. In both cases the solution allows for a bigger grid size. This hold for all phenomena, e.g. combustion, described by functions that have similar iso-contour patterns.

CHAPTER 4

LDC with orthogonal grids

The LDC technique introduced in the previous chapter is applied here to a variety of convection-diffusion-reaction problems that involve local Cartesian or curvilinear grids. In Section 4.1 we discuss the general form of the model problem that we deal with throughout the whole chapter. The first numerical experiments make use of a rectangular fine grid that is *slanted* with respect to the direction of the global axes: this requires a transformation of the PDEs. Special attention is given to the boundary conditions. Furthermore, some results relating the steepness of the solution in the high activity area to the features of the fine grid are given. In Section 4.2 numerical results for several settings that differ either for the shape of the solution or for its orientation with respect to the global domain are presented. The complexity of the LDC algorithm is addressed in Section 4.3. Here two different solution strategies are compared in terms of memory requirements: the tensor product grid approach and the LDC method in combination with slanting grids. In Section 4.4 the issue of combining LDC with curvilinear orthogonal grids is introduced and the diffusion and convection differential operators are expressed in terms of curvilinear coordinates. The relevant numerical experiments are presented in Section 4.5. In Section 4.6 the complexity of LDC with curvilinear grid is assessed.

4.1 LDC in combination with slanting grids

In this chapter we will focus on the application of LDC to a sample problem having a solution characterised by a gradient across a line in the domain, which is not aligned with any of the coordinate axes. This setting is quite general. In fact, it is representative of a large class of real physical phenomena having the same behaviour: a high activity region concentrated in a narrow strip along a line.

Consider the following two-dimensional convection-diffusion-reaction problem (see [7])

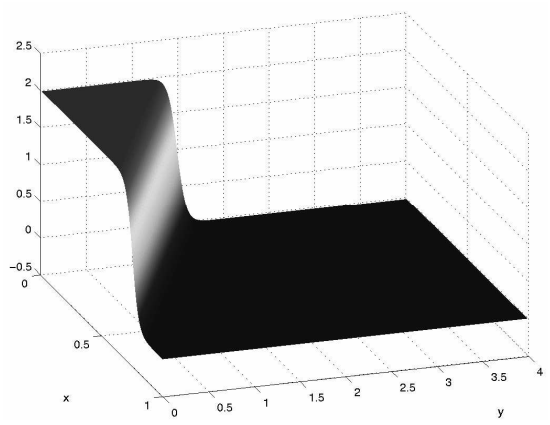


Figure 4.1: Function $u(\mathbf{x})$ in the domain $\Omega = (0,1) \times (0,4)$, with $\beta = 5$, $a = 4$, $b = 2$, $c = 3$.

$$-\nabla^2 u + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega := (0, l_1) \times (0, l_2), \tag{4.1a}$$

$$u = g(\mathbf{x}) := 1 - \tanh[\beta s(\mathbf{x})], \qquad \mathbf{x} \in \partial\Omega, \tag{4.1b}$$

where f is such that the exact solution of (4.1) is given by

$$u(\mathbf{x}) = 1 - \tanh[\beta s(\mathbf{x})]. \tag{4.2}$$

In the first examples we use a function $s(\mathbf{x}) = ax + by - c$ with $a, b, c > 0$. The function $u(\mathbf{x})$ is approximately equal to 2 in the lower left corner of the domain, where $s(\mathbf{x}) < 0$, and its value decreases steeply over the centre line $s(\mathbf{x}) = 0$, becoming 0 in the rest of the domain, where $s(\mathbf{x}) > 0$, see Figure 4.1. The parameter β determines the steepness of $u(\mathbf{x})$ in the vicinity of the centre line. Our purpose is to apply LDC in combination with a slanting local grid enclosing that line.

In order to apply LDC just across this area, two choices are possible. One is to use a few overlapping rectangular fine grids, aligned with the coordinate axes and arranged in a staircase shape. The drawback of this method is that redundant grid points are needed. This makes the procedure quite expensive in terms of memory requirements and CPU time. Moreover, since more than one local fine grid must be used, the convergence of the LDC method might slow down, see [2]. Another possibility is to cover the high activity region with one single fine grid, which is slanted with respect to the x-axis of the global domain. The advantages of the method are that only one single fine grid is needed and the number of fine grid points is considerably reduced. The only, though small, disadvantage is that the fine grid is not aligned with the coarse grid: this makes it more difficult to implement.

To compute the numerical solution of BVP (4.1) using LDC, we need to specify the following:

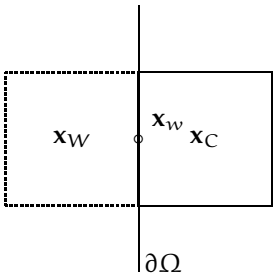


Figure 4.2: $\mathbf{x_W}$ is the virtual grid point for the approximation of boundary conditions.

- The coarse grid and the discretisation of (4.1) on it;
- The fine grid and the discretisation of (4.1) on it;
- An interpolation operator to compute the interface conditions;
- A restriction operator.

Our first step will be the discretisation of (4.1a) in Ω. The finite volume method (FVM) on a *cell centred* grid leads to a second order accurate scheme, that for a general coarse grid point $\mathbf{x}_C$, reads

$$-\frac{u_E - 2u_C + u_W}{H^2} - \frac{u_N - 2u_C + u_S}{H^2} + \frac{u_E - u_W}{2H} + \frac{u_N - u_S}{2H} = f_C. \tag{4.3}$$

The fluxes are approximated by central differences. The subscripts indicate the locations where the terms are evaluated, i.e. u_C is the approximation of the function value $u(\mathbf{x}_C)$, and N, S, W and E label the northern, southern, western and eastern neighbouring points of the central point C, respectively. Moreover, $f_C = f(\mathbf{x}_C)$.

Let us now investigate how to impose the Dirichlet boundary conditions (4.1b). The choice of a *cell centred* grid is such that it is not possible to assign a value on the boundary of the domain, because it contains no grid points. This problem can be solved by introducing *virtual* grid points, as in Figure 4.2. The Dirichlet boundary condition in this example is approximated by

$$\frac{1}{2}(u_W + u_C) = g(\mathbf{x}_w), \tag{4.4}$$

from which u_W can be expressed as a function of u_C and $g(\mathbf{x}_w)$. Using (4.3) in the internal grid points of Ω_H, we obtain a pentadiagonal algebraic system that can easily be solved.

The next step is to define the local problem (3.4). Obviously, a tensor product fine grid would cover a substantial part of the entire domain, resulting in almost a similar number of grid points as it would be obtained from the application of this type of grid on the entire domain. Instead, we solve the local problem (3.4) by using a slanting fine grid

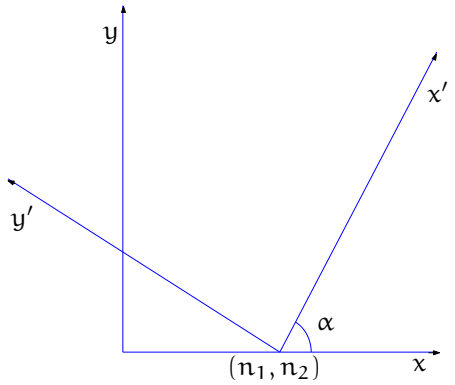


Figure 4.3: Frame of reference (x, y) and (x', y')

and thus trying to reduce the number of grid points needed to give a good representation of the unknown function. The domain Ω^l is defined such that it has to contain the line $s(\mathbf{x}) = 0$ completely; furthermore some redundant points are needed to make the data structure simpler, see Section 4.2. To do so, we have to carry out a coordinate transformation. Suppose that the x'-axis of the frame of reference (x', y') is rotated over an angle α with respect to the x-axis of the frame of reference (x, y) and that its origin coincides with the point having coordinates (n_1, n_2), see Figure. 4.3. The transformation that relates (x, y) and (x', y') is clearly given by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{R}^T \left[\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} \right], \tag{4.5}$$

with $\mathbf{R}$ the orthogonal matrix

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{4.6}$$

From now on, we will use the single quotation mark to indicate differential operators in (x', y'). Let us introduce the chain rule for scalar and for vector quantities, viz.

$$\boldsymbol{\nabla} u = \mathbf{R} \boldsymbol{\nabla}' u, \tag{4.7a}$$
$$\boldsymbol{\nabla} \cdot \mathbf{v} = \mathbf{R} \boldsymbol{\nabla}' \cdot \mathbf{v}. \tag{4.7b}$$

Applying relations (4.7) to the Laplacian on the right-hand side of (4.1a), we get

$$\nabla^2 u = \boldsymbol{\nabla} \cdot (\boldsymbol{\nabla} u) = (\mathbf{R}\boldsymbol{\nabla}') \cdot (\mathbf{R}\boldsymbol{\nabla}' u) = \boldsymbol{\nabla}' \cdot (\mathbf{R}^T \mathbf{R} \boldsymbol{\nabla}' u) = \boldsymbol{\nabla}' \cdot (\boldsymbol{\nabla}' u) = (\nabla^2)' u, \tag{4.8}$$

where $(\nabla^2)' u := \frac{\partial^2 u}{\partial x'^2} + \frac{\partial^2 u}{\partial y'^2}$. The convection terms in (4.1a) can be written as

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = (\mathbf{a} \cdot \boldsymbol{\nabla}) u, \qquad \text{with } \mathbf{a} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.9}$$

Expressing (4.9) in the new frame of reference, we get

$$(\mathbf{a}\cdot\boldsymbol{\nabla})u = (\mathbf{a}\cdot\mathbf{R}\boldsymbol{\nabla}')u = (\mathbf{R}^{T}\mathbf{a}\cdot\boldsymbol{\nabla}')u = (\mathbf{b}\cdot\boldsymbol{\nabla}')u, \tag{4.10}$$

with

$$\mathbf{b} = \begin{pmatrix}\cos\alpha + \sin\alpha\\ \cos\alpha - \sin\alpha\end{pmatrix}. \tag{4.11}$$

Substitution of (4.8) and (4.11) into (4.1a) yields

$$-(\nabla')^2 u + (\mathbf{b}\cdot\boldsymbol{\nabla}')u = f. \tag{4.12}$$

The method used to discretise the global problem on the coarse grid will be applied to the discretisation of (4.12) on the fine grid as well. We will choose a step size equal to h along the x'-axis and a step size equal to mh along the y'-axis, with $m \geq 1$. Actually, since $\partial u/\partial y' = 0$, it is possible to use a coarser grid size along the y'-axis without spoiling the accuracy of the solution. The discretisation of (4.12) yields

$$\begin{aligned}&-\frac{u_E - 2u_C + u_W}{h^2} - \frac{u_N - 2u_C + u_S}{(mh)^2} + (\cos\alpha + \sin\alpha)\frac{u_E - u_W}{2h}\\ &+ (\cos\alpha - \sin\alpha)\frac{u_N - u_S}{2mh} = f_C,\end{aligned} \tag{4.13}$$

where C is a general fine grid point and W, E, N and S are its neighbours.

One of the most important steps of the LDC algorithm is the interpolation of the interface condition on Γ_h: this is done using a bi-linear interpolation operator. This choice relies on the results of Theorem 3.2: the bi-linear interpolation has the same order of accuracy as the discretisation operator used in the global and the local problems. This is confirmed by the numerical experiments in the next section. Suppose that the interface condition is needed at a virtual grid point V. Since we use cell centred discretisation also for the local problem, V does not belong to the interface: it is the mirror point with respect to Γ of a grid point belonging to Ω_h^l. The operator $\mathcal{P}^{h,H}$ determines u_V by interpolation from the four nearest neighbours on the coarse grid. With reference to Figure 4.4, it has the following form

$$u_V = \alpha_A u_A + \beta_B u_B + \gamma_C u_C + \delta_D u_D, \tag{4.14}$$

with the coefficients α_A, β_B, γ_C and δ_D given by

$$\alpha_A = \frac{1}{h_1 h_2}(x_D - x_V)(y_V - y_C), \tag{4.15a}$$

$$\beta_B = \frac{1}{h_1 h_2}(x_V - x_C)(y_V - y_C), \tag{4.15b}$$

$$\gamma_C = \frac{1}{h_1 h_2}(x_C - x_V)(y_A - y_V), \tag{4.15c}$$

$$\delta_D = \frac{1}{h_1 h_2}(x_V - x_C)(y_V - y_A), \tag{4.15d}$$

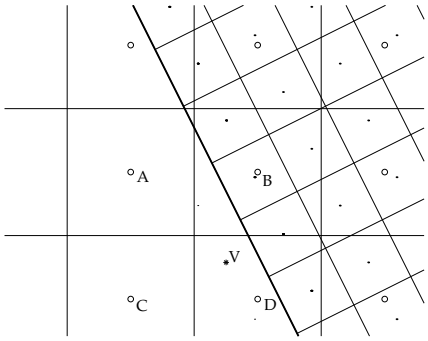


Figure 4.4: At the virtual grid point V, the interface condition is interpolated from the coarse grid points A, B, C and D. The slanting bold line represents the interface between Ω^l and Ω.

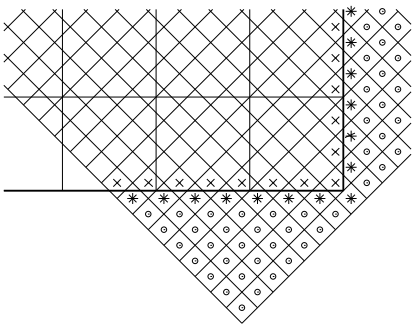


Figure 4.5: Detail of the global domain boundary (bold line) of the $\times$-, $*$-, $\odot$-fine grid points.

and

$$h_1 = x_D - x_C, \qquad h_2 = y_A - y_C. \tag{4.16}$$

The use of the slanting fine grid, as depicted in Figure 4.8, presents an additional difficulty in that there are some points belonging to Ω_h^l that do not belong to the computational domain. As a consequence, $\Omega_h^l \not\subset \Omega$. To overcome this problem we have to recall that the solution at a point C is influenced by the four neighbouring points N, S, W and E, only. This means that we can divide the fine grid points that fall outside Ω in two groups: the points which affect the solution of the fine grid points inside Ω and the points that do not. An arbitrary value of the grid function can be assigned to the points belonging to the second group (points marked by '$\odot$' in Figure 4.5), since they have no relation with the grid function in the interior of the computational domain. The points in the first group (marked by '$*$' in Figure 4.5) belong to the stencil of the points inside the computational domain (marked by '$\times$') and can be linked to them by using a linear extrapolation of the boundary value function g. This allows to eliminate the '$*$'-point values from the '$\times$'-point equations. The matrix that we obtain has a nice banded pentadiagonal structure. In fact, if a '$*$'-point belongs to the stencil of two different '$\times$'-points, two different extrapolating equations are used, each extrapolation involving the boundary value g that lies at the intersection between the boundary and the segment connecting the '$*$'-point with the considered '$\times$'-point. In addition, we include an equation of the form $u_* = c$, where c is an arbitrary value, to preserve the pentadiagonal structure of the matrix, see Figure 4.6.

Once the local BVP has been solved, the grid functions (3.6) can be calculated. To do so, the restriction operator $\mathcal{R}^{H,h}$ has to be specified: also in this case we choose bilinear interpolation because it is computationally inexpensive and, at the same time, it guarantees a sufficient accuracy of the numerical results. Therefore, the operator $\mathcal{R}^{H,h}$ computes the function value $u(\mathbf{x})$ with $\mathbf{x} \in \Omega_H^l$ from the function values at the four

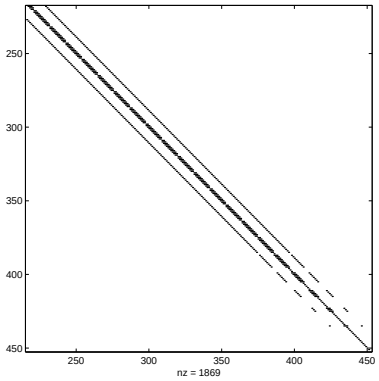


Figure 4.6: Sparsity pattern of the matrix (lower corner) used to solves the local BVP. The matrix has a banded pentadiagonal structure.

nearest fine grid points. It has the same form as $\mathcal{P}^{h,H}$ in (4.14), when the point V is replaced by a point belonging to Ω_H and A, B, C and D by four points belonging to Ω^l_h. Obviously, the interpolation has now to be performed in the frame of reference (x', y'). The subsequent steps are straightforward: once we know w_H we can compute, d_H to perform the last three steps of the LDC Algorithm.

In Chapter 3 the domain Ω^S_H is introduced: in Figure 4.7 its points are marked by '$\bullet$', while the grid points belonging to $\Omega^l_H \backslash \Omega^S_H$ are marked by '$*$'. They are defined in (3.8) noticing that the central difference discretisation uses a five-point stencil.

The following features are interesting to note: under certain conditions we can adapt the width of the local domain Ω^l to the steepness of the solution and thus we can make the number of fine grid points along the x'-axis independent of β. Suppose that $s(\mathbf{x}) = ax + by - c$. We choose the frame of reference (x', y') such that $u(x', y') = 1 - \tanh(\beta l x')$, with $l = \sqrt{a^2 + b^2}$. We first notice that the boundaries of the high activity area parallel to the y'-axis can be determined by requiring the function u to differ from its asymptotic value by more than a certain tolerance, say ϵ. Because the function value can be approximated by 0, when $ax + by > c$, and by 2, when $ax + by < c$, then $(x', y') \in \Omega^l$ if $\epsilon \leq u(x', y') \leq 2 - \epsilon$. Since $\operatorname{Artanh}(z) = \frac{1}{2} \ln\left(\frac{1+z}{1-z}\right)$, this translates into

$$-\frac{1}{2l\beta} \ln\left(\frac{2-\epsilon}{\epsilon}\right) \leq x' \leq \frac{1}{2l\beta} \ln\left(\frac{2-\epsilon}{\epsilon}\right). \tag{4.17}$$

The domain Ω^l is symmetric with respect to x' and its width goes to zero when β increases. If we consider the scaled coordinates

$$x'' := \beta x', \tag{4.18a}$$
$$y'' := \beta y', \tag{4.18b}$$

the inequalities (4.17) transform into

$$-\frac{1}{2l}\ln\left(\frac{2-\epsilon}{\epsilon}\right) \leq x'' \leq \frac{1}{2l}\ln\left(\frac{2-\epsilon}{\epsilon}\right). \tag{4.19}$$

Scaling equation (4.12) by using the coordinate transformation (4.18), we get

$$-(\nabla'')^2 u + \frac{1}{\beta}(\mathbf{b}\cdot\boldsymbol{\nabla}'')u = f_1(x'', y'') + \frac{1}{\beta}f_2(x'', y''), \tag{4.20}$$

where the double prime on the differential operators indicates derivatives with respect to the variables x'' and y''. Furthermore $u(x'', y'')$, unlike $u(x, y)$ in (4.12), does not depend explicitly on β anymore. The error that one makes discretising (4.20) is bounded by the sum of the two terms

$$\frac{1}{12}\left(\left|\frac{\partial^4 u}{\partial x''^4}\right| + \left|\frac{\partial^4 u}{\partial y''^4}\right|\right)(h'')^2 \qquad \text{and} \qquad \frac{1}{6\beta}\left(\left|\frac{\partial^3 u}{\partial x''^3}\right| + \left|\frac{\partial^3 u}{\partial y''^3}\right|\right)(h'')^2, \tag{4.21}$$

where h'' is the step size in the frame of reference (x'', y''). Since the second term is negligible compared to the first one, the error bound turns out to be independent of β and proportional to $(h'')^2 = (\beta h)^2$. From this follows $h = C/\beta$, where C is independent of β. Let the number of the fine grid points along the x'-axis be equal to $M + 1$. Using (4.17), we get

$$M = \frac{1}{l\beta h}\ln\left(\frac{2-\epsilon}{\epsilon}\right) = \frac{1}{lC}\ln\left(\frac{2-\epsilon}{\epsilon}\right). \tag{4.22}$$

Therefore, if β varies, the domain Ω^1 can be stretched or compressed, according to (4.18a), while the (x'', y'')-domain is kept constant, as well as the number of grid points along the x'-axis.

The elements that are used to show this are the fact that u is invertible, i.e. it is monotonically increasing or decreasing with respect to x', and the fact that $u(x, y) = u(\beta x')$. Furthermore, we have assumed that the error bound of the convective term in (4.20) is small, as β increases, with respect to the error bound of the diffusion term. These results can thus be extended to other functions that satisfy the same requirements.

4.2 Slanting grids: numerical results

In this section we consider three examples.

4.2.1 Example 1: straight centre line, no restriction error.

We take $s(\mathbf{x}) = x + y - 1$ and $\beta = 20$ on the square domain $\Omega = (0, 1) \times (0, 1)$. We note that the high activity region is centred around a straight line, inclined at an angle equal to $\frac{3\pi}{4}$: therefore the most natural choice is to put $\alpha = \frac{\pi}{4}$ and $(n_1, n_2) = (1, 0)$ Furthermore the vector $\mathbf{b}$ in (4.12) becomes

$$\mathbf{b} = \begin{pmatrix}\sqrt{2}\\ 0\end{pmatrix}. \tag{4.23}$$

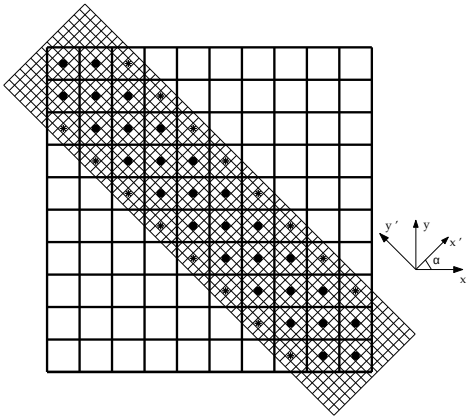


Figure 4.7: Slanting fine grids. The points belonging to Ω_H^S are marked by '$\bullet$'; the points belonging to $\Omega_H^l \setminus \Omega_H^S$ are marked by '$*$'.

We choose $m = 1$. Since $\alpha = \frac{\pi}{4}$, it is possible to arrange the fine grid in such a way that each point of Ω_H^l coincides with a point of Ω_h^l. This happens when the line $x + y = 1$ coincides with a diagonal of the fine grid and $h = \frac{H}{k\sqrt{2}}$, with k an integer. In this case the restriction operator simplifies to

$$\mathcal{R}^{H,h} u_h(x, y) := u_h(x, y), \qquad \text{for } \mathbf{x} \in \Omega_H^l. \tag{4.24}$$

Table 4.1 shows the errors $\|u^* - u_{H,h}\|_\infty$ of the composite grid solution, and the ratio between the errors, for several values of H, u^* being the exact solution of (4.1). It turns out that convergence is achieved already after one iteration of the LDC algorithm. The width of the fine grid, centred around the line $x + y = 1$, is kept almost constant and equal to 0.39. Even if interface conditions are interpolated using a bi-linear operator, the solution approximation is asymptotically second order accurate with respect to h, provided that H is small enough. Note also that, for h fixed and H decreasing, the error is approximately constant. In fact, in this situation, the effect of the discretisation errors on the coarse grid are negligible compared to the local discretisation errors on the fine grid. The results reported in Table 4.2 show that, when β varies, the number of fine grid

h	$H = 10^{-1}$	ratio	$H = 20^{-1}$	ratio	$H = 40^{-1}$	ratio
$\frac{H}{2\sqrt{2}}$	$4.68 \cdot 10^{-2}$		$1.72 \cdot 10^{-2}$		$4.10 \cdot 10^{-3}$	
$\frac{H}{4\sqrt{2}}$	$1.99 \cdot 10^{-2}$	2.4	$4.20 \cdot 10^{-3}$	4.1	$1.10 \cdot 10^{-3}$	3.7
$\frac{H}{8\sqrt{2}}$	$6.90 \cdot 10^{-3}$	2.9	$1.10 \cdot 10^{-3}$	3.8	$2.600 \cdot 10^{-4}$	4.2

Table 4.1: Composite grid approximation error $\|u^* - u_{H,h}\|_\infty$.

points along the x'-axis can be kept constant without affecting the error of the fine grid

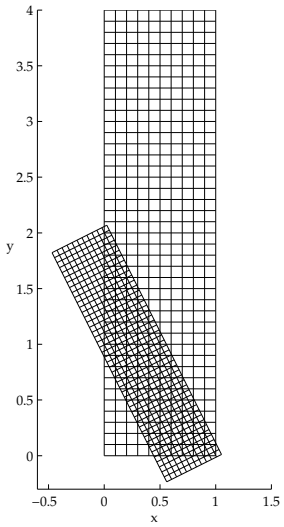


Figure 4.8: Coarse and slanting fine grids.

solution; see Section 3. The width of the local domain is computed using (4.17), with ϵ approximately equal to 10^{-5}. It turns out that, for a fixed H, the error is almost constant when different values of β are considered and when the local domain is correspondingly stretched. This approach fails for large values of H and β, because, in this case, the domain Ω_H^l does not contain enough points to apply the correction.

H	$\beta = 10$	$\beta = 20$	$\beta = 40$
10^{-1}	$1.81 \cdot 10^{-2}$	$1.99 \cdot 10^{-2}$	$7.10 \cdot 10^{-1}$
20^{-1}	$4.30 \cdot 10^{-3}$	$4.20 \cdot 10^{-3}$	$6.70 \cdot 10^{-3}$
40^{-1}	$1.10 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$

Table 4.2: Error $||u^* - u_h||_\infty$ for different values of H and β; $h = \frac{5H}{\beta\sqrt{2}}$.

4.2.2 Example 2: straight centre line

Consider problem (4.1) in the domain $\Omega = (0,1) \times (0,4)$, with $s(\mathbf{x}) = 4x + 2y - 3$ and $\beta = 5$. The high activity region lies across the straight line $4x + 2y = 3$. In Figure 4.8 we depict the grids that are used to solve (4.1), when $H = 10^{-1}$ and $h = 20^{-1}$. Suppose that we want to compute the function u within an accuracy of 10^{-3}. Taking into account

that the discretisation error is approximately bounded by

$$\frac{1}{12}(C_1 + C_2)H^2, \quad \text{with} \quad C_1 = \max\left|\frac{\partial^4 u}{\partial x^4}\right| \quad \text{and} \quad C_2 = \max\left|\frac{\partial^4 u}{\partial y^4}\right|, \tag{4.25}$$

we can try to give an estimate of this quantity in order to choose a suitable grid size. For $(x, y) \in \Omega$ such that $4x + 2y < 1.6$ and $4x + 2y > 4.4$, we find that

$$\left|\frac{\partial^4 u}{\partial x^4}\right| + \left|\frac{\partial^4 u}{\partial y^4}\right| < 4.5. \tag{4.26}$$

Hence, we may define Ω^l as the region bounded by the lines $4x+2y = 1.6$, $4x+2y = 4.4$ and by the two lines $x - 2y = -4.4$ and $x - 2y = 1$. If we choose $H = 10^{-1}$, the discretisation error in $\Omega \backslash \Omega^l$ will be of the order of 10^{-3}. Actually, the dimensions of the refined area are slightly different from the one defined above, in order to make it a multiple of the chosen fine grid step size. The result of the computations performed by LDC are presented in Table 4.3. This gives the error of the composite grid solution when $H = 10^{-1}$. Table 4.4 shows the results obtained by solving the same problem with a uniformly refined grid. By comparing these two tables it is clear that the LDC algorithm is much more efficient with respect to the memory usage, because it gives a comparable accuracy with substantial less grid points. In fact, if we compare, for instance, the last row of Table 4.3 and Table 4.4, we see that we manage to reach a reduction factor, i.e. a ratio between number of points used by LDC and the uniform grid refinement, of circa 0.36.

Table 4.5 gives the error of the composite grid solution, for several values of H and h. It appears that, once the asymptotic behaviour has been reached, the solution is second order accurate with respect to h. The error for h fixed and H decreasing is again approximately constant; see Example 1.

Until now we have uniformly refined the high activity region: this assumption can be dropped if one realizes that the gradient of the continuous solution u is parallel to the x'-axis. This implies that a value $m > 1$ can be chosen. As a consequence, the number of fine grid points can be strongly reduced, while the accuracy of the approximated solution remains comparable. This is shown in Table 4.6, where the global errors of the approximation and the total number of grid points are given. The comparison of Table 4.6 and Table 4.3 shows that the solution on the non-uniform fine grid needs much less grid points, while having roughly the same accuracy. Moreover, by comparing the last row of Table 4.6 and Table 4.4, we see that the reduction factor is, in this case, approximately equal to 0.088.

4.2.3 Example 3: curved centre line

The last example shows that the properties of the LDC method outlined so far also hold when the function $s(\mathbf{x}) = 0$ is not a straight line, see Figure 4.9. We have chosen the shape of the high activity region such that it is similar to the shape of a Bunsen flame [7]. For the sake of simplicity, $s(\mathbf{x})$ is given in the frame of reference (x', y'), that is rotated

h	$\|u^* - u_{H,h}\|_\infty$	Grid points
20^{-1}	$6.72 \cdot 10^{-2}$	$400 + 517$
40^{-1}	$2.85 \cdot 10^{-2}$	$400 + 2162$
80^{-1}	$7.7 \cdot 10^{-3}$	$400 + 8883$

Table 4.3: Error of the composite grid solution. The last column shows the number of grid points. The global domain is discretised using 400 points ($H^{-1} = 10$).

H	$\|u^* - u_H\|_\infty$	Grid points
10^{-1}	$4.839 \cdot 10^{-1}$	400
20^{-1}	$8.06 \cdot 10^{-2}$	1600
40^{-1}	$2.22 \cdot 10^{-2}$	6400
80^{-1}	$5.70 \cdot 10^{-3}$	25600

Table 4.4: Global error obtained with uniform refinement. The last column shows the number of the grid points.

h	$H = 10^{-1}$	ratio	$H = 20^{-1}$	ratio	$H = 40^{-1}$	ratio
$H/2$	$6.72 \cdot 10^{-2}$		$2.95 \cdot 10^{-2}$		$7.00 \cdot 10^{-3}$	
$H/4$	$2.85 \cdot 10^{-2}$	2.36	$7.0 \cdot 10^{-3}$	4.2	$1.70 \cdot 10^{-3}$	4.1
$H/8$	$7.7 \cdot 10^{-3}$	3.6	$1.70 \cdot 10^{-3}$	4.1	$4.32 \cdot 10^{-4}$	3.9

Table 4.5: Composite grid approximation error $\|u^* - u_{H,h}\|_\infty$ for different values of H and h.

h	m	$\|u^* - u_{H,h}\|_\infty$	Grid points
20^{-1}	2	$6.71 \cdot 10^{-2}$	$400 + 231$
40^{-1}	4	$2.73 \cdot 10^{-2}$	$400 + 483$
80^{-1}	4	$8.20 \cdot 10^{-3}$	$400 + 1845$

Table 4.6: Error of the composite grid solution and number of grid points as a function of m ($H = 10^{-1}$).

with respect to (x, y) over an angle $\alpha = \arctan\left(\frac{1}{2}\right)$ and $(n_1, n_2) = (0.75, 0)$, see Section 3, and reads

$$s(x', y') = x' - \frac{1}{10} \sin \frac{2\pi y'}{1.677}. \tag{4.27}$$

A fine grid similar to the one depicted in Figure 4.8 is used to solve the local problem. A slightly bigger number of fine grid points along the x'-axis is needed for low values of H in order to reproduce the interface conditions with the desired accuracy. Table 4.7 shows the error of the composite grid solution after one LDC iteration. All considerations about the accuracy of the solution made in the previous examples are still applicable.

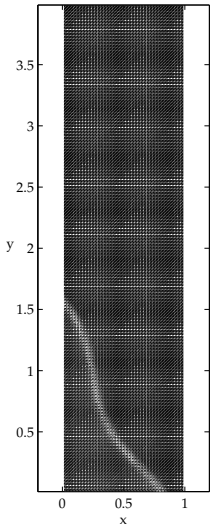


Figure 4.9: Curved high activity region.

h	$H = 10^{-1}$	ration	$H = 20^{-1}$	ration	$H = 40^{-1}$	ratio
$H/2$	$9.54 \cdot 10^{-2}$		$2.16 \cdot 10^{-2}$		$5.10 \cdot 10^{-3}$	
$H/4$	$2.63 \cdot 10^{-2}$	3.6	$5.10 \cdot 10^{-3}$	4.2	$1.20 \cdot 10^{-3}$	4.2
$H/8$	$6.30 \cdot 10^{-3}$	4.2	$1.30 \cdot 10^{-3}$	3.9	$3.57 \cdot 10^{-4}$	3.4

Table 4.7: Composite grid approximation error $\|u^* - u_{H,h}\|_\infty$ for different values of H and h ($\beta = 20$).

4.3 Complexity of the method

A way to assess the efficiency of a numerical method is to estimate its complexity. To this end, we will compare LDC to the tensor-product grid refinement. Suppose we have a rectangular domain Ω, as shown in Figure 4.10, covered with a uniform grid. Let N and AN, with A positive, be the number of coarse grid points along the x-axis and the y-axis, respectively. The high activity region, inclined by an angle $\delta = \frac{\pi}{2} - \alpha$ with respect to the x-axis, is covered by $\frac{M \times BM}{m}$ fine grid points, with $B > 1$. The number of points used by LDC will then be

$$N_{LDC} := AN^2 + \frac{BM^2}{m}, \tag{4.28}$$

where m is the ratio between the step size along the y'-axis and the step size along the x'-axis; see Section 3.

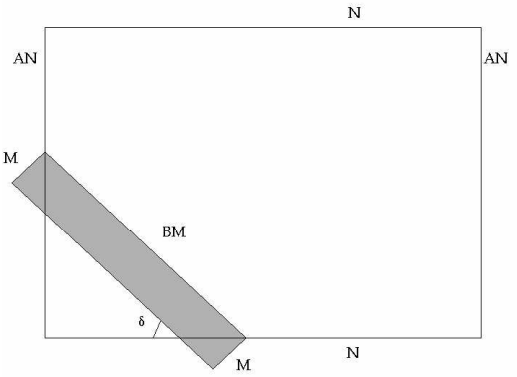


Figure 4.10: Domain Ω with the high activity region.

For the tensor-product grid refinement, we cover the part of Ω containing the high activity area with a finer grid. We need approximately $BM\cos\delta$ grid points on the x-axis and $BM\sin\delta$ grid points on the y-axis, in order to get a grid size comparable with the grid size used in LDC. Thus, we need roughly N_{tensor} grid points, with

$$N_{\text{tensor}} := (BM\cos\delta + N)(BM\sin\delta + AN). \tag{4.29}$$

Note that (4.29) is valid when the ordinate of the upper right corner of the fine grid is smaller than the height of the global domain.

Let us define the gain G as the ratio between N_{tensor} and N_{LDC}. Assuming $N = \gamma M$, G is found to be

$$G(\gamma) = \frac{(B\cos\delta + \gamma)(B\sin\delta + \gamma A)}{A\gamma^2 + \frac{B}{m}}. \tag{4.30}$$

We now assess G by letting γ vary. First suppose that $\gamma \ll 1$: in this case the number of coarse grid points is far less than the number of fine grid points. Such a situation occurs if one reduces the fine grid size h, while keeping constant all geometrical parameters and the width of the local grid. Then the gain G becomes

$$G(\gamma) \doteq \frac{mB\sin(2\delta)}{2}, \tag{4.31}$$

which is apparently independent of A.

Suppose now that $\gamma \gg 1$. This means that the number of fine grid points is much less than the number of coarse grid points. This happens, for instance, when the high activity area takes up just a small corner of the global domain. In such a case we obtain that G does not depend on δ

$$G(\gamma) \doteq \frac{A\gamma^2}{A\gamma^2} = 1. \tag{4.32}$$

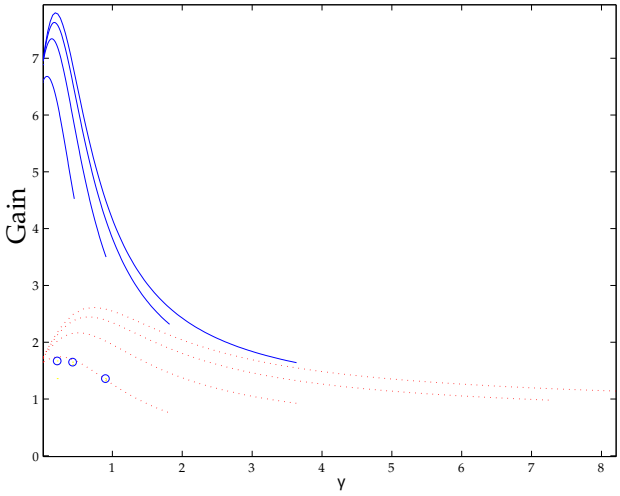


Figure 4.11: Dotted lines have been drawn with $m = 1$, solid lines with $m = 4$. Both sets are parameterised with respect to the dimensions of the domain Ω, that have been assumed equal to $(0,1)\times(0,4)$, $(0,2)\times(0,8)$, $(0,4)\times(0,16)$ and $(0,8)\times(0,32)$, respectively. The bigger Ω, the higher is the gain-curve. On the first dotted line the values of the gain relevant to the second example are reported by circles. They represent the ratio between the number of points that would have been used with the tensor product grid refinement and the number of points reported in Table 4.3.

In the latter situation it is not worthwhile to use LDC because a comparable complexity can be achieved with the tensor product grid refinement.

If $\gamma = 1$, we find

$$G(\gamma) = \frac{(B\cos\delta + 1)(B\sin\delta + A)}{A + \frac{B}{m}}. \tag{4.33}$$

In Figure 4.11 the gain G is plotted versus γ. The geometrical parameters correspond to the second example presented in Section 4: only the dimensions of the global domain are varied to parameterise the curves. The coarse grid size is equal to 10^{-1} and the fine grid size is $h = \frac{l\gamma}{N}$, where l is the width of the local domain. Of course we restrict the value of γ such that $mh \leq H$. In the previous section we have shown that, if a bigger grid size along the y'-axis is chosen, the total number of grid points reduces considerably. Figure 4.11 gives an idea of the increase of the gain G when a factor $m > 1$ is considered. The first set of curves (dotted lines) are computed with $m = 1$ and the second set (solid lines) with $m = 4$. The asymptotic behaviour of the gain when γ increases is clearly shown: once δ has been fixed, the curves tend to 1 when γ is much larger than one, see (4.32).

4.4 Curvilinear refinement domains

Let us focus on the application of LDC with different grid types. In particular let us consider the combination of a global rectangular and a local curvilinear domain. In this

case Ω^l is described in terms of the local (ξ,η)-coordinates. We recall that the one-to-one relation between the physical space (x,y) and the computational space (ξ,η) reads

$$T : \begin{cases} x = x(\xi,\eta), \\ y = y(\xi,\eta). \end{cases} \tag{4.34}$$

We assume that the Jacobian of the transformation, as introduced in (3.32), is different from 0 for all (ξ,η). This implies that T is invertible. In order to express the differential operators in terms of the new curvilinear coordinates, we introduce the function $v(\xi,\eta)$, defined by

$$u(x,y) = u(x(\xi,\eta), y(\xi,\eta)) =: v(\xi,\eta). \tag{4.35}$$

Then, the first partial derivatives and the Laplacian of u become [62]

$$\frac{\partial u}{\partial x} = \frac{1}{J}\left(\frac{\partial y}{\partial \eta}\frac{\partial v}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial v}{\partial \eta}\right), \tag{4.36a}$$

$$\frac{\partial u}{\partial y} = \frac{1}{J}\left(\frac{\partial x}{\partial \xi}\frac{\partial v}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial v}{\partial \xi}\right), \tag{4.36b}$$

$$\begin{aligned} \nabla^2 u = {} & \frac{1}{J^2}\left(g_{\eta\eta}\frac{\partial^2 v}{\partial \xi^2} - 2g_{\xi\eta}\frac{\partial^2 v}{\partial \eta \partial \xi} + g_{\xi\xi}\frac{\partial^2 v}{\partial \eta^2}\right) \\ & - \frac{1}{J^3}\left[\left(g_{\eta\eta}\frac{\partial^2 x}{\partial \xi^2} - 2g_{\xi\eta}\frac{\partial^2 x}{\partial \xi \partial \eta} + g_{\xi\xi}\frac{\partial^2 x}{\partial \eta^2}\right)\left(\frac{\partial y}{\partial \eta}\frac{\partial v}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial v}{\partial \eta}\right)\right. \\ & \left. + \left(g_{\eta\eta}\frac{\partial^2 y}{\partial \xi^2} - 2g_{\xi\eta}\frac{\partial^2 y}{\partial \xi \partial \eta} + g_{\xi\xi}\frac{\partial^2 y}{\partial \eta^2}\right)\left(\frac{\partial x}{\partial \xi}\frac{\partial v}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial v}{\partial \xi}\right)\right], \end{aligned} \tag{4.37}$$

with $g_{\eta\eta}$, $g_{\xi\eta}$ and $g_{\xi\xi}$ given in (3.33). The coefficients $\partial x/\partial \xi$, $\partial y/\partial \xi$, etc. can be computed once transformation (4.34) is known. From (4.36) and (4.37) it appears that the differential operators assume a more complex form in curvilinear coordinates. This effect can be minimized by using an orthogonal grid, i.e. a grid for which the relation

$$\left(\frac{\partial \mathbf{x}}{\partial \xi}, \frac{\partial \mathbf{x}}{\partial \eta}\right) = g_{\xi\eta} = 0, \tag{4.38}$$

holds. This implies that the mixed second derivatives $\frac{\partial^2 x}{\partial \xi \partial \eta}$, $\frac{\partial^2 y}{\partial \xi \partial \eta}$ and $\frac{\partial^2 v}{\partial \xi \partial \eta}$ in (4.37) vanish. Orthogonal coordinate systems not only reduce the number of the additional terms resulting from translating the BVP from the physical into the computational space, but also prevent the introduction of truncation errors in the difference expression, see [62]. In the following we only consider orthogonal grids. Then, the convection-diffusion equation (4.1a) becomes

$$-\left(g_{\eta\eta}\frac{\partial^2 v}{\partial \xi^2} + g_{\xi\xi}\frac{\partial^2 v}{\partial \eta^2}\right) + \frac{1}{J}\left(\varphi\frac{\partial v}{\partial \xi} + \psi\frac{\partial v}{\partial \eta}\right) = J^2 f(x(\xi,\eta), y(\xi,\eta)), \tag{4.39}$$

with

$$\varphi = \sigma\frac{\partial y}{\partial \eta} - \iota\frac{\partial x}{\partial \eta} + J^2(\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}), \tag{4.40a}$$

$$\psi = -\sigma\frac{\partial y}{\partial \xi} + \iota\frac{\partial x}{\partial \eta} + J^2(\frac{\partial x}{\partial \xi} - \frac{\partial y}{\partial \xi}), \tag{4.40b}$$

$$\sigma = g_{\eta\eta}\frac{\partial^2 x}{\partial \xi^2} + g_{\xi\xi}\frac{\partial^2 x}{\partial \eta^2}, \tag{4.40c}$$

$$\iota = g_{\eta\eta}\frac{\partial^2 y}{\partial \xi^2} + g_{\xi\xi}\frac{\partial^2 y}{\partial \eta^2}. \tag{4.40d}$$

Once the coordinate transformation has been accomplished, the resulting BVP has to be discretised in the local domain. The computational domain is built such that (ξ, η) can be regarded as being a uniform rectangular grid, so that all the discretisation techniques suitable for Cartesian grids can be used. Furthermore, since the grid sizes in (ξ, η) are arbitrary and independent of the actual grid size in the physical space, they can be thought of unitary length. Therefore it is not restrictive to take $\xi_i = i$ and $\eta_j = j$, thus simplifying the discretisation of the BVP. Moreover, while switching from a Cartesian to a curvilinear coordinate representation, the order of accuracy of the discretised problem is kept; see [62]. It must be pointed out that also the metric coefficients need to be discretised in the curvilinear space. As it has been shown in [62], the truncation error is reduced if they are evaluated numerically by the same finite difference scheme used for the unknown function.

We close this section by specifying the interpolation operator, $\mathcal{P}^{h,H}$, and the restriction operator, $\mathcal{R}^{H,h}$. As we have done for slanting grids, $\mathcal{P}^{h,H}$ is the bi-linear interpolation operator given by (4.14). The operator $\mathcal{R}^{H,h}$ has to be reformulated, since it makes use of points that do not lie anymore on the vertices of a rectangle. Consider the situation of Figure 4.12 and Figure 4.13, where A, B, C and D are fine grid points and V is now a coarse grid point. The function value in V is thus determined in the following way. First the quadrangle ABCD is split in the two triangles ABC and ACD, one of which contains the point V. Then the function value there is computed as follows

$$u_V = \frac{1}{S}(u_C S_{ABV} + u_B S_{ACV} + u_A S_{BCV}), \tag{4.41}$$

where S_{ABV}, S_{ACV} and S_{BCV} are the areas of the triangles ABV, ACV and BCV, respectively, and S is the area of the quadrangle ABCD.

4.5 LDC applied to curvilinear grids: numerical results

The performance of the LDC method in combination with a local curvilinear grid is assessed by applying it to model problem (4.1), see [22]. In this case the function $s(\mathbf{x})$ is given by

$$s(x, y) = by + ax^2 - r, \tag{4.42}$$

with $a = \frac{1}{2}$, $b = \frac{1}{3}$ and $r = \frac{1}{2}$. Let the global domain be $\Omega = (0, 1.5) \times (0, 4)$ and $\beta = 20$. The function $u(\mathbf{x})$ has again the same behaviour that we have already described in Section 1: it is approximately equal to 2 in the lower part of the domain, where $s(\mathbf{x}) < 0$,

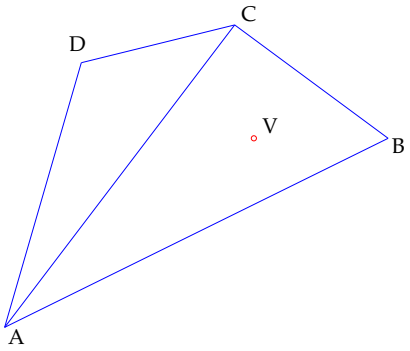


Figure 4.12: Four fine grid points form a quadrangle that contains the coarse grid point.

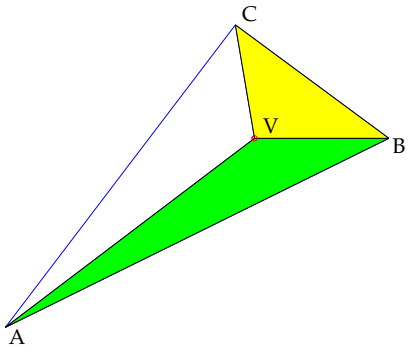


Figure 4.13: The areas of the triangles ABV, CAV and BCV are the coefficients of the interpolation.

and its value decreases steeply over the centre line $s(\mathbf{x}) = 0$, becoming 0 in the upper part of the domain, where $s(\mathbf{x}) > 0$; see Figure 4.14.

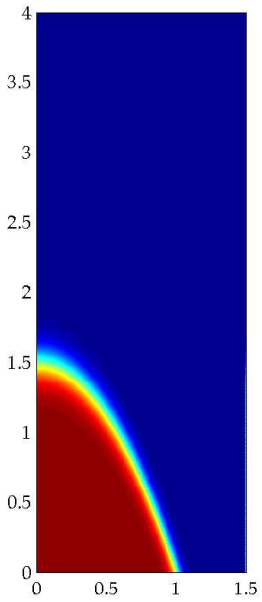
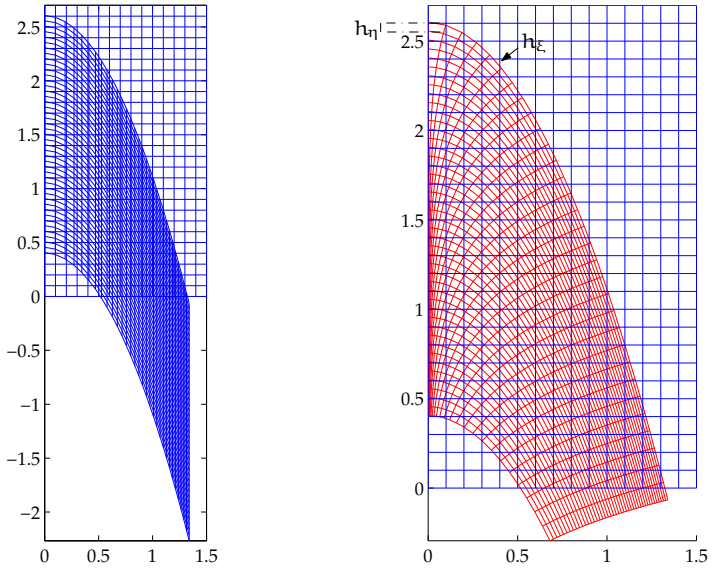


Figure 4.14: Function $u(\mathbf{x})$: view from the top.

In order to discretise the global problem, the domain Ω is covered by a uniform rectangular grid with grid size H. The central difference method is used both on the global and on the local domain. Suppose that we want to compute the solution with an error order of 10^{-3}.



(a) Auxiliary grid. (b) Zoom on the new orthogonal grid.

Figure 4.15: Auxiliary and orthogonal grids.

The upper bound for the discretisation error has been given in (4.25). This implies that if we choose the boundaries of Ω^1 coinciding with the parabolae $by + ax^2 - 0.87 = 0$ and $by + ax^2 - 0.13 = 0$ and set $H = 10^{-1}$, we obtain the solution in $\Omega \backslash \Omega^1$ with the desired accuracy. The local grid is determined by applying the procedure explained in the Section 3.3.1. The auxiliary grid consists of two sets of curves: the η-lines, parallel to the level curves, and the ξ-lines, that are vertical, see Figure 4.15(a). The η-lines have been obtained as follows: by using the coarse grid solution the centre line of the high activity region, corresponding to $u = 1$, is computed by using a least square interpolation polynomial; then this is translated vertically by a certain distance h_η to generate a number of curves sufficient to cover entirely the area to be refined. The orthogonal grid is created starting from the upper curve, where the distance between two consecutive grid points in the physical space is set equal to h_ξ. Both h_η and h_ξ can thus be considered as a measure of the high activity region refinement; see Figure 4.15(b).

The same figure shows that, as already seen with slanting grid, some fine grid points fall outside the global domain. This is unavoidable if the level curves, and then the η-lines, are not perpendicular to its boundaries and we require the to be orthogonal. If such a situation arises, the technique we have used with the Cartesian fine grids is here straightforwardly applicable.

In Table 4.8 we show the infinity norm of the error of the composite grid solution, computed for several values of h_ξ, h_η and H. From these results it can be noticed that the method is still asymptotically second order accurate with respect to h_ξ and h_η, as already shown for local Cartesian grids, see [23]. Furthermore, by comparing the sec-

ond and the fourth columns of Table 4.8, it appears that for constant h_η and h_ξ, errors obtained with $H = 10^{-1}$ are quite close to the errors obtained with $H = 20^{-1}$. This behaviour suggests that, for the H-values considered here, the discretisation errors on the fine grid are dominant. The only value that does not comply with this remark is the last one in the third column. This is due to the fact that the discretisation error on the coarse grid is now of the same order of magnitude as the one on the fine grid. Further refinement of the high activity region will thus not produce any improvement on the composite grid approximation error.

h_η	$H = 10^{-1}$	ratio	$H = 20^{-1}$	ratio
$1 \cdot 10^{-1}$	$4.34 \cdot 10^{-2}$		$3.28 \cdot 10^{-2}$	
$5 \cdot 10^{-2}$	$8.01 \cdot 10^{-3}$	5.4	$7.70 \cdot 10^{-3}$	4.3
$2.5 \cdot 10^{-2}$	$2.20 \cdot 10^{-3}$	3.6	$1.90 \cdot 10^{-3}$	4.1
$1.25 \cdot 10^{-2}$	$1.30 \cdot 10^{-3}$	1.7	$4.76 \cdot 10^{-4}$	4.0

Table 4.8: Composite grid approximation error $\|u^* - u_{H,h}\|_\infty$ ($h_\xi = h_\eta$).

Numerical experiments show that the behaviour of the LDC method in combination with Cartesian or with curvilinear local grids is essentially the same. Furthermore, in the latter case, one can always benefit from the fact that choosing the η-lines coincident with the level curves implies that, along those, gradients are equal to zero or very small. As a consequence, the grid size in this direction can be increased without spoiling the accuracy of the solution. Table 4.9 presents again the composite grid error relevant to our last model problem. Although h_ξ is doubled compared to the values in Table 4.8, the solution is approximated within almost the same accuracy. The last column of Table 4.9 shows the number of η-lines lines used to cover the high activity region.

h_η	$H = 10^{-1}$	ratio	$H = 20^{-1}$	ratio	η — lines
$5 \cdot 10^{-2}$	$1.06 \cdot 10^{-2}$		$1.10 \cdot 10^{-2}$		45
$2.5 \cdot 10^{-2}$	$2.70 \cdot 10^{-3}$	3.9	$1.70 \cdot 10^{-3}$	6.5	89
$1.25 \cdot 10^{-2}$	$1.30 \cdot 10^{-3}$	2.1	$4.30 \cdot 10^{-4}$	4.0	175

Table 4.9: Composite grid approximation error $\|u^* - u_{H,h}\|_\infty$ ($h_\xi = 2h_\eta$).

Another way to reduce the number of the fine grid points consists of thining out the η-lines while moving away from the high activity region. This is done on the basis of the second derivative $\partial^2 u/\partial y^2$, computed along the line $x = 0$. We start from the point $(x_0 = 0, y_0 = 1.5)$, placed on the central line of the high activity area. Suppose now

we want to refine the region above this line starting with a certain grid size h_{η_0} that increases as we move away from $\mathbf{x}_0$. To do so, we evaluate $(\partial^2 u/\partial y^2)(0, y_{i-1} + h_{\eta_{i-1}})$ and update h_η in the following way

- $h_{\eta_i} = h_{\eta_{i-1}} \left(\frac{\partial^2 u}{\partial y^2}\Big|_{\mathbf{x}_{i-1}} / \frac{\partial^2 u}{\partial y^2}\Big|_{\mathbf{x}_i} \right)$;
- if $h_{\eta_i} < h_{\eta_0}$, set $h_{\eta_i} = h_{\eta_0}$;
- if $h_{\eta i} > \nu h_{\eta_0}$ for some $\nu > 1$, set $h_{\eta_i} = h_{\eta_{i-1}}$,

where $\mathbf{x}_i = (0, y_i)$. To understand the meaning of the first if-statement, we must realise that $\partial^2 u/\partial y^2$ is an odd function that is equal to zero just across the line from which we start to refine and that reaches its maximum absolute values above and below this line. Thus, without this statement, the first steps of the algorithm would lead to $h_{\eta_i} < h_{\eta_0}$. Conversely, the second if-statement prevents a small second derivative from resulting in a very big h_η. We have used $\nu = 3$. Since we know the analytical form of u on the boundary $x = 0$, we can easily compute its second derivatives. In other cases it can be evaluated numerically. The same procedure is used to refine the region below the considered η-line. Results are reported in Table 4.10, where h_{η_0} is the grid size across the central line of the high activity region. The approximation errors in Table 4.10 have to be compared with those shown in the second column of Table 4.9 ($H = 10^{-1}$). It appears that, when h_η varies according to the procedure previously explained, the number of η-lines is considerably reduced while the accuracy of the approximated solution is almost the same. The last row of Table 4.10 shows a smaller error even with respect to the corresponding value in Table 4.9. This is due to the fact that, by using a variable h_η, we cannot impose a priori the fine grid to be exactly as wide as the fine grid built with a constant h_η. Thus, in the former case, the fine grid turns out to be slightly wider and this leads to a more accurate solution. The achievable contour-lines reduction depends, in general, on the shape of the unknown function. Here we have managed to save, on average, 30% of them. Let us now compare Table 4.8 and Table 4.9. We see that, for equal values of h_η, the reduction factor achievable by using a bigger h_ξ is equal to 2. On the other hand, by comparing the number of η-lines in Table 4.9 and Table 4.10, we notice that the possibility to use a variable h_η leads, on average, to a reduction factor equal to 1.5. Then, the total reduction factor that can be obtained by using the level curves as set of coordinate lines and without any loss in the accuracy of the solution is roughly equal to 3.

h_{η_0}	h_ξ	$\|\mathbf{u}^* - \mathbf{u}_{H,h}\|_\infty$	**ratio**	η — lines
$5 \cdot 10^{-2}$	$1 \cdot 10^{-1}$	$9.80 \cdot 10^{-3}$		33
$2.5 \cdot 10^{-2}$	$5. \cdot 10^{-2}$	$2.70 \cdot 10^{-3}$	3.63	58
$1.25 \cdot 10^{-2}$	$2.5 \cdot 10^{-2}$	$7.42 \cdot 10^{-4}$	3.64	111

Table 4.10: Composite grid approximation error with $H = 10^{-1}$ and h_η variable.

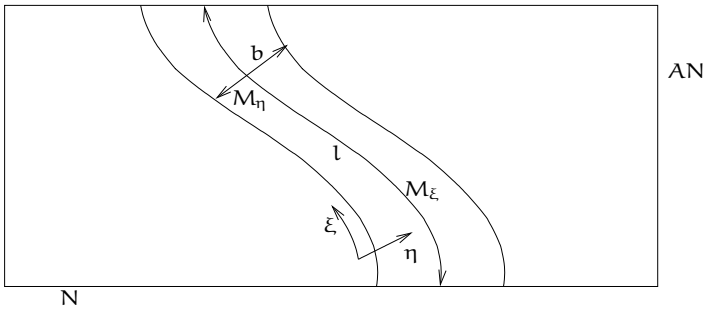


Figure 4.16: Characteristics of the global and the local domain.

The results in Table 4.8, 4.9 and 4.10 are obtained with one complete LDC iteration, after which convergence is achieved.

4.6 Complexity analysis of LDC and curvilinear grids

While the analysis performed in Section 4.3 aims to compare the memory usage of LDC versus tensor product grid, here we try to assess the global complexity of the algorithm that combines the LDC method with curvilinear orthogonal grids.

The setting is defined as follows (see Figure 4.16). The number of coarse grid points along the x- and the y-axis are N and AN, respectively. We introduce a characteristic length l of the fine grid: it can be, for instance, the length of the contour-line η_0 where the activity reaches its maximum value. Moreover, we name b the average width of the fine grid. The grid sizes along l and b are h_ξ and h_η, respectively. Let N_{LDC} be the number of points used by the LDC algorithm. If M_ξ and M_η are the number of the fine grid points along the ξ- and the η-axis, then

$$N_{LDC} = AN^2 + M_\xi M_\eta.$$

Suppose now that in the worst case, i.e. when the high activity region runs from the bottom to the top of the global domain, l is comparable with the height of the global domain Ω. Then, since the η-lines are level curves, the solution is slowly varying along them and we can set $h_\xi = H$. From that follows that $M_\xi = BN = \mathcal{O}(N)$ and the complexity of the LDC method becomes

$$N_{LDC} = AN^2 + BN \cdot M_\eta.$$

The order of magnitude of M_η is usually comparable with N. This implies that

$$N_{LDC} = \mathcal{O}(N^2), \tag{4.43}$$

i.e. a highly detailed solution can be obtained by solving two systems having the same complexity of the coarse grid problem. However, we can try to reduce some more the

complexity of the fine grid BVP by adapting the number of the η-lines to the behaviour of the solution. In fact, we can vary the step size h_η according to

$$h_{\eta_i} w_i = h_{\eta_{i-1}} w_{i-1} = \text{const}, \quad \text{or} \quad h_{\eta_i} = r_{i-1} h_{\eta_{i-1}}. \tag{4.44}$$

Here w is a given weight function that reflects, in some way, the shape of the solution: it can be, for instance, the source term for a flame simulation. Furthermore $r_{i-1} = w_{i-1}/w_i$. If we choose h_η to vary according to (4.44) then, since r is variable, it becomes very difficult to make predictions about the number of contour-lines to be used. So, we take r constant and equal to the geometric average of the variable r_i. This way, we expect not to make a big error in the evaluation of N_η. We can split the area to be refined in two parts: one on the left (η decreasing) and the other on the right (η increasing) of η_0. Let us consider what happen to the area on the left of η_0. We introduce the following quantities: b_l, its average width, N_{η_l}, the number of contour-lines with which it is covered, r_l, the value that r assumes there. We then get

$$b_l = h_{\eta_0}\left(1 + r_l + \ldots + r_l^{N_{\eta_l}-1}\right) = h_{\eta_0}\frac{1 - r_l^{N_{\eta_l}}}{1 - r_l}. \tag{4.45}$$

If $M_{\eta_l} + 1$ is the number of η-lines that has to be used if b_l is uniformly refined, (4.45) can be written as

$$M_{\eta_l}(1 - r_l) = 1 - r_l^{N_{\eta_l}}, \tag{4.46}$$

from which follows

$$N_{\eta_l} = \frac{\ln\left(1 + M_{\eta_l}(r_l - 1)\right)}{\ln r_l}. \tag{4.47}$$

A relation similar to (4.47) holds for the high activity area placed on the right of η_0. We know that, by definition, $r_l > 1$. Furthermore, since the grid sizes are not allowed to change too rapidly where gradients are high, see [62], its value, as well as the values of a single r_i, must be limited. Suppose then to set $1 < r_l < c_l$, with c_l small and define $a_l = r_l - 1$. By expanding the denominator of (4.47) by a Taylor series, we get

$$N_{\eta_l} = \frac{\ln\left(1 + M_{\eta_l} a_l\right)}{a_l - \frac{a_l^2}{2} + \frac{a_l^3}{3} + \ldots} \approx \frac{\ln\left(1 + M_{\eta_l} a_l\right)}{a_l}. \tag{4.48}$$

We see that the value of N_{η_l} in (4.48) tends to M_{η_l} when a_l goes to zero, as expected. Eventually, the complexity of the LDC method can be expressed by the sum of three terms

$$N_{LDC} \approx AN^2 + BN \cdot \frac{\ln\left(1 + M_{\eta_l} a_l\right)}{a_l} + BN \cdot \frac{\ln\left(1 + M_{\eta_r} a_r\right)}{a_r}, \tag{4.49}$$

where M_{η_r} and a_r are the analogues of M_{η_l} and a_l on the right of η_0. The leading term of the sum in (4.49) still remains the coarse grid contribution. Nevertheless, the second and third term can be considerably smaller than M_{η_l} and M_{η_r}, depending on the value of a_l and a_r, respectively. In Figure 4.17 the behaviour of $N_\eta = \ln(1 + M_\eta a)/a$ versus M_η and for several values of a is plotted. We see that the choice of a variable h_η leads to a considerable reduction of the computational efforts required to solve the local BVP.

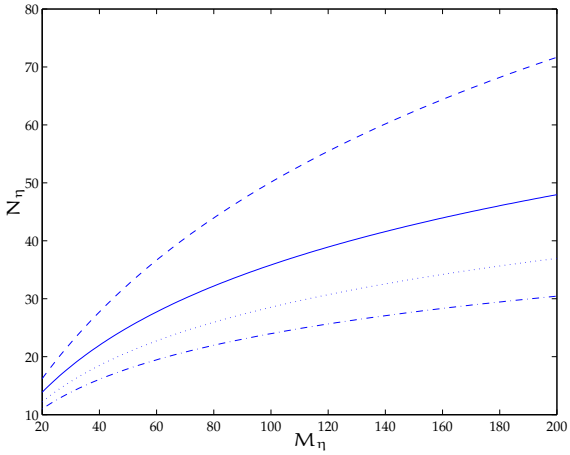


Figure 4.17: The function $N_\eta = \ln(1 + M_\eta a)/a$ is plotted versus M_η. It is parameterised with respect to a, assumed equal to 0.025 (dashed line), 0.05 (solid line), 0.075 (dotted line) and 0.1 (dashdot line).

CHAPTER 5

Solving nonlinear systems via embedding methods

In this chapter we discuss some methods to efficiently solve nonlinear algebraic systems. In Chapter 2 we have already pointed out that the combustion equations are highly nonlinear. After discretisation then, they yield algebraic systems that pose big problems when computing a solution. These difficulties, however, are not only characteristic of the equations that describe combustion, but concern many other phenomena, therefore the methods presented in this chapter are thus generally applicable. The simplest and most widely used iterative method, the *Newton method*, often fails when the initial guess happens not to be close to the sought solution. The goal of this chapter is then to investigate alternative algorithms able to improve its robustness. We do that by embedding the considered nonlinear system into a time dependent problem. This procedure is outlined in Section 5.1. The integration of the resulting system is introduced in Section 5.2, while the implementation of the method is presented in Section 5.3. Finally, Section 5.4 is devoted to numerical experiments.

5.1 The Davidenko equation

We focus on the solution of the following nonlinear algebraic system

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \tag{5.1}$$

where $\mathbf{F} : \mathrm{R}^n \to \mathrm{R}^n$, with a zero at $\mathbf{x} = \mathbf{x}^*$, say. The simplest approach is given by Newton's method. Let us linearise (5.1) about a given point $\mathbf{x}^i$, see [5], i.e.

$$\mathbf{F}(\mathbf{x}) \doteq \mathbf{F}(\mathbf{x}^i) + \mathbf{J}(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i), \tag{5.2}$$

where $\mathbf{J}(\mathbf{x}^i)$ is the Jacobi matrix of $\mathbf{F}(\mathbf{x})$ evaluated at the point $\mathbf{x}^i$. If $\mathbf{J}(\mathbf{x}^i)$ is non-singular, (5.2) gives rise to a series of successive approximations of $\mathbf{x}^*$, defined as

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i), \qquad i = 0, 1, 2, \ldots \tag{5.3}$$

given an initial guess $\mathbf{x}^0$. The vector $-\mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i)$ defined by (5.3) is known as the *Newton update vector*. In order to move from the $\mathbf{x}^i$-iterate to the successive $\mathbf{x}^{i+1}$, a unit step size is taken in the direction $-\mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i)$. Newton's method is known to give local second order convergence, i.e.

$$\| \mathbf{x}^* - \mathbf{x}^{i+1} \| = \mathcal{O}\big(\| \mathbf{x}^* - \mathbf{x}^i \|^2 \big), \qquad i \to \infty. \tag{5.4}$$

In spite of its simplicity and its good convergence properties, direct application of Newton's method often does not work with highly nonlinear problems. In fact, its convergence region, i.e. that part of the domain of $\mathbf{F}$ to which $\mathbf{x}^0$ has to belong to achieve global convergence, can be very small. Therefore, the main focus is on how to obtain $\mathbf{x}^0$ such that global convergence is possible. The robustness of Newton's method can be improved by introducing a damping factor $0 < \lambda_i \leq 1$, that yields the iteration

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda_i \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i), \qquad i = 0, 1, 2, \ldots. \tag{5.5}$$

Such a procedure relies on the observation that, given an *objective function* $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, defined as $g(\mathbf{x}) := \frac{1}{2} \| \mathbf{F}(\mathbf{x}) \|_2$, the *Newton direction* is a *descent direction* with respect to $g(\mathbf{x})$, i.e. $\mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i) \cdot \boldsymbol{\nabla} g < 0$, see [5]. For nonlinear systems obtained from discretisation of combustion problems, a good strategy to determine λ_i is not straightforward. A different approach is then required.

Hence, we introduce a solution strategy that is based on *embedding* the original system into an initial value problem. We can do that in two different ways. In fact we can consider both

$$\frac{d\mathbf{x}}{d\tau} = \mathbf{F}(\mathbf{x}), \tag{5.6a}$$

$$\mathbf{x}(0) = \mathbf{x}_0, \tag{5.6b}$$

and

$$\frac{d\mathbf{x}}{d\tau} = -\mathbf{J}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x}), \tag{5.7a}$$

$$\mathbf{x}(0) = \mathbf{x}_0, \tag{5.7b}$$

with τ an artificial time.

Problem (5.6) is a reminiscence of a parabolic problem where $\mathbf{F}(\mathbf{x})$ is an elliptic (discrete) operator. Although $\mathbf{x}^*$ is a rest point of (5.6), it may not be a stable one. The other option, i.e. the *Davidenko equation*, has a better potential, therefore we concentrate on the solution of the IVP (5.7). It is worth to remark that the explicit Euler scheme applied to (5.7) with a step size $\lambda_i = \tau_{i+1} - \tau_i$ gives exactly *damped Newton*. This implies that improving robustness for the *damped Newton method* applied to the original nonlinear system (5.1) translates into improving stability of the Euler forward scheme applied to the system (5.7). In the following we will indicate with $\mathbf{x}^i$ a numerical approximation of $\mathbf{x}(\tau_i)$.

Before we start looking for such a technique, we have to state the conditions under which the ODE system (5.7) is asymptotically stable. In fact, only in this case the search

for a zero of $\mathbf{F}(\mathbf{x})$ is meaningful. The following assumptions and lemma have been presented in [35]. Here we introduce them in an adapted form suitable to be applied to (5.7).

Assumption 5.1 *Suppose there is a ball* $B(\mathbf{x}^*, R)$*, of centre* $\mathbf{x}^*$ *and radius* R*, such that*

(i) *The functions* $\mathbf{F}(\mathbf{x})$*,* $J(\mathbf{x})$ *and* $J^{-1}(\mathbf{x})$ *are bounded on* $B(\mathbf{x}^*, R)$*. Then*

$$\| \mathbf{F}(\mathbf{x}) \| \leq C_F, \quad \| \mathbf{J}(\mathbf{x}) \| \leq C_J, \quad \| \mathbf{J}^{-1}(\mathbf{x}) \| \leq C_M, \quad \big(\textit{for all } \mathbf{x} \in B(\mathbf{x}^*, R)\big). \tag{5.8}$$

(ii) *The functions* $\mathbf{J}(\mathbf{x})$ *and* $\mathbf{J}^{-1}(\mathbf{x})$ *are Lipschitz continuous on* $B(\mathbf{x}^*, R)$ *with Lipschitz constants* L_J *and* L_M*, respectively, i.e.*

$$\| \mathbf{J}(\mathbf{x}) - \mathbf{J}(\mathbf{y}) \| \leq L_J \| \mathbf{x} - \mathbf{y} \| \quad \textit{and} \quad \| \mathbf{J}^{-1}(\mathbf{x}) - \mathbf{J}^{-1}(\mathbf{y}) \| \leq L_M \| \mathbf{x} - \mathbf{y} \| . \tag{5.9}$$

Based on Assumptions 5.1, it has been shown, see [35], that once the initial guess $\mathbf{x}_0$ is chosen such that it belongs to a certain ball $B(\mathbf{x}^*, r)$, the function $\mathbf{x}(\tau)$, solution of (5.7), remains in $B(\mathbf{x}^*, r)$. We can formulate this more precisely. Let us introduce a constant $\hat{C} = \hat{C}(R)$ such that

$$\hat{C} := \max_{\mathbf{x} \in B(\mathbf{x}^*, R)} \frac{< \mathbf{x} - \mathbf{x}^*, -\mathbf{J}^{-1}(\mathbf{x})(\mathbf{F}(\mathbf{x}) - \mathbf{J}(\mathbf{x})(\mathbf{x} - \mathbf{x}^*)) >}{\| \mathbf{x} - \mathbf{x}^* \|^3}, \tag{5.10}$$

where $< \cdot, \cdot >$ denotes the Euclidean inner product. Then the following lemma holds

Lemma 5.1 *If* $r < \min(\hat{C}^{-1}, R)$*, then the solution* $\mathbf{x}(\tau)$ *of (5.7) satisfies the following*

$$\forall \mathbf{x}_0 = \mathbf{x}(0) \in B(\mathbf{x}^*, r) : \mathbf{x}(\tau) \in B(\mathbf{x}^*, r). \tag{5.11}$$

Moreover

$$\| \mathbf{x}(\tau) - \mathbf{x}^* \| \leq \exp((-1 + \hat{C}r)\tau) \| \mathbf{x}_0 - \mathbf{x}^* \| . \tag{5.12}$$

5.2 The mixed Euler method

In this section we look for an integration method that is able to enhance stability of the forward Euler formula applied to (5.7). We must stress the fact that the final goal is not to obtain an accurate temporal evolution of $\mathbf{x}(\tau)$, but to reach a steady state solution of the ODE system (5.7). This implies that we would prefer to have large time steps, especially when $\mathbf{x}^i$ approaches $\mathbf{x}^*$. The only limitations that we have to impose to the time stepping are those necessary not to jeopardize the stability of the process.

The simplest candidate that satisfies these requirements is the backward Euler formula. If applied to (5.7a), it reads

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda_i \mathbf{J}^{-1}(\mathbf{x}^{i+1}) \mathbf{F}(\mathbf{x}^{i+1}). \tag{5.13}$$

Looking at (5.13), we realise that the Euler backward scheme has a major drawback that makes its application unfeasible: it is fully implicit, i.e. in the left-hand side there are two functions, $\mathbf{F}(\mathbf{x})$ and $\mathbf{J}^{-1}(\mathbf{x})$ that have to be evaluated at the new time level. In particular the term $\mathbf{J}^{-1}(\mathbf{x})$ represents the most expensive part. In fact, this would require several iterations and a relatively big computational effort. The situation can be improved if we consider a *mixed Euler* formula, i.e. a mix between an explicit and an implicit scheme, as follows

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda_i \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^{i+1}). \tag{5.14}$$

The iteration method (5.14) is computationally less expensive than (5.13). Furthermore, it can be shown, see [35], that it is consistent of order one and that its stability properties are similar to those of the implicit Euler method.

System (5.14) is again nonlinear in $\mathbf{z} := \mathbf{x}^{i+1}$. It can be symbolically written as

$$\mathbf{G}(\mathbf{z}) := \mathbf{z} + \lambda_i \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{z}) - \mathbf{x}^i = \mathbf{0}, \tag{5.15}$$

and solved iteratively by applying again Newton's method. This results in

$$\mathbf{z}^{l+1} = \mathbf{z}^l - \left(\mathbf{I} + \lambda_i \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{J}(\mathbf{z}^l)\right)^{-1}\mathbf{G}(\mathbf{z}^l). \tag{5.16}$$

We notice that the solution of (5.16) requires the evaluation of the Jacobi matrix every time a new approximation $\mathbf{z}^{l+1}$ is computed. Since this would heavily affect the computational efficiency of the method, we simply replace $\mathbf{J}(\mathbf{z}^l)$ by $\mathbf{J}(\mathbf{x}^j)$, i.e. we freeze the Jacobi matrix in the inner iteration. This way, (5.16) reduces to

$$\mathbf{z}^{l+1} = \frac{\lambda_i}{1+\lambda_i}\left(\mathbf{z}^l - \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{z}^l)\right) + \frac{1}{1+\lambda_i}\mathbf{x}^i. \tag{5.17}$$

Thus, our method consists of two iteration levels: the external one given by (5.14) whose iterations determine the temporal evolution of $\mathbf{x}$, and the internal one, i.e. (5.17), due to the nonlinearity contained in (5.14).

Some interesting findings relevant to the convergence of the *mixed Euler* method are shown in [35]. We recall here a theorem that summarizes those results.

Theorem 5.1 *Let* $r < \min(\hat{C}^{-1}, R)$, *let* $\mathbf{x}^i \in B(\mathbf{x}^*; r)$ *and suppose that* λ_i *is sufficiently small to guarantee that also* $\mathbf{x}^{i+1} \in B(\mathbf{x}^*; r)$. *If* $1 - \hat{C}r - C_J L_M \parallel \mathbf{x}^i - \mathbf{x}^{i+1} \parallel > 0$ *then*

$$\parallel \mathbf{x}^{i+1} - \mathbf{x}^* \parallel \leq \frac{\parallel \mathbf{x}^i - \mathbf{x}^* \parallel}{b_i}, \tag{5.18}$$

with

$$b_i := 1 + \lambda_i\left(1 - C_J L_M \parallel \mathbf{x}^i - \mathbf{x}^{i+1} \parallel - \hat{C} \parallel \mathbf{x}^i - \mathbf{x}^* \parallel\right) > 1. \tag{5.19}$$

Moreover, the vector $\mathbf{x}^{i+1}$ *is in the sphere with centre*

$$\left(1 - \frac{1}{2b_i}\right)\mathbf{x}^* + \frac{1}{2b_i}, \tag{5.20}$$

and radius

$$\frac{\parallel \mathbf{x}^i - \mathbf{x}^* \parallel}{2b_i}. \tag{5.21}$$

From (5.18) and (5.19) we see that, as soon as $\| \mathbf{x}^i - \mathbf{x}^{i+1} \| < (1 - \hat{C}r)/2C_J L_M$, the constant b_i becomes bigger than 1, whichever is λ_i. Then the choice of a growing λ_i with increasing i implies superlinear convergence.

5.3 Implementation

A crucial issue for the success of the method is the control of the step size λ_i. A reasonable way to do this is to choose the time steps such that the discretisation error does not exceed a certain fixed tolerance. Obviously, we would also like to build an algorithm such that λ_i eventually becomes infinitely large. There are several ways to achieve this objective. In the following subsection, we discuss the method presented in [35], which is based on the possibility of giving a discrete estimate of the derivative of the function $\mathbf{x} = \mathbf{x}(\tau)$. Furthermore, we introduce two improvements in this technique: the first one is based on the use of a *second order backward difference* formula, the second one on *extrapolation*.

5.3.1 Step size control via the second derivative approximation

Consider the following generic integration scheme for problem (5.7), see [5]

$$\mathbf{x}^{i+k} = \beta_{k-1}\mathbf{x}^{i+k-1} + \cdots + \beta_0 \mathbf{x}^i + \lambda \Phi(\mathbf{x}^{i+k}, \cdots, \mathbf{x}^i; \lambda), \qquad i = 0, 1, 2, \cdots, \tag{5.22}$$

which holds for λ constant. Here, β_0, β_1, $\cdots$, β_{k-1} are known constants and Φ is an increment function depending on $\mathbf{J}(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x})$. The *local truncation* or *discretisation error* δ_i at τ_i is defined as

$$\delta_i := \lambda^{-1}\Big[\mathbf{x}(t_{i+k}) - \sum_{j=0}^{k-1} \mathbf{x}(\tau_{i+j})\beta_j - \Phi\big(\mathbf{x}(\tau_{i+k}), \cdots, \mathbf{x}(\tau_i); \lambda\big)\Big], \tag{5.23}$$

and the scheme is said to be *consistent of order p* if $\delta_i = \mathcal{O}(\lambda^p)$. For such schemes, δ_i can be expressed as

$$\delta_i(\lambda) = C\mathbf{x}^{(p+1)}(\tau_i)\lambda^p + \mathcal{O}(\lambda^{p+1}), \tag{5.24}$$

where C is a constant and $\mathbf{x}^{(p+1)}(\tau_i)$ the $(p+1)$-order derivative of $\mathbf{x}$ with respect to τ evaluated in τ_i.

Let us now come back to the mixed Euler method: in [35] it has been shown to be consistent of order one. We would like to determine the step size such that the integral of the discretisation error over a time interval remains approximately bounded. More precisely, we require

$$\big|\lambda\delta_i(\lambda)\big| \approx \mathrm{TOL}, \tag{5.25}$$

with TOL our desired tolerance. Applying (5.24) to the *mixed Euler method*, we can give an estimate of the local truncation error trying to evaluate numerically the second derivative of the function $\mathbf{x}(\tau)$, where $C = \frac{1}{2}$. Then we obtain

$$\mathrm{EST} := \lambda_i^2 \left| \frac{\| \mathbf{x}^i - \mathbf{x}^{i-1} \|}{\lambda_{i-1}} - \frac{\| \mathbf{x}^{i-1} - \mathbf{x}^{i-2} \|}{\lambda_{i-2}} \right| \cdot \frac{1}{\lambda_{i-1} + \lambda_{i-2}} \approx \big|\lambda\delta_i(\lambda)\big|. \tag{5.26}$$

In the following algorithm, EST is required to satisfy

$$\mathrm{EST} \leq \mathrm{ATOL} + \mathrm{RTOL} \cdot \parallel \mathbf{x}^i \parallel, \tag{5.27}$$

where ATOL and RTOL are an absolute and a relative tolerance, respectively. When the function to be integrated is smooth, the application of (5.25) implies that bigger step sizes are allowed, while, for a more irregular behaviour, the step size is reduced to enhance robustness. The values ATOL and RTOL determine the accuracy with which the temporal evolution of the system is computed and are up to the user. Since we are only interested in the steady state solution and not in the transient behaviour, we like to keep these tolerances as large as possible, thus reducing the computational work to achieve convergence. On the other hand, when very stiff nonlinear problems have to be solved, it is possible to decrease the value of ATOL and RTOL such that a more cautious, and thus slower, path towards the steady state can be followed. Based on these considerations the following algorithm, similar to the one presented in [35], has been developed

Mixed Euler Algorithm (MEA)

- Let $\mathbf{x}^0$ be given;
- For $i = 0, 1$
  - Set $\lambda_i = \frac{1}{1+\|\mathbf{F}(\mathbf{x}^i)\|}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$ using (5.17);
    * If the Newton process (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
- For $i = 2, 3, \cdots$
  - Set $\lambda_i = \lambda_{i-1}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$
    * If the Newton process (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
  - Compute EST with (5.26);
  - Evaluate the test function

$$\mathrm{TEST} := \frac{\mathrm{EST}}{\mathrm{ATOL} + \mathrm{RTOL} \cdot \parallel \mathbf{x}^i \parallel}; \tag{5.28}$$

  - If $\frac{1}{\rho} \leq \mathrm{TEST} \leq \rho$ the solution $\mathbf{x}^{i+1}$ is accepted;
  - If $\mathrm{TEST} > \rho$ the discretisation error is too big, the solution $\mathbf{x}^{i+1}$ is rejected and a new solution is computed after updating λ_i according to

$$\lambda_i =: \alpha \frac{\lambda_i}{\sqrt{\mathrm{TEST}}}; \tag{5.29}$$

- If $\text{TEST} < \frac{1}{\rho}$ the temporal evolution is being followed too accurately: the new solution $\mathbf{x}^{i+1}$ is accepted and the step size is increased according to (5.29);
- If λ_i has not been changed for the last n iterations, set $\lambda_i := 2\lambda_i$;
- If $\text{TEST} < \rho$ check the stopping criterion on $\| \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^{i+1}) \|$.

The values of n, m_1, m_2 and ρ can be arbitrarily chosen. In the numerical experiments of Section 5.4 we use $m1 = n = 3$, $m2 = 10$ and $\rho = 4$. Expression (5.29) relies on the fact that EST, according to (5.26), is proportional to λ_i^2. Moreover, α is a *safety coefficient* used to deal with severe nonlinearities. Obviously, it has to satisfy the inequality $0 < \alpha \leq 1$. This coefficient is usually set equal to 0.9.

5.3.2 Step size control via a higher order method

In spite of its simplicity, the estimate of the discretisation error via the second derivative it is not always reliable. In fact the numerical evaluation of $d^2\mathbf{x}/d\tau^2$ can be misleading, especially if there are regions where the solution pattern evolves irregularly. The requirement to obtain a robust method induces to look for other kind of solutions. A suitable alternative is to employ embedding in combination with a method presented in [5]. Let us consider two different schemes: one of order p and the other of order q, with $q > p$, applied to system (5.7). We denote by $\mathbf{x}^i$ and $\hat{\mathbf{x}}^i$ its numerical solution computed at the time τ_i by the first and the second scheme, respectively. Moreover, let $\mathbf{x}(\tau)$ be the exact solution of (5.7), subject to the condition $\mathbf{x}(\tau_i) = \mathbf{x}^i$. We want to determine λ_i in order to compute $\mathbf{x}^{i+1}$. For the scheme of order p, we can write

$$\lambda\delta(\mathbf{x}(\tau_{i+1}), \lambda) = \mathbf{x}(\tau_{i+1}) - \mathbf{x}^{i+1}, \quad \text{with} \quad \delta(\mathbf{x}(\tau_{i+1}), \lambda) = \mathcal{O}(\lambda^p), \tag{5.30}$$

and, for the scheme of order q

$$\lambda\hat{\delta}(\mathbf{x}(\tau_{i+1}), \lambda) = \mathbf{x}(\tau_{i+1}) - \hat{\mathbf{x}}^{i+1}, \quad \text{with} \quad \hat{\delta}(\mathbf{x}(\tau_{i+1}), \lambda) = \mathcal{O}(\lambda^q), \tag{5.31}$$

that hold for λ constant. Subtracting (5.31) from (5.30), we get

$$\begin{aligned} \hat{\mathbf{x}}^{i+1} - \mathbf{x}^{i+1} &= \lambda\big(\delta(\mathbf{x}(\tau_{i+1}), \lambda) - \hat{\delta}(\mathbf{x}(\tau_{i+1}), \lambda)\big) \\ &= \lambda\delta(\mathbf{x}(\tau_{i+1}), \lambda) + \mathcal{O}(\lambda^{q+1}). \end{aligned} \tag{5.32}$$

From (5.32) it follows that

$$\text{EST} := \| \hat{\mathbf{x}}^{i+1} - \mathbf{x}^{i+1} \| \doteq \| \lambda\delta(\mathbf{x}(\tau_{i+1}), \lambda) \| . \tag{5.33}$$

Let us then apply (5.33) in a variable step size context. Based on this last expression, the relation between two successive time steps able to equidistribute the local error over the time interval becomes

$$\left(\frac{\lambda_i^{new}}{\lambda_i^{old}}\right)^{p+1} = \frac{\text{TOL}}{\text{EST}}. \tag{5.34}$$

The implementation of (5.33) requires then two schemes with different consistency order. Obviously, our choice is determined by the requirement to save computational

work, thus we set $p = 1$ and $q = 2$. The mixed Euler method is the most natural candidate when considering the order one scheme; it has now to be coupled with an order two scheme. There are several possibilities for this. We choose the second order backward difference formula

$$\hat{\mathbf{x}}^{i+1} = \frac{4}{3}\hat{\mathbf{x}}^{i} - \frac{2}{3}\lambda_i \mathbf{J}^{-1}(\hat{\mathbf{x}}^{i+1})F(\hat{\mathbf{x}}^{i+1}) - \frac{1}{3}\hat{\mathbf{x}}^{i-1}, \tag{5.35}$$

that has stability properties similar to the first order Euler backward scheme. Again, (5.35) is a fully implicit scheme, thus useless to our purposes if not adequately worked out. We could be tempted to implement it in a mixed way, as already done for the first order backward formula, and write

$$\hat{\mathbf{x}}^{i+1} = \frac{4}{3}\hat{\mathbf{x}}^{i} - \frac{2}{3}\lambda_i \mathbf{J}^{-1}(\hat{\mathbf{x}}^{i})\mathbf{F}(\hat{\mathbf{x}}^{i+1}) - \frac{1}{3}\hat{\mathbf{x}}^{i-1}. \tag{5.36}$$

But we notice that, in this case, (5.35) transforms irreparably into a first order formula. In fact, while in [35] it is shown that the discretisation error corresponding to the mixed Euler method is still bounded by the product of a certain constant and the time step λ_i, now it is not possible to arrange (5.36) such that the upper bound of the local error is proportional to λ_i^2.

To find out the accuracy of (5.36) we prefer to work with the equivalent scalar equation

$$\hat{x}^{i+1} = \frac{4}{3}\hat{x}^{i} - \frac{2}{3}\lambda_i f'^{-1}(\hat{x}^{i})f(\hat{x}^{i+1}) - \frac{1}{3}\hat{x}^{i-1}. \tag{5.37}$$

Results can be then easily generalised to the multidimensional case. Let us rearrange (5.37) as follows

$$\begin{aligned} \hat{x}^{i+1} - \frac{4}{3}\hat{x}^{i} + \frac{2}{3}\lambda_i f'^{-1}(\hat{x}^{i+1})f(\hat{x}^{i+1}) + \frac{1}{3}\hat{x}^{i-1} \\ = \frac{2}{3}\lambda_i \big(f'^{-1}(\hat{x}^{i+1}) - f'^{-1}(\hat{x}^{i})\big)f(\hat{x}^{i+1}). \end{aligned} \tag{5.38}$$

Moreover, we have that

$$f'^{-1}(\hat{x}^{i+1}) = f'^{-1}\big(\hat{x}^{i} + \dot{\hat{x}}\lambda_i + \mathcal{O}(\lambda_i)^2\big), \tag{5.39}$$

where $\dot{\hat{x}}$ indicates the time derivative of $\hat{x}$. Using the Taylor expansion, it follows from (5.39) that

$$f'^{-1}(\hat{x}^{i+1}) - f'^{-1}\big(\hat{x}^{i}\big) = \frac{df'^{-1}(\hat{x}^{i})}{dx}\dot{\hat{x}}\lambda_i + \mathcal{O}(\lambda_i)^2. \tag{5.40}$$

The discretisation error bound for (5.38) then reads

$$\| \hat{\delta}(\hat{x}(\tau_{i+1}), \lambda_i) \| \leq \frac{2}{3}\lambda_i L_M C_F \max_{\tau \in (\tau_i, \tau_{i+1})} |\dot{\hat{x}}| + \mathcal{O}(\lambda_i^2), \tag{5.41}$$

where C_F and L_M have been introduced in Section 5.1.

We can improve the accuracy of (5.36) in the following way. In order to simplify the notation, we consider still the 1D problem. Let us set $\hat{x}^{i+1} - x^{i+1} = \delta x^{i+1}$. If expanded in a Taylor series about the point x^{i+1}, f'^{-1} yields

$$f'^{-1}(\hat{x}^{i+1}) = f'^{-1}(x^{i+1}) + \frac{df'^{-1}(x^{i+1})}{dx}\delta x^{i+1} + \mathcal{O}((\delta x^{i+1})^2). \tag{5.42}$$

From (5.32) and considering that $p = 1$, we see that δx^{i+1} is a second order term in λ_i. Bearing in mind that we use (5.35) after the new approximation x^{i+1} has become available, we can replace $f'^{-1}(\hat{x}^{i+1})$ by $f'^{-1}(x^{i+1})$ up to $\mathcal{O}(\lambda_i^2)$. Finally, we are able to write a mixed second order formula to integrate the Davidenko equation. In a multidimensional setting it reads

$$\hat{\mathbf{x}}^{i+1} = \frac{4}{3}\hat{\mathbf{x}}^{i} - \frac{2}{3}\lambda_i \mathbf{J}^{-1}(\mathbf{x}^{i+1})\mathbf{F}(\hat{\mathbf{x}}^{i+1}) - \frac{1}{3}\hat{\mathbf{x}}^{i-1}. \tag{5.43}$$

Note that (5.43) is not fully implicit anymore. Introducing again the function $\hat{\mathbf{G}}(\mathbf{z})$ as in (5.16) and setting $\mathbf{J}(\mathbf{z}^l) = \mathbf{J}(\hat{\mathbf{x}}^i)$, we get the iteration formula

$$\mathbf{z}^{l+1} = \frac{\lambda_i}{1+\lambda_i}\left(\mathbf{z}^l - \tfrac{2}{3}\mathbf{J}^{-1}\left(\mathbf{x}^{i+1}\right)\mathbf{F}\left(\mathbf{z}^l\right)\right) + \frac{1}{1+\lambda_i}\left(\tfrac{4}{3}\hat{\mathbf{x}}^i - \tfrac{1}{3}\hat{\mathbf{x}}^{i-1}\right). \tag{5.44}$$

When using (5.43) to evaluate the local discretisation error, its right-hand side has to be slightly modified. In fact, our analysis is based on the assumption that $\mathbf{x}^{i+1}$ and $\hat{\mathbf{x}}^{i+1}$ are evaluated starting from the same local initial values, i.e. the $\hat{\mathbf{x}}^i = \mathbf{x}^i$ and $\hat{\mathbf{x}}^{i-1} = \mathbf{x}^{i-1}$. This also prevents that the two solution curves evolve along two separate patterns. Thus we can use the following formula

$$\mathbf{z}^{l+1} = \frac{\lambda_i}{1+\lambda_i}\left(\mathbf{z}^l - \tfrac{2}{3}\mathbf{J}^{-1}\left(\mathbf{x}^{i+1}\right)\mathbf{F}\left(\mathbf{z}^l\right)\right) + \frac{1}{1+\lambda_i}\left(\tfrac{4}{3}\mathbf{x}^i - \tfrac{1}{3}\mathbf{x}^{i-1}\right). \tag{5.45}$$

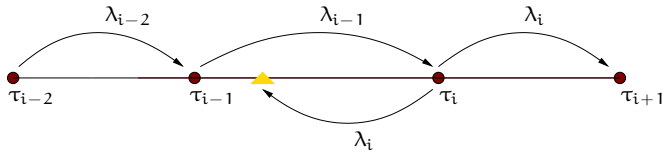


Figure 5.1: In order to get $\mathbf{x}^{i+1}$, $\mathbf{x}^{i-1}$ has to be recomputed at the time level indicated by the triangle, such that $\lambda_{i-1} = \lambda_i$. In this case $\lambda_{i+1} < \lambda_{i-1}$, so (5.46a) is used for the interpolation.

Before introducing the new algorithm, another issue has to be addressed. In fact, equation (5.35) holds for uniform time intervals, i.e. $\lambda_{i-1} = \lambda_i$. Since we rather wish that the time step does change, even rapidly when the solution approaches the steady state, we must find a way to update $\mathbf{x}^{i-1}$ such that $\lambda_{i-1} = \lambda_i$ still holds. Note that $\mathbf{x}^{i-1}$ and $\mathbf{x}^i$ in (5.45) are computed with a BDF1 formula. This implies that a linear approximation is sufficient to get the new value of $\mathbf{x}^{i-1}$ with the desired accuracy, see Figure 5.1as follows

$$\mathbf{x}^{i-1} = \mathbf{x}^{i} - \left(\mathbf{x}^{i} - \mathbf{x}^{i-1}\right)\frac{\lambda_i}{\lambda_{i-1}}, \qquad \text{if} \quad \lambda_i \le \lambda_{i-1}, \tag{5.46a}$$

$$\mathbf{x}^{i-1} = \mathbf{x}^{i-1} + \left(\mathbf{x}^{i-1} - \mathbf{x}^{i-2}\right)\frac{\lambda_i - \lambda_{i-1}}{\lambda_{i-2}}, \quad \text{if} \quad \lambda_i \ge \lambda_{i-1}. \tag{5.46b}$$

In summary, we obtain the following new algorithm

BDF1-BDF2 Algorithm (BDF1/2A)

- Let $\mathbf{x}_0$ be given;
- For $i = 0, 1$
  - Set $\lambda_i = \frac{1}{1+\|\mathbf{F}(\mathbf{x}^i)\|}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$;
    - If the Newton process (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
- For $i = 2, 3, \cdots$
  - Set $\lambda_i = \lambda_{i-1}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$
    - If the Newton process (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
  - Solve (5.43) for $\hat{\mathbf{x}}^{i+1}$ using (5.45) and (5.46);
  - Compute
$$\text{EST} = \| \hat{\mathbf{x}}^{i+1} - \mathbf{x}^{i+1} \|; \tag{5.47}$$
  - Compute the test function
$$\text{TEST} := \frac{\text{EST}}{\text{ATOL} + \text{RTOL} \cdot \| \mathbf{x}^{i+1} \|}; \tag{5.48}$$
  - If $\frac{1}{\rho} \le \text{TEST} \le \rho$ the solution $\mathbf{x}^{i+1}$ is accepted;
  - If $\text{TEST} > \rho$ the discretisation error is too big: the solution $\mathbf{x}^{i+1}$ is rejected. Then
    - Update λ_i according to
$$\lambda_i = \alpha\frac{\lambda_i}{\sqrt{\text{TEST}}}; \tag{5.49}$$
    - Compute a new $\mathbf{x}^{i-1}$ according to (5.46);
  - If $\text{TEST} < \frac{1}{\rho}$ accept the new solution $\mathbf{x}^{i+1}$, increase the step size according to (5.49) and recompute $\mathbf{x}^{i-1}$ with (5.46);
  - If λ has not been changed for the last n; iterations, set $\lambda_i := 2\lambda_i$;

- If TEST $<\rho$ check the stopping criterion on $\| \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^{i+1}) \|$.

It is worthwhile to notice that, like for the MEA, the Jacobi matrix has to be computed only once for each time step. In fact, unless the iteration is rejected, the matrix $\mathbf{J}(\mathbf{x}^{i+1})$ used by the BDF2 formula at the $(i+1)$-th step is also employed by the BDF1 formula at the $(i+2)$-th step. Then the Jacobi matrix has to be recomputed only if the solution at a certain time step is rejected.

5.3.3 Step size control via extrapolation

Whereas in the previous subsection the estimate of the discretisation error was obtained by comparing two solutions at the same time level and computed by BDF formulae of different order, we now introduce a method based on the opposite approach. In fact, we try to approximate δ by comparing two solutions computed with the same scheme: one that reaches τ_{i+1} in one step of length $\lambda_i = \tau_{i+1} - \tau_i$, the other that reaches τ_{i+1} in two steps equal to $\lambda_i/2$; see Figure 5.2. We denote by $\mathbf{x}_{i+1}$ the solution obtained with

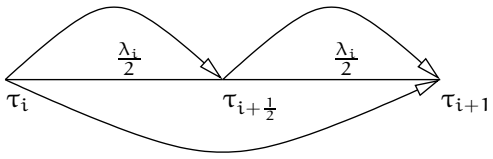


Figure 5.2: The solution of the ODE at the time τ_{i+1} is computed first in one step of size λ_i and then in two steps of size $\frac{\lambda_i}{2}$.

the first procedure; by $\hat{\mathbf{x}}_{i+1}$ the solution obtained with the second. The corresponding discretisation error will be again δ and $\hat{\delta}$, respectively. The general expression for δ when considering a first order method yields

$$\delta(\mathbf{x}(\tau_{i+1}), \lambda) = c\big(\tau_i, \mathbf{x}(\tau_i)\big)\lambda + \mathcal{O}(\lambda^2). \tag{5.50}$$

Using the Davidenko equation, the solution $\hat{\mathbf{x}}^{i+1}$ reads

$$\begin{aligned} \hat{\mathbf{x}}^{i+1} &= \hat{\mathbf{x}}^{i+\frac{1}{2}} - \frac{\lambda}{2}\mathbf{J}^{-1}(\hat{\mathbf{x}}^{i+\frac{1}{2}})\mathbf{F}(\hat{\mathbf{x}}^{i+1}) \\ &= \mathbf{x}^i - \frac{\lambda}{2}\mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\hat{\mathbf{x}}^{i+\frac{1}{2}}) - \frac{\lambda}{2}\mathbf{J}^{-1}(\hat{\mathbf{x}}^{i+\frac{1}{2}})\mathbf{F}(\hat{\mathbf{x}}^{i+1}). \end{aligned} \tag{5.51}$$

Let us suppose that $\mathbf{x}_i = \mathbf{x}(\tau_i)$. To evaluate the discretisation error associated with (5.51) we use again the equivalent 1D equation

$$\begin{aligned}
&\hat{\delta}(x(\tau_{i+1}),\lambda) = \lambda^{-1}\left[x(\tau_{i+1}) - \hat{x}^{i+1}\right] \\
&= \lambda^{-1}\Big[x(\tau_{i+1}) - x(\tau_{i+\frac{1}{2}}) + \frac{\lambda}{2}f'^{-1}(\hat{x}^{i+\frac{1}{2}})f(\hat{x}^{i+1}) \\
&+ x(\tau_{i+\frac{1}{2}}) - x^i + \frac{\lambda}{2}f'^{-1}(x^i)f(\hat{x}^{i+\frac{1}{2}})\Big] \\
&= \lambda^{-1}\Big[c\Big(\tau_i + \tfrac{\lambda}{2}, x(\tau_i + \tfrac{\lambda}{2}) + \mathcal{O}(\lambda^2)\Big)\Big(\frac{\lambda}{2}\Big)^2 + c\Big(\tau_i, x(\tau_i)\Big)\Big(\frac{\lambda}{2}\Big)^2\Big] + \mathcal{O}(\lambda^2) \\
&\doteq c\Big(\tau_i, x(\tau_i)\Big)\Big(\frac{\lambda}{2}\Big)^2 + \mathcal{O}(\lambda) = \frac{1}{2}\delta(x(\tau_{i+1},\lambda)) + \mathcal{O}(\lambda^2).
\end{aligned} \tag{5.52}$$

From the relation

$$x(\tau_{i+1}) = x^{i+1} + \lambda\delta(x(\tau_{i+1},\lambda)) = \hat{x}^{i+1} + \lambda\hat{\delta}(x(\tau_{i+1},\lambda)), \tag{5.53}$$

it follows that

$$\delta(x(\tau_{i+1},\lambda)) = 2\big(\hat{x}^{i+1} - x^{i+1}\big) + \mathcal{O}(\lambda^2). \tag{5.54}$$

The expression (5.54) is the estimate of the local error that we were looking for. To improve the computational efficiency of this method, in (5.51) we replace $\mathbf{J}^{-1}(\hat{x}^{i+\frac{1}{2}})$ with $\mathbf{J}^{-1}(x^i)$. These results can be straightforwardly extended to the multidimensional case. The new algorithm then reads

BDF1-Extrapolation Algorithm (BDF1/EA)

- Let $\mathbf{x}^0$ be given;
- For $i = 0$
  - Set $\lambda_i = \frac{1}{1+\|\mathbf{F}(x^i)\|}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$ using (5.17);
    - * If the Newton processs (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
- For $i = 1, 2, 3, \cdots$
  - Set $\lambda_i = \lambda_{i-1}$;
  - Solve (5.14) for $\mathbf{x}^{i+1}$
    - * If the Newton process (5.17) does not converge after $l = m_1$ iterations, set $\lambda_i = \frac{1}{2}\lambda_i$ and continue iterating until either convergence or $l = m_2$;
  - Set $\hat{\lambda}_i = \frac{\lambda_i}{2}$ and solve (5.14) for $\hat{\mathbf{x}}^{i+\frac{1}{2}}$;
  - Solve again (5.14) for $\hat{\mathbf{x}}^{i+1}$;
  - Compute

$$\text{EST} = 2 \parallel \hat{\mathbf{x}}^{i+1} - \mathbf{x}^{i+1} \parallel; \tag{5.55}$$

- Compute the test function

$$\text{TEST} := \frac{\text{EST}}{\text{ATOL} + \text{RTOL} \cdot \parallel \mathbf{x}^{i+1} \parallel}; \tag{5.56}$$

- If $\frac{1}{\rho} \leq \text{TEST} \leq \rho$ the solution $\mathbf{x}^{i+1}$ is accepted;
- If $\text{TEST} > \rho$ the discretisation error is too big, the solution $\mathbf{x}^{i+1}$ is rejected and a new solution is computed after updating λ_i according to

$$\lambda_i = \alpha \frac{\lambda_i}{\sqrt{\text{TEST}}}; \tag{5.57}$$

- If $\text{TEST} < \frac{1}{\rho}$ accept the new solution $\mathbf{x}^{i+1}$ and increase the step size according to (5.49);
- If λ_i has not been changed for the last n iterations, set $\lambda_i := 2\lambda_i$;
- If $\text{TEST} < \rho$ check the stopping criterion on $\parallel \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^{i+1}) \parallel$.

5.4 Numerical results

In this section we use some benchmark problems to assess the performance of the methods introduced so far. Most of those have been introduced in [1] and then again in [35] as test problems. When we consider a solution computed by the MEA, we refer to our implementation and not to the results reported in [35]. In fact there are some discrepancies in most of the cases. This may have several reasons. First of all, in [35] nothing is said about the criterion to compute the time step corresponding to the first two iterations. Furthermore, the convergence criterion for the Newton solver (5.17) is not specified. These two elements can of course have a strong influence on the transient behaviour of the system and then on the number of iterations necessary to reach the steady state solution. Here, the considered stopping criterion is $\parallel \mathbf{J}^{-1}(\mathbf{x}^i)\mathbf{F}(\mathbf{x}^i) \parallel_\infty \leq 10^{-6}$ for all problems. Moreover, in (5.29), (5.49) and (5.57) we have set $\alpha = 1$. We use the results obtained with MEA as a comparison to assess the characteristics of the other two algorithms.

Example 1
Consider the function, see [11]

$$\mathbf{F}(\mathbf{x}) := \begin{pmatrix} x_1^2 - x_2 + 1 \\ x_1 - \cos(\frac{\pi x_2}{2}) \end{pmatrix}. \tag{5.58}$$

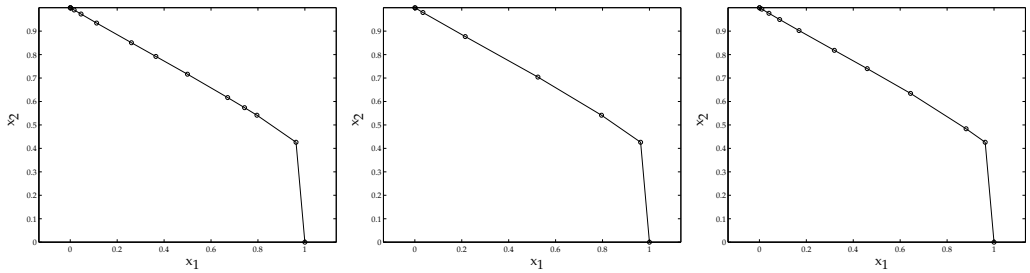
Its solution is $\mathbf{x}^* = (0, 1)^T$. As initial guess we use

- Case 1: $\mathbf{x}^0 = (1, 0)^T$,
- Case 2: $\mathbf{x}^0 = (-1, -1)^T$.

Figures 5.3 and 5.4 show the path of the solutions computed with the three different algorithms for Case 1 and Case 2, respectively. We see that the solution curve of the of Problem 2 runs very close to the curve

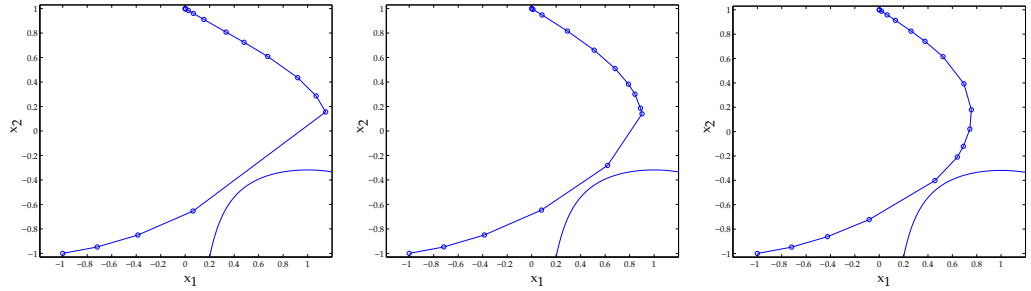
$$\pi x_1 \sin\left(\frac{\pi x_2}{2}\right) = -1, \tag{5.59}$$

where the Jacobi matrix is singular. This implies that a very good control of the step size is necessary for the success of the computation.



(a) MEA. (b) BDF1/2A. (c) BDF1/EA

Figure 5.3: Example 1, Case 1. Transient behaviour from the initial guess $x^0 = (1, 0)^T$ to the steady state solution $x^* = (0, 1)^T$.

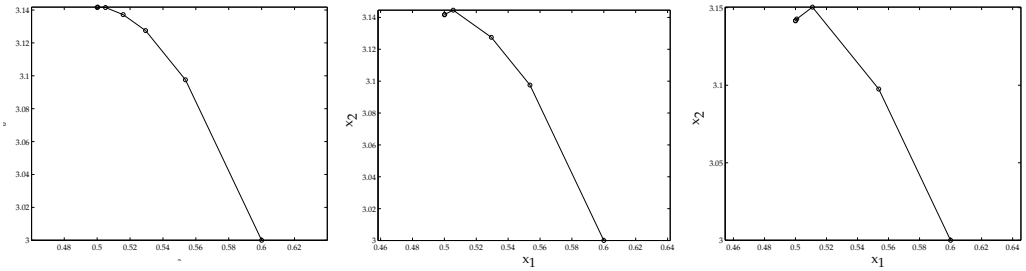


(a) MEA. (b) BDF1/2A. (c) BDF1/EA.

Figure 5.4: Example 1, Case 2. Transient behaviour from the initial guess $x^0 = (-1, -1)^T$ to the steady state solution $x^* = (0, 1)^T$ (line with circles) and curve where the Jacobi matrix is singular.

Example 2
The second problem reads

$$\mathbf{F}(\mathbf{x}) := \begin{pmatrix} \frac{1}{2}\sin(x_1 x_2) - \frac{x_2}{4\pi} - \frac{x_1}{2} \\ \left(1 - \frac{1}{4\pi}\right)(e^{2x_1} - e) + \frac{e x_2}{\pi} - 2e x_1 \end{pmatrix}, \tag{5.60}$$



(a) MEA (b) BDF1/2A (c) BDF1/EA

Figure 5.5: Example 2. Transient behaviour from the initial guess $x^0 = (0.6, 3)^T$ to the steady state solution $x^* = (0.5, \pi)^T$.

whose solution is $\mathbf{x}^* = (0.5, \pi)^T$, see Figure 5.5. We start integrating from $\mathbf{x}^0 = (0.6, 3)^T$.

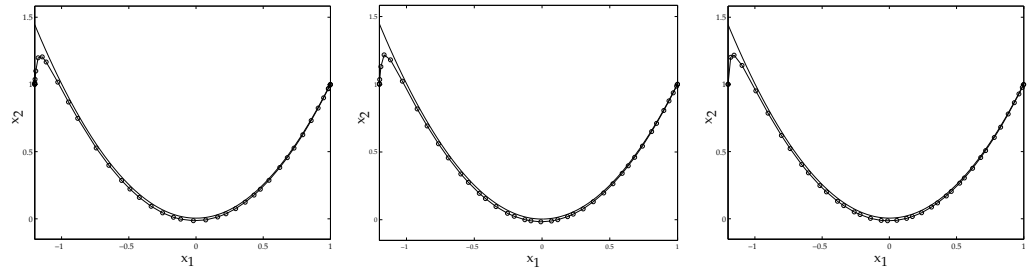
Example 3
Our final 2D test problem is now

$$\mathbf{F}(\mathbf{x}) := \begin{pmatrix} 400x_1(x_1^2 - x_2) + 2(x_1 - 1) \\ -200(x_1^2 - x_2) \end{pmatrix}, \tag{5.61}$$

whose solution is $\mathbf{x}^0 = (1, 1)^T$, see Figure 5.6. We use as initial guess the following values

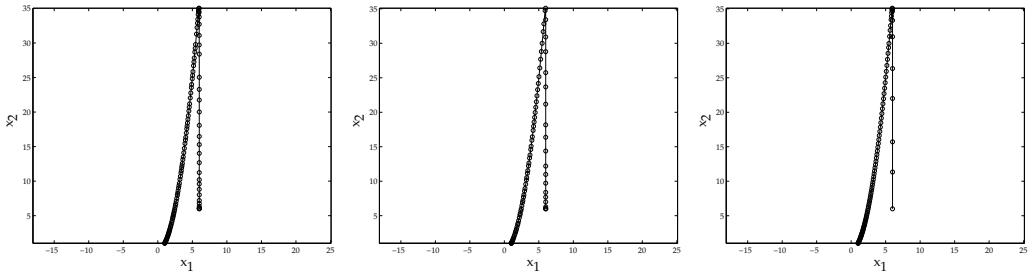
- Case 1: $\mathbf{x}^0 = (-1.2, 1.0)^T$,
- Case 2: $\mathbf{x}^0 = (6.0, 6.0)^T$,
- Case 3: $\mathbf{x}^0 = (20.0, 20.0)^T$.

From Figures 5.6, 5.7 and 5.8, we see that the solution, from whichever point we start, moves toward the parabola $x_1^2 = x_2$ to which the exact solution belong and that lies very close to the curve $x_1^2 = x_2 - 0.005$, where the Jacobi matrix is singular; see also [35].



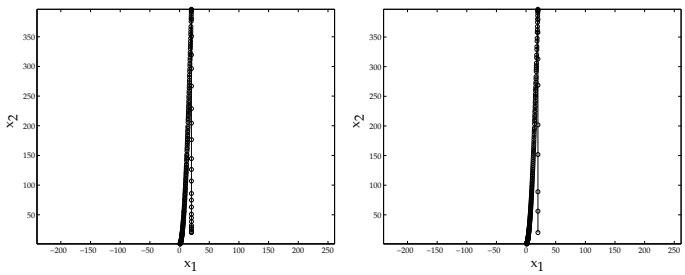
(a) MEA (b) BDF1/2A (c) BDF1/EA

Figure 5.6: Example 3, Case 1. Transient behaviour from the initial guess $\mathbf{x^0} = (-1.2, 1)^\mathsf{T}$ to the steady state solution $\mathbf{x}^* = (1, 1)^\mathsf{T}$ (line with circles) and curve where the Jacobi matrix becomes singular (solid line).



(a) MEA (b) BDF1/2A (c) BDF1/EA

Figure 5.7: Example 3, Case 2. Transient behaviour from the initial guess $\mathbf{x^0} = (6, 6)^\mathsf{T}$ to the steady state solution $\mathbf{x}^* = (1, 1)^\mathsf{T}$.



(a) BDF1/2A (b) BDF1/EA

Figure 5.8: Example 3, Case 3. Transient behaviour from the initial guess $\mathbf{x^0} = (20, 20)^\mathsf{T}$ to the steady state solution $\mathbf{x}^* = (1, 1)^\mathsf{T}$.

Example 4
We consider now the BVP

$$3u\frac{\partial^2 u}{\partial x^2} + \left(\frac{\partial u}{\partial x}\right)^2 = 0, \qquad 0 < x < 1, \tag{5.62a}$$

$$u(0) = 0, \tag{5.62b}$$

$$u(1) = 20. \tag{5.62c}$$

see [35]. Equation (5.62a) is discretised by finite differences on a uniform grid using N grid points. The exact solution is given by the curve $u(x) = 20\,x^{\frac{2}{3}}$. We have solved the problem on two grids, viz.

- Case 1: $N = 10$,
- Case 2: $N = 20$.

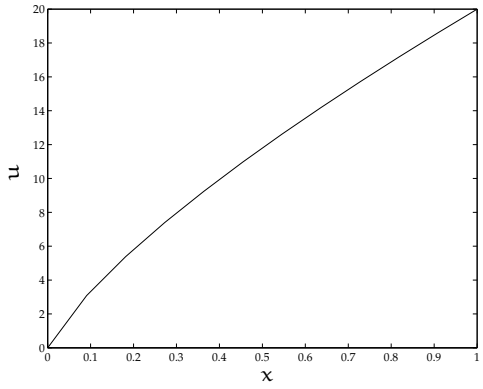


Figure 5.9: Solution of BVP (5.62). The curve here shown has been obtained by using the BDF1/2A with $N = 20$ grid points.

For both tests, the constant initial guess $\mathbf{u}^0 = 10$ has been chosen. Figure 5.9 shows the numerical solution computed with BDF1/2A when $N = 20$.

Example 5
Consider the BVP, see [27],

$$\frac{d}{dx}\left(x^2\frac{du}{dx}\right) = x^2 f(u), \qquad 0 < x < 1, \tag{5.63a}$$

$$\frac{du}{dx}(0) = 0, \tag{5.63b}$$

$$u(1) = 1, \tag{5.63c}$$

where

$$f(u) = \varepsilon^{-1}\frac{u}{u+k}, \tag{5.64}$$

with ε and k positive. We take $k = 0.1$ and consider the following three cases

- Case 1: $\varepsilon = 10^{-3}$,
- Case 2: $\varepsilon = 10^{-4}$,
- Case 3: $\varepsilon = 10^{-5}$.

Discretisation of (5.63a) by finite differences yields

$$(u_2 - u_1)(x_{\frac{3}{2}}^2 - \tfrac{1}{3}x_{\frac{1}{2}}^2) = x_1^2 \Delta x^2 f(x_1), \qquad j = 1, \tag{5.65a}$$

$$x_{j+\frac{1}{2}}^2 (u_{j+1} - u_j) - x_{j-\frac{1}{2}}^2 (u_j - u_{j-1}) = x_i^2 \Delta x^2 f(x_j), \qquad j = 2, \cdots, N-2, \tag{5.65b}$$

$$x_{N-\frac{1}{2}}^2 (1 - u_{N-1}) - x_{N-\frac{3}{2}}^2 (u_{N-1} - u_{N-2}) = x_{N-1}^2 \Delta x^2 f(x_{N-1}), \quad j = N-1, \tag{5.65c}$$

where $\Delta x = \frac{1}{N+1}$. The derivation of (5.65b) and (5.65c) is straightforward. We have derived equation (5.65a) in the following way. At the first point of the interval, $\mathbf{x}_0$, the boundary condition (5.63b) is discretised by using the second order accurate one-sided formula

$$\frac{-3u_0 + 4u_1 - u_2}{\Delta x} = 0. \tag{5.66}$$

From (5.66), u_0 can be expressed as a function of u_1 and u_2 and subsequently substituted in the difference scheme. For all test problems we take $N = 200$ and u_i^0 as initial guess, where $u_i^0 = (1 - \varepsilon k)x_i^2$, for $i = 1, \cdots, N$. Figure 5.10 shows the solution computed with the BDF1/EA.

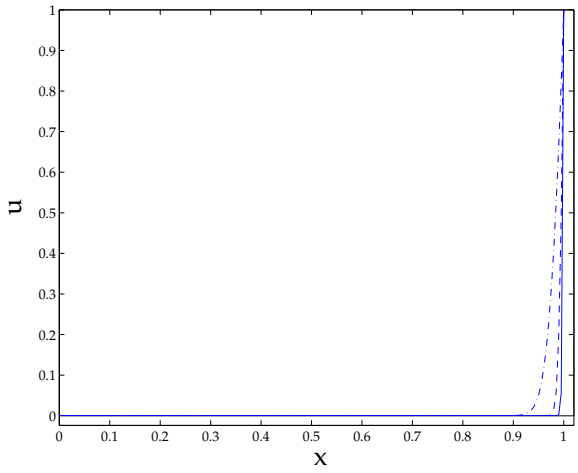


Figure 5.10: Solution of BVP (5.63). Dash-dot line: $\varepsilon = 10^{-3}$; dashed line: $\varepsilon = 10^{-4}$; solid line: $\varepsilon = 10^{-5}$.

Example 6
The last example is the following

$$\frac{d^2u}{dx^2} = \sinh(nu), \qquad 0 < x < 1, \tag{5.67a}$$

$$u(0) = 0, \qquad u(1) = 1, \tag{5.67b}$$

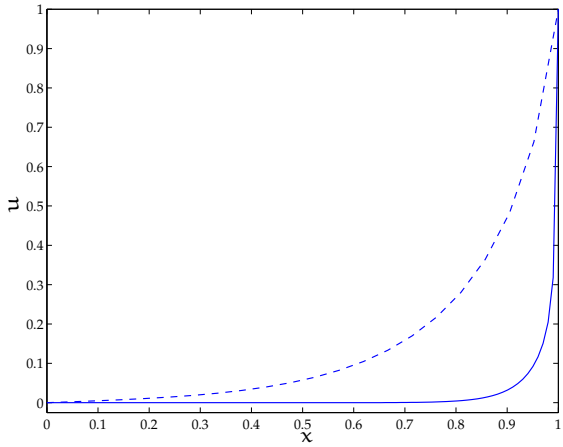


Figure 5.11: Solution of BVP (5.67). Dotted line: $n = 5$; solid line: $n = 20$

where n is an integer. We consider the following two cases

- Case 1: $n = 5$ and $N = 20$,
- Case 3: $n = 20$ and $N = 100$.

The solution is shown in Figure 5.11. We start with the following initial guess

$$u^0(x) = x. \tag{5.68}$$

We compare now the three methods presented in terms of efficiency and robustness. All problems are solved with ATOL $=$ RTOL $= 10^{-1}$, with the exception of Example 3, where ATOL $=$ RTOL $= 10^{-2}$ are used. Tables 5.1 and 5.4 refers to the MEA. We indicate with TotalIt the number of iterations, with nJ the number of Jacobi matrix evaluations, with nF the number of function evaluations. It has to be noted that, for this algorithm, the number of iterations is equal to or larger than the number of Jacobi matrix evaluations. In fact, when a solution at a certain time step is rejected because the corresponding test function does not comply with the condition TEST $\leq \rho$, the Jacobi matrix can still be used for a new iteration. The considered convergence criterion for the Newton process (5.17) is $\| \mathbf{z}^{l+1} - \mathbf{z}^{l} \|_\infty \leq$ TolInt. The value of TolInt is always equal to 10^{-1} except that in Example 3, Case 1 for the MEA and the BDF1/2A where it is set equal to 10^{-2}.

Tables 5.2 and 5.5 show the results obtained solving the benchmark problems with the BDF1/2A. We have now to consider that the number of Jacobi matrix evaluations is equal to the total number of iterations. In fact, $\mathbf{J}^{-1}(\mathbf{x}^{i+1})$ is computed after the $(i+1)$-th BDF1 step and is used both to compute the $\hat{\mathbf{x}}^{i+1}$- and the $\mathbf{x}^{i+2}$-solutions. If $\mathbf{x}^{i+1}$ is rejected, the $\mathbf{J}^{-1}(\mathbf{x}^{i+1})$ evaluation is useless. Although this could be seen as a drawback of the BDF1/2A, both Tables 5.5 and 5.2 show that the number of iterations rejected is quite low with respect to the total number. The row ValidIt refers then to the number of iterations that have been accepted.

Finally, the solutions obtained with the BDF1/EA are shown in Table 5.3 and Table 5.6. Here again the number of Jacobi matrix computations is equal to the number of accepted iterations and the row TotalIt indicates the total number of iterations performed.

Let us now compare the results shown in the previous tables. The first thing to notice is that in one case the MEA fails, i.e. it reveals to be not robust enough when the initial guess is far from the zero of the system. This is a very important issue, since our main goal was to build an algorithm that could improve robustness of the existing ones. The better capability of BDF1/2A and BDF1/EA in dealing with more complex problems appears also when considering, for instance, Example 6. There, the control of the discretisation error leads to a much faster convergence compared with MEA. Furthermore, if we look at the computational work, we see that both the BDF1/2A and the BDF1/EA are very competitive compared to the MEA when considering the number of Jacobi matrix evaluations. This is a quite important advantage, since the evaluation of the Jacobi matrix is the most expensive and time consuming part of each iteration step. The only point for which the MEA appears to be better than the other two is the number of function evaluations. This is unavoidable, since both the BDF1/2A and the BDF1/EA compute twice the solution at a certain time level τ_{i+1}.

Example	1		2	3		
Case	1	2	1	1	2	3
TotalIt	14	19	8	52	97	FAIL
nJ	14	18	8	52	94	–
nF	23	31	10	173	315	–

Table 5.1: Examples 1 to 3 solved with the MEA. Tolerances and computational work.

Example	1		2	3		
Case	1	2	1	1	2	3
ValidIt	10	14	5	45	79	234
nJ	10	14	5	45	84	244
nF	29	52	10	202	372	1585

Table 5.2: Examples 1 to 3 solved with the BDF1/2A. Tolerances and computational work.

Example	1		2	3		
Case	1	2	1	1	2	3
TotalIt	9	17	5	49	83	185
nJ	9	17	5	43	83	185
nF	42	151	16	166	514	2126

Table 5.3: Problems 1 to 3 solved with the BDF1/EA. Tolerances and computational work.

Example	4		5			6	
Case	1	2	1	2	3	1	2
TotalIt	23	23	23	29	22	11	39
nJ	21	21	22	28	23	11	31
nF	45	45	34	36	34	15	41

Table 5.4: Examples 4 and 5 and 6 solved with the MEA. Tolerances and computational work.

Example	4		5			6	
Case	1	2	1	2	3	1	2
ValidIt	15	16	24	23	15	9	22
nJ	16	17	24	23	15	9	22
nF	121	116	56	56	41	26	44

Table 5.5: Problems 4 and 5 and 6 solved with the BDF1/2A. Tolerances and computational work.

Example	4		5			6	
Case	1	2	1	2	3	1	2
ValidIt	14	14	25	25	16	7	22
nJ	14	14	25	25	16	7	22
nF	87	92	98	109	90	26	67

Table 5.6: Problems 4 and 5 and 6 solved with the BDF1/EA. Tolerances and computational work.

CHAPTER 6

Laminar flame simulation

In this chapter we apply the LDC technique with curvilinear local grid refinement to simulate the two flame models described in Chapter 2. We have thus two main sections: in the first one the thermo-diffusive model is discussed, in the second one the Bunsen flame simulation is presented.

6.1 The thermo-diffusive model

The thermo-diffusive model introduced in Chapter 2 is here further elaborated. In Section 6.1.1. some dimensionless quantities are introduced and equations are scaled. Moreover, with the assumption that the *Lewis number* is equal to 1, they are transformed into a single equation for the temperature whose steady state solution describes a travelling wave, see [24]. Next, in Section 6.1.2 the solution strategy is outlined adapting the LDC standard method to this specific problem. Finally, numerical results are presented.

6.1.1 Governing equations

We recall that the thermo-diffusive model is based on the following two main assumptions: the *isobaric approximation*, i.e. the pressure is considered constant in space, and the *constant density approximation*. As a consequence, we can split the conservation laws in the flow equations, to compute the velocity $\mathbf{v}$ and the pressure p, and the combustion equations. The latter read

$$\rho \frac{\partial Y}{\partial t} + \rho \mathbf{v} \cdot \boldsymbol{\nabla} Y = \rho \boldsymbol{\nabla} \cdot (D \nabla Y) - M \omega(\rho Y, T), \tag{6.1a}$$

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p \mathbf{v} \cdot \boldsymbol{\nabla} T = \boldsymbol{\nabla} \cdot (\lambda \nabla T) + M Q \omega(\rho Y, T), \tag{6.1b}$$

where Y, T and Q are the reactant mass fraction, the temperature and the heat release per unit mass of fuel consumption, respectively. Other parameters in (6.1) are the diffusion coefficient D, the molar mass of the mixture M, the specific heat at constant pressure c_p,

the thermal conductivity λ and the universal gas constant R. Moreover, the source term in (6.1) is given by the Arrhenius expression

$$\omega(\rho Y, T) = \frac{\rho Y}{M} B \exp\left(-\frac{E}{RT}\right). \tag{6.2}$$

Next, we apply the scaling introduced in [47] to system (6.1). Let the subscripts b and u to denote the burnt and unburnt gases, respectively, then we define the variables

$$\Theta := \frac{T - T_u}{T_b - T_u}, \qquad Z := \frac{Y}{Y_u}, \tag{6.3}$$

and the parameters

$$\alpha := \frac{T_b - T_u}{T_b}, \qquad \beta := \frac{E(T_b - T_u)}{RT_b^2}. \tag{6.4}$$

So Θ and Z are the dimensionless temperature and the normalised mass fraction, respectively. Moreover the coefficient α represents a nondimensional heat release parameter and the coefficient β the dimensionless activation energy, also known as *Zeldovich number*, [68]. An additional dimensionless parameter that characterises the behaviour of the thermo-diffusive model solution is the *Lewis number*, defined as

$$Le := \frac{\lambda}{\rho c_p D}. \tag{6.5}$$

It represents the ratio between thermal conduction and mass diffusion and is an important measure of the stability of a flame, [47]. Substituting (6.3), (6.4) and (6.5) in (6.1), we get

$$\frac{\partial \Theta}{\partial t} = \nabla^2 \Theta + w(Z, \Theta), \tag{6.6a}$$

$$\frac{\partial Z}{\partial t} = \frac{1}{Le} \nabla^2 Z - w(Z, \Theta), \tag{6.6b}$$

with

$$w(Z, \Theta) = \frac{\beta^2}{2Le} Z \exp\left(-\frac{\beta(1-\Theta)}{1-\alpha(1-\Theta)}\right). \tag{6.7}$$

In (6.6) the velocity field is set equal to 0 and Le and λ are assumed to be constant. The treatment of a laminar flame by means of activation energy asymptotics allows us to write the source term as in (6.7), see [47].

Our goal is to solve equations (6.6) in a channel with open ends whose walls are adiabatic and inert, see Figure 6.1.

Let us now formulate the boundary conditions for such a problem. The value of the temperature at the burn and unburnt gas sides, as well as the value for the reactant mass fractions, are easily obtained when definitions (6.3) are considered. Since at $x = -\infty$ we have the completely unburnt gas and at $x = \infty$ the completely burnt gas, we get

$$\Theta(-\infty, y, t) = 0, \qquad \Theta(\infty, y, t) = 1, \tag{6.8a}$$

$$Z(-\infty, y, t) = 1, \qquad Z(\infty, y, t) = 0. \tag{6.8b}$$

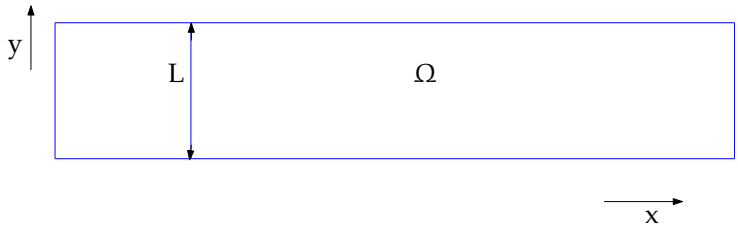


Figure 6.1: Computational domain Ω.

Furthermore, on the horizontal walls we have

$$\frac{\partial \Theta(x,0,t)}{\partial y} = \frac{\partial \Theta(x,L,t)}{\partial y} = 0, \tag{6.9a}$$

$$\frac{\partial Z(x,0,t)}{\partial y} = \frac{\partial Z(x,L,t)}{\partial y} = 0. \tag{6.9b}$$

Condition (6.9a) is found from setting to zero the heat flux through the walls, whereas condition (6.9b) expresses the absence of any mechanism able to enhance or inhibit any chemical reaction. Moreover, the initial conditions for both the normalised temperature and mass fraction, i.e. $\Theta(x,y,t) = \Theta_0(x,y)$, $Z(x,y,t) = Z_0(x,y)$, are needed and will be specified later.

Let us now consider a 2D curved flame propagating in a nonuniform gas flow. We assume $Le = 1$, i.e. that Θ and Z are similar. This implies a stable flame, see [47]. Adding both equations in (6.6) and considering the form of the boundary conditions, the choice $Le = 1$ implies also that $\Theta + Z = 1$. Thus we restrict ourselves to the Θ - equation. Furthermore, we impose the nonuniform velocity field given by

$$\mathbf{v} = \begin{bmatrix} V\cos\left(\frac{\pi y}{2L}\right) \\ 0 \end{bmatrix}, \tag{6.10}$$

that is parallel to the walls. It is easy to verify that the velocity field (6.10) is also divergence-free, thus satisfying the continuity equation. Thus we have

$$\frac{\partial \Theta}{\partial t} + V\cos\left(\frac{\pi y}{2L}\right)\frac{\partial \Theta}{\partial x} = \nabla^2\Theta + w(1-\Theta,\Theta). \tag{6.11}$$

In [8] it is shown that equation (6.11) allows for a travelling wave solution of the form $\Theta(x + V_0 t, y)$. In this context, V_0 represents the velocity of the travelling wave (the flame front) that is directed towards the unburnt gases, i.e. the left part of the computational domain. This implies that (6.11) reduces to the steady equation

$$-\nabla^2\Theta + \left(V_0 + V\cos\left(\frac{\pi y}{2L}\right)\right)\frac{\partial \Theta}{\partial x} = w(1-\Theta,\Theta), \tag{6.12}$$

subject to the following boundary conditions

$$\Theta(-\infty, y) = 0, \qquad \Theta(\infty, y) = 1, \tag{6.13a}$$

$$\frac{\partial \Theta(x, 0)}{\partial y} = \frac{\partial \Theta(x, L)}{\partial y} = 0. \tag{6.13b}$$

The expression for the velocity V_0 can be obtained by integrating (6.12) over the entire computational domain Ω, giving

$$V_0 = \frac{1}{L} \iint_{\Omega} w \, dA - \frac{2V}{\pi}. \tag{6.14}$$

It is worth noting that the integral of $\nabla^2\Theta$ is equal to zero: physically, this means that there is no energy exchange between the fluid and the external environment.

Equation (6.12) expresses the balance between the diffusion, convection and reaction contributions in a frame of reference attached to the flame front. In fact, while the gas flow moves from left to right with a speed given by (6.10), the flame front moves at velocity V_0 from right to left. Thus, an observer travelling with this velocity, perceives the gas flow approaching with speed $V_0 + V\cos(y\pi/2L)$.

Let us see what happens when we let V increase. Since we expect that increasing values of V do not lead to an equivalent growth of the reaction term, and therefore of its integral over the domain Ω, expression (6.14) implies that V_0 can assume negative values. If V is large enough in absolute value, it may happen that in some parts of the tube also the term $V_0 + V\cos\left(\pi y/2L\right)$ becomes negative. This behaviour corresponds to a non-classical, nevertheless physical, situation. In fact, while the gas velocity $V_0 + V\cos(y\pi/2L)$ is usually positive, pointing from the fresh mixture towards the burnt gases, there are some parts of the domain where an inversion of the flow is produced. Although this situation has never been reproduced by experiments dealing with a flame in a tube, there are observations pertaining to diffusion flames, see [68], that confirm that the velocity may be negative in proximity of the flame front. This is explained by assuming that local diffusion dominates the convective contribution. We will see that this effect gives the flame front a very peculiar shape that makes it suitable for being studied with a local defect correction approach in combination with curvilinear grids.

6.1.2 Solution strategy and numerical results

Our goal is now to solve (6.12) together with (6.14) both in the global and in a solution-fitted local domain. The transformation from Cartesian to curvilinear coordinates for the local BVP is performed by applying (4.36a) and (4.37).

To solve the thermo-diffusive model, we restrict the domain to a segment Ω of the infinite tube, with $\Omega = (-8.1, 8.1) \times (0, 4)$. In this case the expression of V_0 given by (6.14) is no longer exact as the computational domain has finite length and therefore the integral of $\nabla^2\Theta$ over Ω is not exactly equal to zero. Nevertheless, if Ω is long enough, i.e. if the boundaries are sufficiently far away from the flame front, the use of (6.14) allows to obtain accurate results.

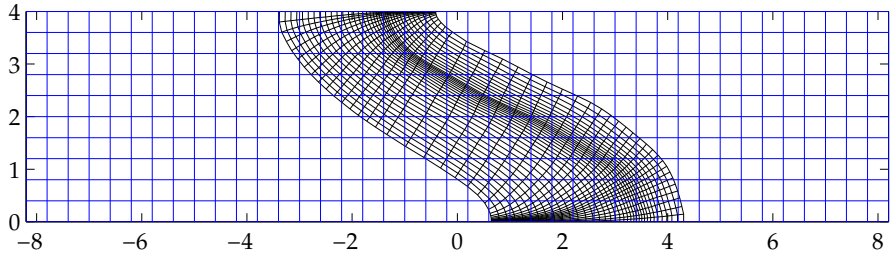


Figure 6.2: Coarse and fine grids.

We choose $\beta = 10$, $\alpha = 0.84$ and set $V = 3$ and use central difference discretisation for both the global and the local problem. The coarse grid size is uniform and equal to $H = 4.0 \cdot 10^{-1}$. We build the local fine grid by using the following procedure. Considering that the normalized temperature ranges between 0 and 1, three level curves are drawn at $\Theta_a = \epsilon_a$, Θ_b and $\Theta_c = 1 - \epsilon_c$, with $\epsilon_a = 2.0 \cdot 10^{-1}$ and $\epsilon_c = 1.0 \cdot 10^{-2}$. We choose the level curve $\Theta = \Theta_b$ at the location where w reaches its maximum value, which can be computed analytically. Across the Θ_b iso-line, $h_{\xi_0} = H/8$ is set. Then the grid sizes on the left and on the right of Θ_b are computed according to $h_i = r_{i-1} h_{i-1}$, where we set

$$r_{i-1} = \min\left(\frac{w_{i-1}}{w_i}, 1.1\right).$$

Furthermore, $h_\eta = H$. The curves in the intervals (Θ_a, Θ_b) and (Θ_b, Θ_c) are not real iso-lines, but are determined by spline interpolation from the aforementioned three lines, see Figure 6.2.

Next, we consider relation (6.14). There, the integral of the source term w over the domain Ω is computed by the Simpson formula for the 2-dimensional rectangular element, i.e.

$$\begin{aligned} I \cong \frac{(x_{i+1} - x_i)(y_{j+1} - y_j)}{36} \Big[& w(i,j) + w(i+1,j) + w(i,j+1) + w(i+1,j+1) \\ & + 4\Big(w(i,j+\tfrac{1}{2}) + w(i+1,j+\tfrac{1}{2}) + w(i+\tfrac{1}{2},j) + w(i+\tfrac{1}{2},j+1)\Big) \\ & + 16 w(i+\tfrac{1}{2},j+\tfrac{1}{2})\Big]. \end{aligned} \tag{6.15}$$

We notice that the problems solved in the global and in the local domain are different. The first one consists of both equation (6.12) for the temperature and relation (6.14) for the velocity. In fact, V_0 represents a part of the convection coefficient and it must be updated at each iteration for the temperature field. Conversely, when solving the local problem, V_0 is not adapted after each iteration, since it is essentially a global variable. Therefore, it is only updated after the restriction step. Both the local and the global problems are solved using MEA, introduced in Chapter 5. As initial solution in the

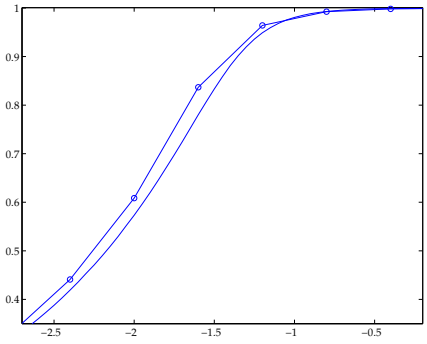


Figure 6.3: Coarse grid solution and composite grid solution after one LDC iteration

global domain we consider the following

$$\Theta(x, y) = 1 - \tanh\Big(- x + \cos\big(\frac{\pi y}{2L}\big)\Big), \tag{6.16}$$

for the inizialisation step in LDC. The coarse grid solution is computed using *continuation* in the parameter β. First, problem (6.12) is solved with $\beta = 1$, then its solution is used as initial guess for an updated problem with $\beta + \delta\beta$, until the value $\beta = 10$ is reached.

It is worth noting that the solution of (6.12) is determined up to a translation. In order to keep the flame front in a certain position, we fix the temperature in one single grid point.

We have performed 2 LDC iterations. After one iteration we find a maximum difference between the composite grid solution and the coarse grid solution of $7 \cdot 10^{-2}$. After the second iteration, we find a maximum difference of $1.3 \cdot 10^{-3}$ between the new composite grid solution and the one at the previous step. This allows to state that after the first iteration a very good approximation of the sought solution is already obtained. In Figure 6.3 we show the temperature profiles of the composite grid solution after one LDC iteration and the coarse grid solution. They are taken at $y = 3.7$.

The speed V_0 converges to -0.4150. While Figures 6.4 and 6.6 show the nondimensional temperature and reaction rate in the coarse grid, Figure 6.5 and 6.7 show the same functions computed in the fine grid after one LDC step. The two-norm of the fine grid solution residual is found to be of the order of 10^{-9}. We see that the solution shows a curved flame front and the expected inversion of the flow. In fact, while in the lower part of the channel the velocity relative to the flame front remains positive, it becomes negative and equal to V_0 for $y = L$. We notice also that, the boundary conditions for the temperature at $y = 0$ and $y = L$ imply that the isotherms are orthogonal to the walls, as well as our local grid.

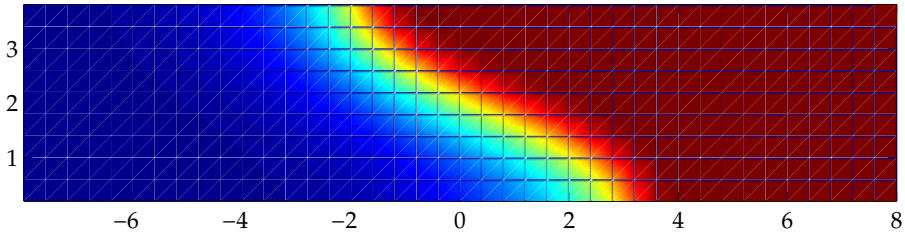


Figure 6.4: Normalised temperature. Coarse grid solution.

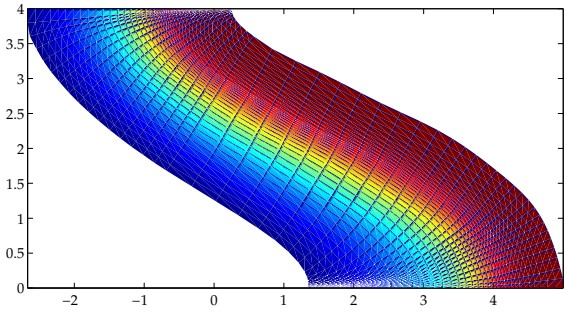


Figure 6.5: Normalised temperature. Fine grid solution.

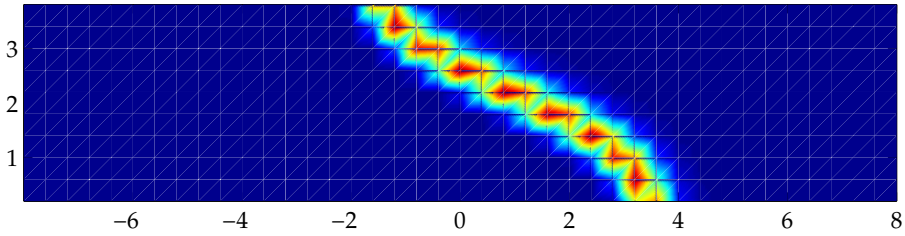


Figure 6.6: Normalised reaction rate. Coarse grid solution.

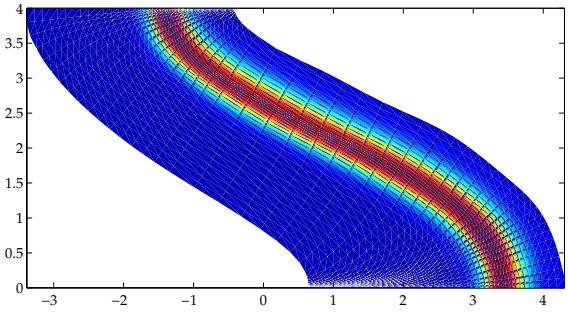


Figure 6.7: Normalised reaction rate. Fine grid solution.

6.2 Bunsen flames

The second section of the chapter is organised as follows. In Section 6.2.1 we recall the mathematical model, discuss the choice of the primary velocity variables and transform the governing equations from Cartesian to curvilinear coordinates. In Section 6.2.2 the discretisation of the combustion system is presented and the *Patankar-flux* approximation introduced. There, we discuss also the discretisation of the viscous terms. Next, in Section 6.2.3 we explain the used solution strategy. Moreover, the pressure correction method is introduced and adapted to general coordinates. Numerical experiments to test the equations are presented in Section 6.2.4, where also the pressure wiggles issue is addressed. Finally, Section 6.2.5 is devoted to show the results of Bunsen flame simulations.

6.2.1 Governing equations

Let us recall the conservation equations of the laminar flame model derived in Chapter 2. They read

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v}) = 0, \tag{6.17a}$$

$$\frac{\partial \rho Y_i}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} Y_i) = \boldsymbol{\nabla} \cdot (\rho D_{im} \boldsymbol{\nabla} Y_i) + s_i, \qquad i = 1, \cdots, N_s - 1, \tag{6.17b}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} \otimes \mathbf{v}) = -\boldsymbol{\nabla} \cdot \mathbf{P} + \rho \mathbf{g}, \tag{6.17c}$$

$$\frac{\partial \rho h}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{v} h) = \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) + \boldsymbol{\nabla} \cdot \left(\rho \sum_{i=1}^{N_s} h_i D_{im} \boldsymbol{\nabla} Y_i \right). \tag{6.17d}$$

These equations have to be completed with the equation of state

$$\frac{p}{\rho} = RT \sum_{i=1}^{N_s} \frac{Y_i}{M_i}, \tag{6.18}$$

and the equations that specify h as a function of Y_i and T

$$h = \sum_{i=1}^{N_s} Y_i h_i, \qquad h_i = h_i^0 + \int_{T^0}^{T} c_{p,i}(\tau)\, d\tau. \tag{6.19}$$

We consider a one-step reaction mechanism where the number of species is $N_s = 5$, see e.g. [30]. For the sake of completeness, we repeat the expression for the stress tensor $\mathbf{P}$

$$\mathbf{P} = \Big(p + \frac{2}{3} \mu (\boldsymbol{\nabla} \cdot \mathbf{v}) \Big) \mathbf{I} - \mu \Big((\boldsymbol{\nabla} \otimes \mathbf{v}) + (\boldsymbol{\nabla} \otimes \mathbf{v})^T \Big) = p\mathbf{I} + \boldsymbol{\tau}, \tag{6.20}$$

where $\boldsymbol{\tau}$ represents the contribution of the viscous stresses.

System (6.17) has to be solved first in the global domain representing the Bunsen burner and then in the local domain defined by the shape of the flame front. This last problem requires the transformations of the combustion and flow equations from Cartesian

to curvilinear coordinates. While the combustion equations (6.17b) and (6.17d) can be straightforwardly rewritten, the reformulation of the momentum equations (6.17c) is more difficult. The reason for this is that we have to make the proper choice for the primary velocity components. In fact, we have three possibilities: Cartesian, covariant and contravariant components can be used. The latter are defined in (6.22). It has been shown, see i.e. [53], that the use of the Cartesian velocity components implies that the equations can be written in a strong conservative form. This is quite an interesting feature for us, in that it not only leads to comparatively less complex equations, but matches also well with the finite volume discretisation. Moreover, the form of the resulting flow equations is such that it fits with the laminar flame code on top of which we build our subroutines. Besides the Cartesian velocities, we also use the contravariant velocity components in both the flow and the combustion equations to evaluate the convective fluxes across the surfaces of the control volumes.

Let us then introduce the contravariant base vectors. If $\mathbf{i}$ and $\mathbf{j}$ are the unit vectors parallel to the Cartesian x- and y-axis, respectively, and J the Jacobian of the transformation from the Cartesian to the curvilinear coordinates, as introduced in (3.32), the contravariant unit vectors are given by

$$\tilde{\mathbf{a}}^{\xi} = \frac{1}{J}\Big[\mathbf{i}\frac{\partial y}{\partial \eta} - \mathbf{j}\frac{\partial x}{\partial \eta}\Big], \tag{6.21a}$$

$$\tilde{\mathbf{a}}^{\eta} = \frac{1}{J}\Big[-\mathbf{i}\frac{\partial y}{\partial \xi} + \mathbf{j}\frac{\partial x}{\partial \xi}\Big]. \tag{6.21b}$$

Both $\tilde{\mathbf{a}}^{\xi}$ and $\tilde{\mathbf{a}}^{\eta}$ are orthogonal to the ξ-and the η-lines, respectively, with $\partial x/\partial \xi$, $\partial x/\partial \eta$, $\partial y/\partial \xi$ and $\partial y/\partial \eta$ the coefficients introduced in Section 4.4. Associated with the contravariant unit vectors is the contravariant velocity $\tilde{\mathbf{U}}$, whose components read

$$\tilde{U} = \mathbf{v}\cdot\tilde{\mathbf{a}}^{\xi} = \frac{u\frac{\partial y}{\partial \eta} - v\frac{\partial x}{\partial \eta}}{J}, \tag{6.22a}$$

$$\tilde{V} = \mathbf{v}\cdot\tilde{\mathbf{a}}^{\eta} = \frac{-u\frac{\partial y}{\partial \xi} + v\frac{\partial x}{\partial \xi}}{J}. \tag{6.22b}$$

In the following we use $\mathbf{a}^{\xi} = J\tilde{\mathbf{a}}^{\xi}$, $\mathbf{a}^{\eta} = J\tilde{\mathbf{a}}^{\eta}$ and the velocity components $U = J\tilde{U}$, $V = J\tilde{V}$, rather than the ones defined in (6.22). This allows us to express the transformed equations in a slightly lighter form.

We are now ready to restate the expressions for the differential operators in curvilinear coordinates. We assume the grids to be orthogonal and consider the transformation of (6.17) term by term. Let us start with the convective contribution. For a generic variable A, see [62], the following holds

$$\boldsymbol{\nabla}\cdot(A\mathbf{v}) \to \frac{1}{J}\frac{\partial(AU)}{\partial \xi} + \frac{1}{J}\frac{\partial(AV)}{\partial \eta}. \tag{6.23}$$

The transformation of the gradient reads

$$\boldsymbol{\nabla}A \to \frac{1}{J}\frac{\partial(\mathbf{a}^{\xi}A)}{\partial \xi} + \frac{1}{J}\frac{\partial(\mathbf{a}^{\eta}A)}{\partial \eta}. \tag{6.24}$$

If we consider the diffusion term, we get

$$\boldsymbol{\nabla}\cdot(\mu\boldsymbol{\nabla}A) \rightarrow \frac{1}{J}\frac{\partial}{\partial\xi}\Big(\frac{\mu g_{\eta\eta}}{J}\frac{\partial A}{\partial\xi}\Big)+\frac{1}{J}\frac{\partial}{\partial\eta}\Big(\frac{\mu g_{\xi\xi}}{J}\frac{\partial A}{\partial\eta}\Big), \tag{6.25}$$

with $g_{\eta\eta}$ and $g_{\xi\xi}$ defined in (3.33).

Using transformations (6.23) and (6.25) it is easy to rewrite the (6.17) as follows

$$J\frac{\partial\rho}{\partial t}+\frac{\partial}{\partial\xi}(\rho U)+\frac{\partial}{\partial\eta}(\rho V)=0, \tag{6.26a}$$

$$\begin{aligned}&J\frac{\partial\rho Y_i}{\partial t}+\frac{\partial}{\partial\xi}(\rho U Y_i)+\frac{\partial}{\partial\eta}(\rho V Y_i)=\frac{\partial}{\partial\xi}\Big(\frac{\rho D_{im} g_{\eta\eta}}{J}\frac{\partial Y_i}{\partial\xi}\Big)\\&+\frac{\partial}{\partial\eta}\Big(\frac{\rho D_{im} g_{\xi\xi}}{J}\frac{\partial Y_i}{\partial\eta}\Big)+Js_i, \qquad i=1,\cdots,N_s-1,\end{aligned} \tag{6.26b}$$

$$J\frac{\partial(\rho u)}{\partial t}+\frac{\partial}{\partial\xi}(\rho U u)+\frac{\partial}{\partial\eta}(\rho V u)=\frac{\partial}{\partial\xi}(\frac{\partial y}{\partial\eta}p)-\frac{\partial}{\partial\eta}(\frac{\partial y}{\partial\xi}p)+\tau_1+J\rho\mathbf{g}\cdot\mathbf{i}, \tag{6.26c}$$

$$J\frac{\partial(\rho v)}{\partial t}+\frac{\partial}{\partial\xi}(\rho U v)+\frac{\partial}{\partial\eta}(\rho V v)=\frac{\partial}{\partial\eta}(\frac{\partial x}{\partial\xi}p)-\frac{\partial}{\partial\xi}(\frac{\partial x}{\partial\eta}p)+\tau_2+J\rho\mathbf{g}\cdot\mathbf{j}, \tag{6.26d}$$

$$\begin{aligned}&J\frac{\partial\rho h}{\partial t}+\frac{\partial}{\partial\xi}(\rho U h)+\frac{\partial}{\partial\xi}(\rho V h)=\frac{\partial}{\partial\xi}\Big(\frac{\lambda g_{\eta\eta}}{J}\frac{\partial T}{\partial\xi}\Big)+\frac{\partial}{\partial\eta}\Big(\frac{\lambda g_{\xi\xi}}{J}\frac{\partial T}{\partial\eta}\Big)+\\&\frac{\partial}{\partial\xi}\Big(\frac{\rho g_{\eta\eta}}{J}\sum_{i=1}^{N_s}h_i D_{im}\frac{\partial Y_i}{\partial\xi}\Big)+\frac{\partial}{\partial\eta}\Big(\frac{\rho g_{\xi\xi}}{J}\sum_{i=1}^{N_s}h_i D_{im}\frac{\partial Y_i}{\partial\eta}\Big).\end{aligned} \tag{6.26e}$$

To obtain the system above, the momentum equation (6.17c) has been split in its Cartesian components before transformation. The transformed viscous terms, represented by τ_1 and τ_2, respectively, read as follows

$$\tau_1=\frac{\partial}{\partial\xi}\Big(\frac{\partial y}{\partial\eta}\tau_{xx}-\frac{\partial x}{\partial\eta}\tau_{xy}\Big)-\frac{\partial}{\partial\eta}\Big(\frac{\partial y}{\partial\xi}\tau_{xx}-\frac{\partial x}{\partial\xi}\tau_{xy}\Big)=\tau_1^{\xi}+\tau_1^{\eta}, \tag{6.27a}$$

$$\tau_2=\frac{\partial}{\partial\xi}\Big(\frac{\partial y}{\partial\eta}\tau_{yx}-\frac{\partial x}{\partial\eta}\tau_{yy}\Big)-\frac{\partial}{\partial\eta}\Big(\frac{\partial y}{\partial\xi}\tau_{yx}-\frac{\partial x}{\partial\xi}\tau_{yy}\Big)=\tau_2^{\xi}+\tau_2^{\eta}, \tag{6.27b}$$

where the components of the viscous stress tensor are given

$$\tau_{xx}=\mu\Big[2\frac{\partial u}{\partial x}-\frac{2}{3}\Big(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\Big)\Big], \tag{6.28a}$$

$$\tau_{xy}=\tau_{yx}=\mu\Big[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\Big], \tag{6.28b}$$

$$\tau_{yy}=\mu\Big[2\frac{\partial v}{\partial y}-\frac{2}{3}\Big(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\Big)\Big]. \tag{6.28c}$$

6.2.2 Discretisation

System (6.17) has to be discretised both in space and time. For space discretisation we use the finite volume method, see [29]. For any control volume V_c, the conservation law

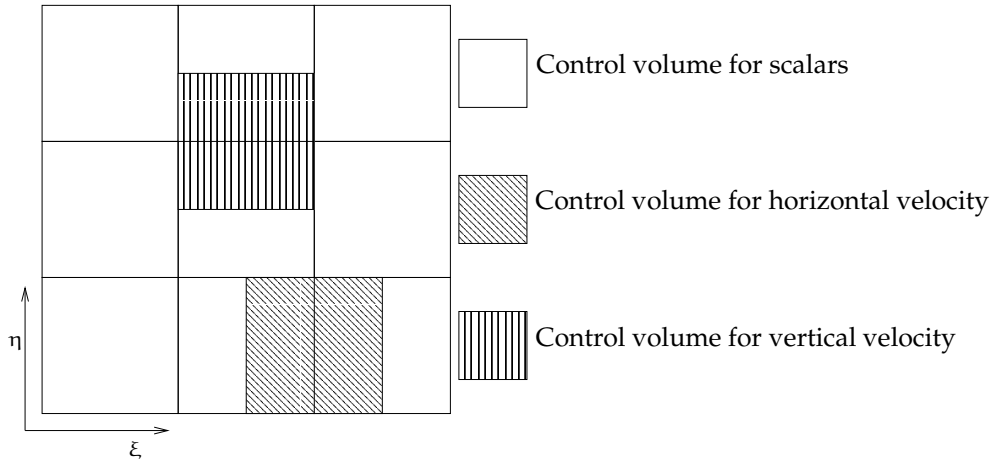


Figure 6.8: A patch of a staggered grid in the computational space.

in integral form reads

$$\int_{V_c} \frac{\partial \varphi}{\partial t}\, dA + \oint_{\partial V_c} \mathbf{f} \cdot \mathbf{n}\, ds - \int_{V_c} s(\varphi)\, dA = 0, \tag{6.29}$$

where φ is the quantity to be conserved, $\mathbf{f}$ the flux term that includes both the convective and the diffusive contributions and $s(\varphi)$ the source term. When the pressure field is computed directly from either system (6.17) or (6.26) there is the possibility that pressure oscillations occur. This issue is addressed in more detail in the next section. Here we only anticipate a possible remedy for this problem, viz. *staggered grids*, where the control volumes for the momentum equations are shifted with respect to the control volumes for the combustion equations, see Figure 6.8. We use a *cell centred* setting, i.e. the point where each variable is computed is located in the centre of the corresponding control volume, see Figure 6.9. As a consequence of this staggering procedure density, temperature and mass fractions are computed at the grid cell centres, while the Cartesian velocity components are computed at the middle of the cell faces of the control volume of a scalar. In this same location, the contravariant velocity components have to be evaluated as well using (6.22). This implies that both u and v have to be available at each cell face. Bilinear interpolation is used to compute u in the centre of the control volume corresponding to v and vice versa.

Conservation law (6.29) has to be approximated in each control volume. Applying the midpoint rule to the integrals, see Figure 6.10, we get

$$\frac{\partial \varphi_C(t)}{\partial t} + \frac{F(\mathbf{x}_e, t) - F(\mathbf{x}_w, t)}{\Delta \xi} + \frac{G(\mathbf{x}_n, t) - G(\mathbf{x}_s, t)}{\Delta \eta} - s(\mathbf{x}_C) = 0, \tag{6.30}$$

with $\varphi_C(t)$ the approximation of $\varphi(\mathbf{x}_C, t)$ and $F(\mathbf{x}_e, t)$ and $F(\mathbf{x}_w, t)$ approximations of the projection of the fluxes $\mathbf{f}(\mathbf{x}_e, t)$ and $\mathbf{f}(\mathbf{x}_w, t)$ along the direction orthogonal to the cell

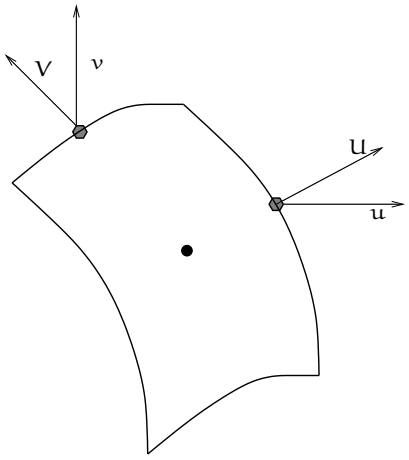


Figure 6.9: Curvilinear cell for a scalar variable in the physical space.

faces in e and w, respectively. Similarly, $G(\mathbf{x}_n, t)$ and $G(\mathbf{x}_s, t)$ represent the approximations of the projection of the fluxes $\mathbf{f}(\mathbf{x}_n, t)$ and $\mathbf{f}(\mathbf{x}_s, t)$ along the direction orthogonal to the cell faces in n and s. Here the neighbouring points of C are labelled N, E, S W. Labels n, e, s and w, instead, refers to the middle points of the northern, eastern, southern and western faces of the considered control volume, respectively.

There are several possibilities to compute the fluxes $F(\mathbf{x}, t)$ and $G(\mathbf{x}, t)$ that appear in (6.30). The simplest choice would be the central difference scheme. However, when convection is dominant, unphysical oscillations can occur. These *convection wiggles* can be avoided using *exponential schemes*, which can be interpreted in terms of a modified diffusion. To compute the numerical fluxes, let us consider the steady-state one-dimensional convection-diffusion model equation

$$\frac{df}{d\xi} = 0, \qquad f(\xi) = m\varphi - D\frac{d\varphi}{d\xi}, \tag{6.31}$$

with m and D constant. The general solution of (6.31) is

$$\varphi(\xi) = C_1 e^{m\xi/D} + C_2. \tag{6.32}$$

Let us define the *Peclet number* as $Pe := m\Delta\xi/D$, with $\Delta\xi = \xi_E - \xi_P$. We note that the solution (6.32) can be interpreted as the central difference numerical solution of

$$\frac{df}{d\xi} = 0, \qquad f(\xi) = m\varphi - D_{new}\frac{d\varphi}{d\xi}, \tag{6.33}$$

with D_{new} defined by

$$D_{new} = D\Big(\frac{Pe}{e^{Pe} - 1} + \frac{Pe}{2}\Big). \tag{6.34}$$

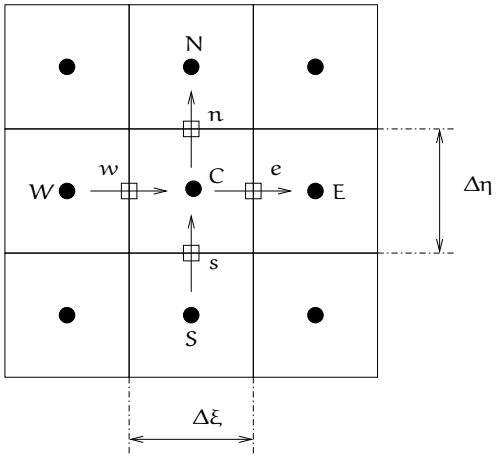


Figure 6.10: Point C: control volume and neighbouring points in the computational space.

The new diffusion coefficient D_{new} adds numerical diffusion for large Peclet numbers to suppress oscillations. This scheme can be extended to two-dimensional problems, see [31].

When applying this *exponential scheme* to the combustion equations in curvilinear coordinates, the Peclet number (6.34) entering e.g. (6.26b) should be replaced by

$$\mathrm{Pe}^{\xi} = \frac{U J \Delta\xi}{D_{im} g_{\eta\eta}}, \qquad \mathrm{Pe}^{\eta} = \frac{V J \Delta\eta}{D_{im} g_{\xi\xi}}. \tag{6.35}$$

In a similar way we evaluate the Peclet numbers for equation (6.26e).

When transforming the flow equations from Cartesian to curvilinear coordinates, special care is needed when discretising the viscous terms, that become slightly more involved. We discuss here the discretisation of the term τ_1^{ξ} only and refer to Appendix 1 for both τ_1^{η} and τ_2. Let us consider the momentum equation (6.26c) at the point e, see Figure 6.10. The discretised τ_1^{ξ} term reads

$$\tau_1^{\xi}\big|_e \doteq \Big(\frac{\partial y}{\partial \eta}\tau_{xx} - \frac{\partial x}{\partial \eta}\tau_{xy}\Big)_E - \Big(\frac{\partial y}{\partial \eta}\tau_{xx} - \frac{\partial x}{\partial \eta}\tau_{xy}\Big)_C. \tag{6.36}$$

It is worth to recall that the grid line spacing in the computational space can be arbitrary, so we set $\Delta\xi = \Delta\eta = 1$. We focus on the computation of the term $(\frac{\partial y}{\partial \eta}\tau_{xx} - \frac{\partial x}{\partial \eta}\tau_{xy})$ at a generic point C, with τ_{xx} and τ_{xy} defined in (6.28a) and (6.28b), respectively. Using finite differences for the gradient transformations given in (6.24), we get

$$\frac{\partial u}{\partial x}\Big|_C \doteq \frac{1}{J_C}\Big(\Big(\frac{\partial y}{\partial \eta}u\Big)_e - \Big(\frac{\partial y}{\partial \eta}u\Big)_w\Big) - \frac{1}{J_C}\Big(\Big(\frac{\partial y}{\partial \xi}u\Big)_n - \Big(\frac{\partial y}{\partial \xi}u\Big)_s\Big), \tag{6.37a}$$

$$\frac{\partial u}{\partial y}\Big|_C \doteq -\frac{1}{J_C}\Big(\Big(\frac{\partial x}{\partial \eta}u\Big)_e - \Big(\frac{\partial y}{\partial \eta}u\Big)_w\Big) + \frac{1}{J_C}\Big(\Big(\frac{\partial x}{\partial \xi}u\Big)_n - \Big(\frac{\partial x}{\partial \xi}u\Big)_s\Big). \tag{6.37b}$$

A similar expression can be used for the v-derivatives. The discretised viscous term eventually becomes

$$\left(\frac{\partial y}{\partial \eta}\tau_{xx} - \frac{\partial x}{\partial \eta}\tau_{xy}\right)\Big|_C \doteq \frac{\mu}{J_C}\left[\frac{2}{3}\left(\frac{\partial y}{\partial \eta}\right)_C\left(\left(2u\frac{\partial y}{\partial \eta} + v\frac{\partial x}{\partial \eta}\right)_e - \left(2u\frac{\partial y}{\partial \eta} + v\frac{\partial x}{\partial \eta}\right)_w\right.\right.$$
$$- \left(2v\frac{\partial y}{\partial \xi} + v\frac{\partial x}{\partial \xi}\right)_n + \left(2v\frac{\partial y}{\partial \xi} + v\frac{\partial x}{\partial \xi}\right)_s\Big) + \left(\frac{\partial x}{\partial \eta}\right)_C\Big(\left(u\frac{\partial x}{\partial \eta} - v\frac{\partial y}{\partial \eta}\right)_e - \left(u\frac{\partial x}{\partial \eta} - v\frac{\partial y}{\partial \eta}\right)_w$$
$$\left.- \left(u\frac{\partial x}{\partial \xi} - v\frac{\partial y}{\partial \xi}\right)_n + \left(u\frac{\partial x}{\partial \xi} - v\frac{\partial y}{\partial \xi}\right)_s\Big)\right]. \tag{6.38}$$

It has to be noted that both the u-component in n and s and the v-component in e and w are not directly available from the computations and have to be interpolated using its four neighbouring values.

Finally, central differences are used to compute the coefficients $\frac{\partial x}{\partial \eta}$, $\frac{\partial x}{\partial \xi}$, $\frac{\partial y}{\partial \eta}$ and $\frac{\partial y}{\partial \xi}$ entering both the flow and the combustion equations.

Once our system of equations has been discretised in space, numerical integration has to be carried out. The discretised system can be written as

$$\frac{d\psi}{dt} = \mathbf{F}(\psi), \tag{6.39}$$

where ψ is the vector of all the unknowns. Since chemical reactions sometimes occur at extremely small time scales, combustion problems are usually very stiff. Thus, if explicit integration methods are to be used, the time step has to be chosen commensurate with the smaller physical time scale. In order to avoid this, the backward Euler scheme is used

$$\psi^n = \psi^{n-1} + \Delta t\mathbf{F}(\psi^n), \tag{6.40}$$

where Δt is the time step.

6.2.3 Solution strategy

The solution of (6.40) is usually very difficult to obtain, also due to the isobaric nature of the combustion process. Pressure correction methods allow for decoupling the computation of the flow and combustion variables from the pressure. The scheme used here is similar to the one presented in [33] and extensively studied in [30]. We derive first the pressure correction algorithm for Cartesian coordinates, then we discuss the modifications needed to formulate it in curvilinear coordinates.

We present the pressure correction algorithm by following the transverse method of lines, i.e. we use continuous space derivatives such that, at a later stage, either Cartesian or curvilinear coordinates can be used. We start by writing system (6.17) in the following

slightly different form

$$\frac{\partial \rho}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho \mathbf{v}), \tag{6.41a}$$

$$\rho\frac{\partial Y_i}{\partial t} = Y_i\big[\boldsymbol{\nabla}\cdot(\rho\mathbf{v})\big] - \boldsymbol{\nabla}\cdot(\rho\mathbf{v}Y_i) + \boldsymbol{\nabla}\cdot(\rho D_{im}\boldsymbol{\nabla}Y_i) + s_i, \quad i = 1,\cdots,N_s-1, \tag{6.41b}$$

$$\rho\frac{\partial \mathbf{v}}{\partial t} = \mathbf{v}\big[\boldsymbol{\nabla}\cdot(\rho\mathbf{v})\big] - \boldsymbol{\nabla}\cdot(\rho\mathbf{v}\otimes\mathbf{v}) - \boldsymbol{\nabla}p - \boldsymbol{\nabla}\cdot\tau + \rho\mathbf{g}, \tag{6.41c}$$

$$\rho\frac{\partial h}{\partial t} = h\big[\boldsymbol{\nabla}\cdot(\rho\mathbf{v})\big] - \boldsymbol{\nabla}\cdot(\rho\mathbf{v}h) + \boldsymbol{\nabla}\cdot(\lambda\nabla T) + \boldsymbol{\nabla}\cdot\left(\rho\sum_{i=1}^{N_s} h_i D_{im}\boldsymbol{\nabla}Y_i\right). \tag{6.41d}$$

Let us introduce the vector $\boldsymbol{\varphi} := [Y_1,\dots Y_{N_s-1}, h]^T$ of the combustion variables and express the density as a function of $\boldsymbol{\varphi}$, i.e. $\rho = Q(\boldsymbol{\varphi})$, via the equation of state (6.18) and the relations for h (6.19). Then, equation (6.41a) can be rewritten as

$$\frac{\partial Q}{\partial t} = \sum_i \frac{\partial Q}{\partial \varphi_i}\frac{\partial \varphi_i}{\partial t} = -\boldsymbol{\nabla}\cdot(\rho\mathbf{v}), \tag{6.42}$$

where

$$\frac{\partial Q}{\partial \boldsymbol{\varphi}} = -\rho\Big[\cdots, \bar{M}\big(\frac{1}{M_j} - \frac{1}{M_{N_s}}\big) - \frac{1}{c_p T}(h_j - h_{N_s}), \cdots, \frac{1}{c_p T}\Big]^T, \tag{6.43}$$

with $\bar{M}$ the average molecular mass. Furthermore, the combustion equations (6.41b) and (6.41d) and the flow equation (6.41c) can be symbolically written as

$$\rho\frac{\partial \boldsymbol{\varphi}}{\partial t} = \boldsymbol{\varphi}\big[\boldsymbol{\nabla}\cdot(\rho\mathbf{v})\big] - \boldsymbol{\nabla}\cdot\big(\rho\mathbf{v}\boldsymbol{\varphi}\big) + \mathcal{A}(\boldsymbol{\varphi}), \tag{6.44a}$$

$$\rho\frac{\partial \mathbf{v}}{\partial t} = \mathcal{B}_1(\boldsymbol{\varphi},\mathbf{v}) - \boldsymbol{\nabla}p. \tag{6.44b}$$

The continuity equation, with the help of (6.44a), can be rewritten as

$$-\boldsymbol{\nabla}\cdot(\rho\mathbf{v}) - \sum_i \frac{\partial Q}{\partial \varphi_i}\Big(\frac{\varphi_i}{\rho}\boldsymbol{\nabla}\cdot(\rho\mathbf{v}) - \frac{1}{\rho}\boldsymbol{\nabla}\cdot(\rho\mathbf{v}\varphi_i)\Big) = \sum_i \frac{\partial Q}{\partial \varphi_i}\mathcal{A}_i(\boldsymbol{\varphi}). \tag{6.45}$$

Using the identity

$$\sum_{i=1}^{N_s-1}\frac{\partial Q}{\partial Y_i}Y_i = -\rho\left(1 - \frac{\bar{M}}{M_{N_s}}\right) + \frac{\rho}{c_p T}(h - h_{N_s}), \tag{6.46}$$

equation (6.45) can be rearranged as

$$\boldsymbol{\nabla}\cdot(\rho\mathbf{v}) + \sum_i \alpha_i\boldsymbol{\nabla}\cdot(\rho\mathbf{v}\varphi_i) = \sum_i \alpha_i\mathcal{A}_i(\boldsymbol{\varphi}), \tag{6.47}$$

with the coefficient α_i given by

$$\alpha_i = \frac{1}{\rho}\frac{\partial Q}{\partial \varphi_i}\left(\frac{\bar{M}}{M_{N_S}} - \frac{h_{N_S}}{c_p T}\right)^{-1}. \tag{6.48}$$

Finally, equation (6.47) can be formally written as

$$\mathcal{P}_1(\boldsymbol{\varphi})\mathbf{v} = \mathcal{P}_2(\boldsymbol{\varphi}). \tag{6.49}$$

The pressure correction scheme essentially consists of the following steps. At each time step, predictor values for the velocity are computed first, then they are corrected such that the *constraint equation* (6.49) is satisfied. The system then reads

$$\mathcal{P}_1(\boldsymbol{\varphi}^n)\mathbf{v}^n = \mathcal{P}_2(\boldsymbol{\varphi}^n), \tag{6.50a}$$

$$\rho^n \frac{\boldsymbol{\varphi}^n - \boldsymbol{\varphi}^{n-1}}{\Delta t} = \boldsymbol{\varphi}^n \left[\boldsymbol{\nabla}\cdot(\rho^n \mathbf{v}^*)\right] - \boldsymbol{\nabla}\cdot(\rho \mathbf{v}^* \boldsymbol{\varphi}^n) + \mathcal{A}(\boldsymbol{\varphi}^n), \tag{6.50b}$$

$$\rho^n \frac{\mathbf{v}^* - \mathbf{v}^{n-1}}{\Delta t} = \mathcal{B}_1(\boldsymbol{\varphi}^n, \mathbf{v}^*) - \boldsymbol{\nabla} p^{n-1}, \tag{6.50c}$$

$$\rho^n \frac{\mathbf{v}^n - \mathbf{v}^{n-1}}{\Delta t} = \mathcal{B}_1(\boldsymbol{\varphi}^n, \mathbf{v}^*) - \boldsymbol{\nabla} p^n, \tag{6.50d}$$

where $\mathbf{v}^*$ is the predicted velocity field and the superscript n refers to the quantities computed at the time t^n.

It is useful to remark that system (6.50) presents a peculiarity in that it uses the *constraint equation* (6.50a) instead of the usual continuity equation. The reason for this is explained in [30] and is related to instabilities that can occur when (6.41a) is used instead of (6.50a). We should emphasize that system (6.50) is much easier to solve than (6.40) since the pressure is decoupled from the other variables.

Subtracting (6.50c) from (6.50d), we get

$$\frac{\rho^n}{\Delta t}(\mathbf{v}^n - \mathbf{v}^*) = -\boldsymbol{\nabla} q, \tag{6.51}$$

with $q = p^n - p^{n-1}$. From this follows

$$\mathbf{v}^n = \mathbf{v}^* - \frac{\Delta t}{\rho^n}\boldsymbol{\nabla} q. \tag{6.52}$$

Combining (6.50a) and (6.52), we get the Poisson equation for pressure update

$$\frac{\Delta t}{\rho^n}\mathcal{P}_1(\boldsymbol{\varphi}^n)(\boldsymbol{\nabla} p^n - \boldsymbol{\nabla} p^{n-1}) = -\mathcal{P}_2(\boldsymbol{\varphi}^n) + \mathcal{P}_1(\boldsymbol{\varphi}^n)\mathbf{v}^*. \tag{6.53}$$

The final algorithm consists of the following steps

- Compute $\boldsymbol{\varphi}^n$ and $\mathbf{v}^*$ from (6.50b) and (6.50c);
- Compute p^n from (6.53);
- Update $\mathbf{v}^n$ using (6.52).

Let us now go one step further and consider curvilinear coordinates. First, we reformulate the constraint equation, then we obtain the new Poisson equation. In this case the (6.47) transforms into

$$\frac{1}{J}\Big[\frac{\partial}{\partial\xi}(\rho\mathbf{U})+\frac{\partial}{\partial\eta}(\rho\mathbf{V})\Big]+\frac{1}{J}\sum_i\alpha_i\Big[\frac{\partial}{\partial\xi}(\rho\mathbf{U}\varphi_i)+\frac{\partial}{\partial\eta}(\rho\mathbf{V}\varphi_i)\Big]=\mathcal{P}_2(\boldsymbol{\varphi}). \tag{6.54}$$

Omitting the superscript n, the discretisation of (6.54) yields

$$\begin{aligned}&\Big[-1+\sum_i\alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_e)\Big](\rho U)_e-\Big[-1+\sum_i\alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_w)\Big](\rho U)_w+\\&\Big[-1+\sum_i\alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_n)\Big](\rho V)_n-\Big[-1+\sum_i\alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_s)\Big](\rho V)_s=J_C\mathcal{P}_2(\boldsymbol{\varphi}).\end{aligned} \tag{6.55}$$

Taking the inner product subsequently by $\mathbf{a}^\xi$ and $\mathbf{a}^\eta$, (6.51) becomes

$$\frac{\rho^n}{\Delta t}(U^n-U^*)=-\mathbf{a}^\xi\cdot\boldsymbol{\nabla}q, \tag{6.56a}$$

$$\frac{\rho^n}{\Delta t}(V^n-V^*)=-\mathbf{a}^\eta\cdot\boldsymbol{\nabla}q, \tag{6.56b}$$

while $\boldsymbol{\nabla}q$ in curvilinear coordinates transforms according to (6.24).

Let us consider the right-hand side of (6.56a). We find

$$\mathbf{a}^\xi\cdot\boldsymbol{\nabla}q=\frac{1}{J}\Big(\mathbf{a}^\xi\cdot\mathbf{a}^\xi\frac{\partial q}{\partial\xi}+\mathbf{a}^\xi\cdot\frac{\partial\mathbf{a}^\xi}{\partial\xi}q+\mathbf{a}^\xi\cdot\mathbf{a}^\eta\frac{\partial q}{\partial\eta}+\mathbf{a}^\xi\cdot\frac{\partial\mathbf{a}^\eta}{\partial\eta}q\Big), \tag{6.57}$$

where the four terms in the right-hand side of (6.57) read

$$\mathbf{a}^\xi\cdot\mathbf{a}^\xi=\Big(\frac{\partial x}{\partial\eta}\Big)^2+\Big(\frac{\partial y}{\partial\eta}\Big)^2=g_{\eta\eta}, \tag{6.58a}$$

$$\mathbf{a}^\xi\cdot\frac{\partial\mathbf{a}^\xi}{\partial\xi}=\frac{\partial y}{\partial\eta}\frac{\partial^2y}{\partial\eta\partial\xi}+\frac{\partial x}{\partial\eta}\frac{\partial^2x}{\partial\eta\partial\xi}, \tag{6.58b}$$

$$\mathbf{a}^\xi\cdot\mathbf{a}^\eta=0, \tag{6.58c}$$

$$\mathbf{a}^\xi\cdot\frac{\partial\mathbf{a}^\eta}{\partial\eta}=-\frac{\partial y}{\partial\eta}\frac{\partial^2y}{\partial\xi\partial\eta}-\frac{\partial x}{\partial\eta}\frac{\partial^2x}{\partial\xi\partial\eta}. \tag{6.58d}$$

Using the relations above we obtain for the right-hand side of (6.57) the final form

$$\mathbf{a}^\xi\cdot\boldsymbol{\nabla}q=\frac{1}{J}g_{\eta\eta}\frac{\partial q}{\partial\xi}. \tag{6.59}$$

The same way we can show that

$$\mathbf{a}^\eta\cdot\boldsymbol{\nabla}q=\frac{1}{J}g_{\xi\xi}\frac{\partial q}{\partial\eta}. \tag{6.60}$$

Finally, the relations between the contravariant velocities and the pressure gradients become

$$\frac{\rho^n}{\Delta t}(U^n - U^*) = -\frac{1}{J} g_{\eta\eta} \frac{\partial q}{\partial \xi}, \tag{6.61a}$$

$$\frac{\rho^n}{\Delta t}(V^n - V^*) = -\frac{1}{J} g_{\xi\xi} \frac{\partial q}{\partial \eta}. \tag{6.61b}$$

To obtain the Poisson equation, we introduce (6.61) into (6.55), getting

$$\begin{aligned}
&\Big[-1 + \sum_i \alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_e)\Big]\Delta t\Big(\frac{1}{J} g_{\eta\eta}\Big)_e (q_E - q_C) - \\
&\Big[-1 + \sum_i \alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_w)\Big]\Delta t\Big(\frac{1}{J} g_{\eta\eta}\Big)_w (q_C - q_W) + \\
&\Big[-1 + \sum_i \alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_n)\Big]\Delta t\Big(\frac{1}{J} g_{\xi\xi}\Big)_n (q_N - q_C) - \\
&\Big[-1 + \sum_i \alpha_i(\boldsymbol{\varphi}_C)\varphi_i(\mathbf{x}_s)\Big]\Delta t\Big(\frac{1}{J} g_{\xi\xi}\Big)_s (q_C - q_S) = -J_C \mathcal{P}_2(\boldsymbol{\varphi}^n) + \mathcal{P}_1(\boldsymbol{\varphi}^n)\mathbf{U}^*.
\end{aligned} \tag{6.62}$$

In Appendix 2 we give details on the algorithm used to solve the combustion and the flow equations together with the pressure correction method.

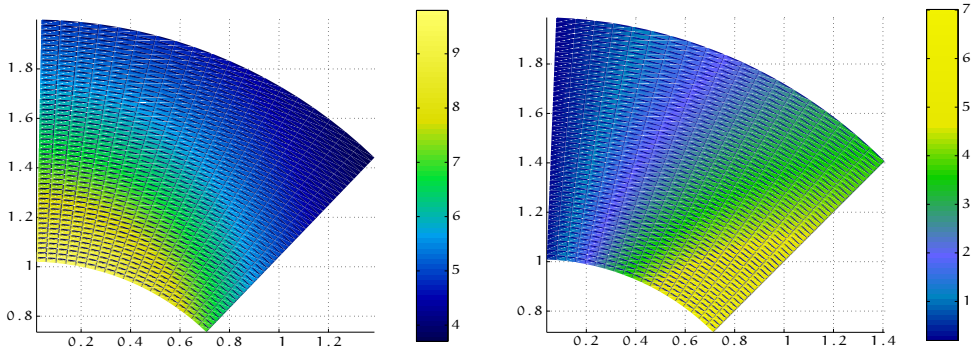
6.2.4 Numerical experiments and pressure oscillations

Some numerical experiments are presented here to test the model introduced so far. In this section, we only consider the Euler equations, i.e. (6.17a) and (6.17c) where in the stress tensor $\mathbf{P}$ the contribution of the viscous terms are omitted.

Example 1
We consider one eighth of a circle with inflow at the inner radius, outflow at the outer radius and slip conditions at the other two sides, see Figure 6.11. The density is assumed to be constant and the inflow velocity is orthogonal to the boundary and equal to 10. The grid is composed of 20 radial lines and 20 circular lines. As initial guess we assume $\mathbf{u} = 0$ in the whole domain. Figure 6.11 shows the converged solution. As expected, the velocity decreases from 10 at the inflow to 5 at the outflow while the pressure increases monotonically.

Example 2
The behaviour of the solution changes if we consider a channel with similar boundary conditions, whose angular coordinate ranges from 0 to $\pi/2$. As soon as the iterative solver starts, pressure oscillations appear in the cells adjacent to the x-axis, see Figure 6.12. Nevertheless, after a few iterations, oscillations are damped away and the solution evolves towards the steady state shown in Figure 6.13. In this case the inflow velocity is set equal to 1.

Let us consider the so-called *checkerboard* or *even-odd decoupling* problem. Pressure oscillations can occur when the pressure gradient is discretised with second order central



(a) Vertical velocity. (b) Horizontal velocity.

Figure 6.11: Cartesian velocities for Example 1.

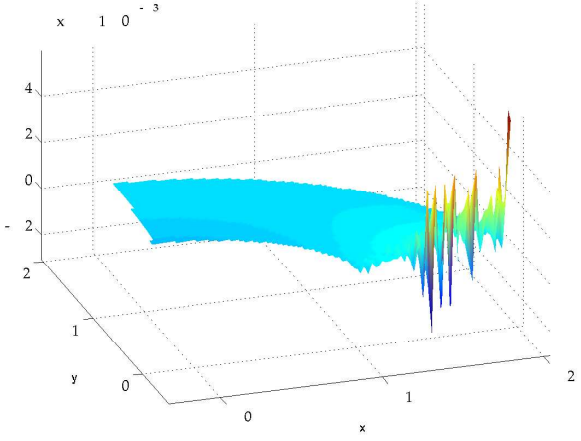
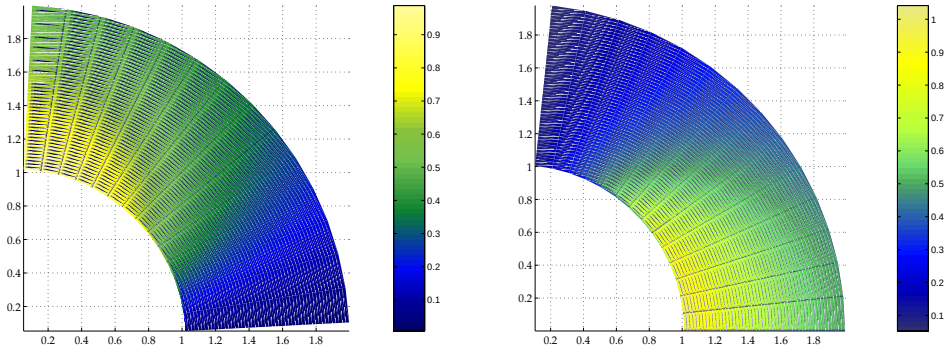


Figure 6.12: Pressure field.



(a) Vertical velocity. (b) Horizontal velocity.

Figure 6.13: Cartesian velocities for Example 2.

differences on collocated Cartesian grids. The same problem arises if a cell centred finite volume discretisation is used and fluxes are computed using central differences. In these cases, it can be shown that if $p_{i,j}$ is the pressure approximation at the grid point labelled (i, j), also

$$\tilde{p}_{i,j} = p_{i,j} + \beta p^{ch}, \qquad \text{with,} \tag{6.63a}$$

$$p^{ch} = 1, (-1)^i, (-1)^j \text{or} (-1)^{i+j}, \tag{6.63b}$$

satisfy the pressure field equations, see e.g. [30]. The solutions in (6.63b) are called *spurious modes* and belong to the null space of the gradient operator. There are several ways to avoid spurious modes. As introduced in the previous section, the use of *staggered grids* is a common practice to achieve this.

Let us now consider the curvilinear BVP. In Example 2 we see that, although the flow equations are discretised in a staggered fashion, spurious modes again appear in a small part of the computational domain. This is caused by the 90 degrees turning of the grid with respect to its original orientation. To understand what happens, we look at Figure 6.14. At position A, velocities are driven by the difference of the pressure in the cell centres. In this case staggering is effective. When the coordinates lines are turned 90 degrees, i.e. at position B, velocities are driven by the average of the pressure taken in the cell corners. We must then consider what happens to the momentum equations, and, in particular, to the pressure gradients. In the following we assume that i and j increase together with η and ξ, respectively. We consider the term $\partial p/\partial x$. After transformation from Cartesian to (ξ, η)-coordinates, we have

$$\frac{\partial p}{\partial x} \doteq \frac{1}{J}\left(\frac{\partial}{\partial \eta}\left(\frac{\partial y}{\partial \xi} p\right) - \frac{\partial}{\partial \xi}\left(\frac{\partial y}{\partial \eta} p\right)\right). \tag{6.64}$$

This term has to be computed in the point where u is stored. Let us consider the contri-

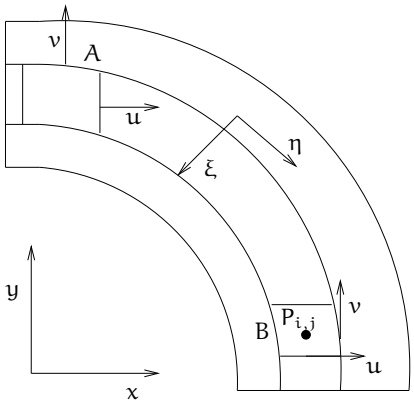


Figure 6.14: Change in the grid orientation.

bution given by $\partial(\frac{\partial y}{\partial \eta}p)/\partial \xi$, in (6.64). If we suppose that $\frac{\partial y}{\partial \eta}$ is constant, we have

$$\begin{aligned}\left(\frac{\partial p}{\partial \xi}\right)_{i+\frac{1}{2},j} &= \left(p_{i,j} + p_{i+1,j} + p_{i,j+1} + p_{i+1,j+1}\right) - \left(p_{i,j} + p_{i-1,j} + p_{i,j+1} + p_{i-1,j+1}\right) \\ &= p_{i+1,j} + p_{i+1,j+1} - p_{i-1,j} - p_{i-1,j+1}.\end{aligned} \tag{6.65}$$

We notice that the right-hand side of (6.65) does not contain any contribution of the pressure in i. Therefore, if also $\frac{\partial y}{\partial \xi} = 0$, i.e. if the η-lines are parallel to the x-axis, decoupling in the i-direction occurs and it is not possible to avoid the spurious mode to enter the solution. A similar argument holds for the momentum equation in the y-direction.

Strictly speaking, spurious modes should only occur in the very few cases determined by the conditions stated so far. For the momentum equation in x-direction they require that $\frac{\partial y}{\partial \eta}$ is constant and $\frac{\partial y}{\partial \xi} = 0$. This is also what is claimed, for instance, in [53]. Nevertheless, the grid of Example 2 does not belong to this category. Such behaviour of the pressure field has already been observed by other authors, see e.g. [49] in a finite element context. Even-odd decoupling occurs even if the metric coefficients appearing in (6.64) do not comply exactly with the conditions that we have introduced. According to the interpretation given in [49], a grid that does not support spurious pressure modes can be seen as a (small) perturbation of a grid that does support them.

6.2.5 Bunsen flame simulation

Let us now present the results of our Bunsen flame simulation. First consider the global problem. The burner geometry is shown in Figure 6.15. Boundary conditions are set as follows. At the inflow we assign velocity, temperature and composition of the gaseous

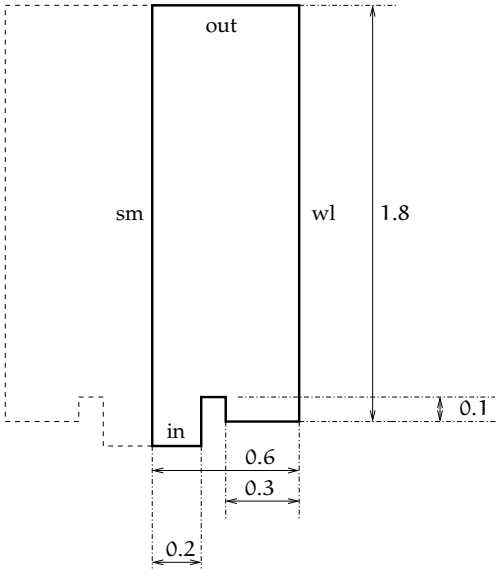


Figure 6.15: Global Bunsen burner domain. Dimensions are in cm. Since the burner is symmetric, only its right half is considered for computations. The inflow, outflow, wall and symmetry boundaries are marked with 'in', 'out', 'wl' and 'sm', respectively.

mixture

$$v = 125, \qquad u = 0, \qquad T = 333, \tag{6.66a}$$

$$Y_j = Y_{0j}, \qquad j = 1 \cdots N_{s-1}, \tag{6.66b}$$

where the inflow mass fractions corresponds to a methane/air mixture with stoichiometric ratio equal to 0.8. At the walls, instead, we have

$$v = u = 0, \qquad T = 333, \tag{6.67a}$$

$$\frac{\partial Y_j}{\partial \mathbf{n}} = 0, \qquad j = 1 \cdots N_{s-1}, \tag{6.67b}$$

where $\mathbf{n}$ is the outward unit vector normal to the boundary. Boundary conditions at the symmetry axis read,

$$\frac{\partial v}{\partial x} = u = \frac{\partial T}{\partial x} = 0, \tag{6.68a}$$

$$\frac{\partial Y_j}{\partial x} = 0, \qquad j = 1 \cdots N_{s-1}. \tag{6.68b}$$

Finally, at the outflow we have

$$\frac{\partial v}{\partial y} = u = \frac{\partial T}{\partial y} = 0, \tag{6.69a}$$

$$\frac{\partial Y_j}{\partial y} = 0, \qquad j = 1 \cdots N_{s-1}. \tag{6.69b}$$

Pressure boundary conditions need to be specified as well. From (6.51), we see that if the normal component of the velocity is given, at the boundary the condition

$$\frac{\partial q}{\partial \mathbf{n}} = 0, \tag{6.70}$$

holds. However, a Dirichlet condition has to be imposed at least in one point to avoid the Poisson equation to become singular. This can be done where no velocity is given, i.e. at the outflow boundary. There, we impose the condition

$$q = 0, \tag{6.71}$$

whereas at inflow walls and at the symmetry axis condition (6.70) holds. The reason for (6.71) is that, because of the isobaric approximation, pressure is determined up to a constant. This implies that differences, rather than pressure values, are important and thus an arbitrary pressure level can be set in (6.71).

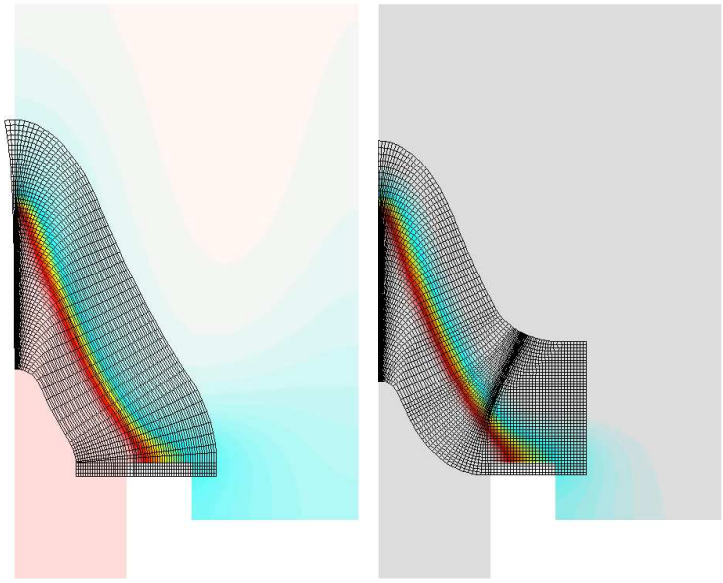
We now introduce the local problem to refine the flame front. Two types of grid are used. The first one is shown in Figure 6.16(a), the second one in Figure 6.16(b), where they are plotted over the flame front. From now on we refer to them as GRD1 and GRD2, respectively. Both grids are built by using the procedure explained in Chapter 3 and choosing the methane level curves as set of coordinate lines to be retained. The width of the grid is determined with the aim to cover completely the flame front and to avoid unacceptably small grid cells. The number of grid points is 3192 for GRD1 and 4941 for GRD2. The bigger grid size along the level curves is chosen approximately equal to the grid size of the finest coarse level. Where the grid lines deviate from the solution iso-lines, the grid size is conveniently reduced. The grid size in the direction orthogonal to the level curves is approximately one fourth of finest coarse level. For both grids, boundary conditions (6.68) are applied at the symmetry axis and (6.67) at the boundary of the local domain that coincides with the lower wall. At the interface points, bilinear interpolation is used to map values from the coarse to the fine grid. The coarse grid solution is computed with the multigrid method outlined in Appendix 2. We use four refinement levels covering the global domain. The finest coarse level consists of 6998 grid points.

When we try to solve the local problem on GRD1, pressure oscillations occur in the lower part of the local domain. In this case they do not damp as the computation proceeds. Therefore, we get rid of the checkerboard oscillation by post-processing the computed pressure with a simple averaging procedure. For each point C, with reference to Figure 6.10, the smoothed pressure $\hat{p}_C$ is given by

$$\hat{p}_C = \frac{p_C}{2} + \frac{p_S + p_N + p_W + p_E}{8}. \tag{6.72}$$

This way, we manage to damp completely the checkerboard oscillations in the areas where they occur; in other parts of the local domain this results in an additional smoothing of the pressure field.

In an attempt to completely avoid even-odd decoupling, we have used GRD2. In fact, in this case no change occurs in control volumes orientation. The use of GRD2 implies one



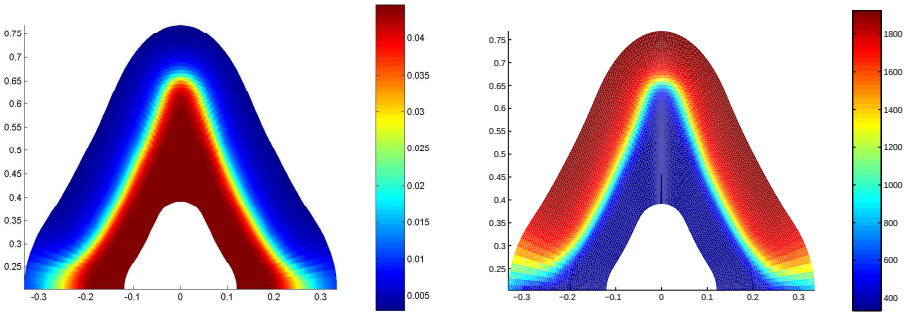
(a) GRD1. (b) GRD2.

Figure 6.16: Local fine grids.

drawback. In order to avoid the 90 degrees turning of the grid cells, the grid lines have to deviate from the methane iso-curves. Together with this requirement, the necessity to have a smooth grid pattern implies a quite bigger number of cells with respect to GRD1.

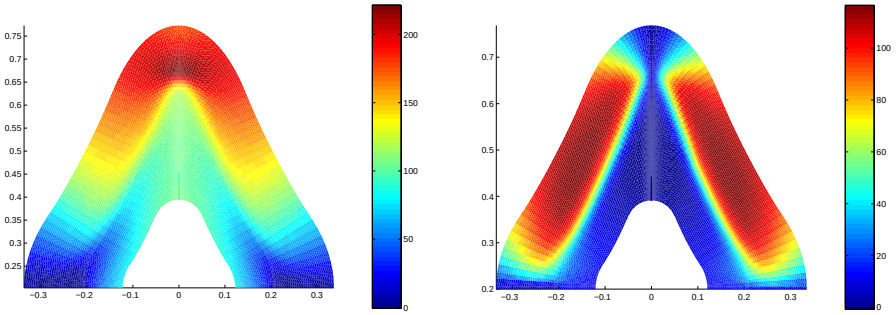
The results for the fine grid computations are shown in Figures 6.17-6.19 for GRD1 and in Figures 6.20-6.22 for GRD2.

Finally, these fine grid solutions can be used to compute the defect needed to update the coarse grid solution, applying a procedure similar to the one used for the thermo-diffusive model.



(a) Methane mass fraction. (b) Temperature.

Figure 6.17: GRD1. Temperature and methane mass fraction.



(a) Vertical velocity. (b) Horizontal velocity.

Figure 6.18: GRD1. Velocities.

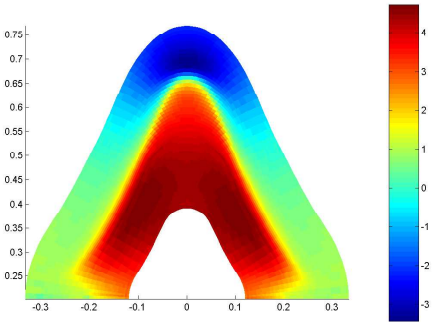
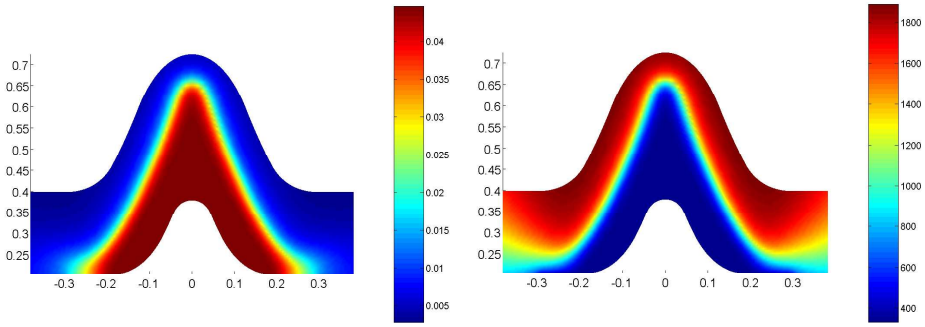
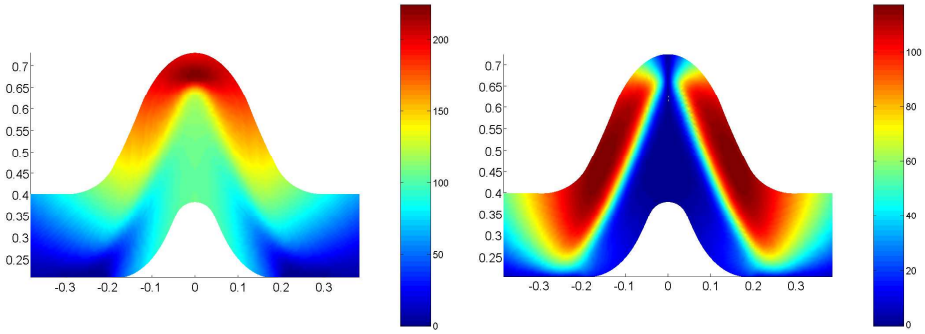


Figure 6.19: GRD1. Pressure.



(a) Methane mass fraction. (b) Temperature.

Figure 6.20: GRD2. Temperature and methane mass fraction.



(a) Vertical velocity. (b) Horizontal velocity.

Figure 6.21: GRD2. Velocities.

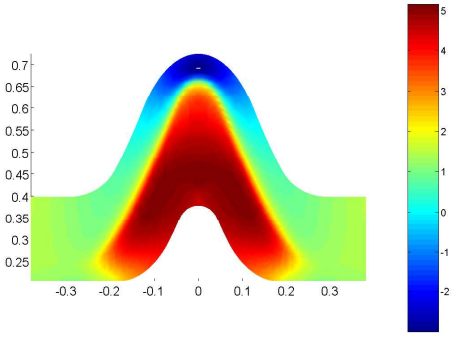


Figure 6.22: GRD2. Pressure.

CHAPTER 7

Conclusions and recommendations

In this thesis we have studied the behaviour of the Local Defect Correction method in combination with different grid types. The standard LDC method is meant to deal with boundary value problems that present high gradients in limited parts of the global domain. It constitutes a powerful tool that, while having the advantage of handling simple data structures, allows to solve complex problem limiting the memory requirements. When LDC is combined with curvilinear grids, the number of grid points used to solve the local problem can be further reduced. This is achieved by using solution-fitted fine grids that tightly follow the shape of the solution. Moreover, we require the fine grids to be orthogonal in order to reduce the complexity of the equations after transformation from the Cartesian to the curvilinear coordinates and to increase the accuracy of the difference approximation. A number of model problems has been used to assess the properties of the method. We have shown that LDC can be used in combination with both rectangular grids that are slanted with respect to the global domain and grids in general coordinates. By performing a complexity asymptotic analysis we have seen that LDC turns out to be very competitive with respect to tensor product grid methods, especially when a set of grid lines is aligned with the solution iso-contours. Moreover, if we consider solution-fitted fine grids and split the complexity of the method in the sum of the global coarse and the local fine problem contributions, we see that the former represents the leading term of the sum.

An important step in this method is the generation of a solution-fitted local grid. We have chosen an algorithm that is both simple to implement and not expensive. Although we have never experienced such kinds of problems, the method does not guarantee that when the lines orthogonal to the level curves are generated, they do not cross each other in regions of high convexity where the grid become denser in points. The use of more robust grid generators can allow for improved performance of LDC in solving BVP that show high activity areas with complex shape.

The LDC method based on the use of several refinement levels has already been suc-

cessfully applied, see [2], combining patches of rectangular grids. On the contrary, in our approach we have used a single refinement level. In fact, since the fine grid is generated following the shape of the solution, the line set aligned with the iso-contours can be made less dense as we get far from the high activity region, in areas where the solution is less steep. Although the developed technique has demonstrated to be effective, a possible evolution is to use the LDC method by combining several grid types at different levels.

The LDC method in combination with orthogonal curvilinear grids has then been used to solve combustion problems. In the first application we have studied the thermo-diffusive model. Although less involved with respect to other combustion models, it allows to get a good insight in the flame structure and in the behaviour of the front in some peculiar flow conditions. The second combustion problem that has been tackled is the simulation of a Bunsen flame. This has been solved using an already existing laminar flame simulation code that has been extended in order to solve the combustion equations in general coordinates. Therefore, some of the features of the numerical methods employed have been determined on the basis of the existing background. The flow equations have been translated from to curvilinear coordinates, as well as the Poisson equation for the pressure. In particular, the latter requires special attention. In fact, we have seen that the staggered grids, used to avoid pressure wiggles, lose their effectiveness when used in combination with certain curvilinear grid configurations. This phenomenon has shown to be more persistent with respect to what is usually outlined in literature, even when filters purposely designed are applied. The causes and remedies for this problem are certainly worth to be further investigated.

A part of this research has been devoted to the study of solution methods for nonlinear systems. The interest for this subject arises from the difficulty in computing the solution of systems obtained from the discretisation of the equations modeling combustion. Nevertheless, they are completely general and applicable to a large variety of problems. The algorithms developed in this thesis have been tested with several benchmark problems, showing good convergence properties even when starting with initial guesses that are far from the solution of the system. The obtained results encourage further research in view of the application to more complex problems.

APPENDIX **A**

Discretisation of the viscous terms

Here we present the discretisation of the viscous terms (6.27). The approximation of τ_1^ξ has already been introduced in Section 6.2.2. Now we look at the terms τ_1^η, τ_2^ξ and τ_2^η. Let us start with τ_1^η. With reference to Figure A.1, we have

$$\tau_1^\eta\big|_e \doteq \Big(-\frac{\partial y}{\partial \xi}\tau_{xx} + \frac{\partial x}{\partial \xi}\tau_{xy}\Big)_{hn} - \Big(-\frac{\partial y}{\partial \xi}\tau_{xx} + \frac{\partial x}{\partial \xi}\tau_{xy}\Big)_{hs}. \tag{A.1}$$

From (A.1) we see that now both the velocities and the metric coefficients have to be evaluated in the upright corner of the control volume centred about C. The following holds

$$\begin{aligned}\Big(\frac{\partial x}{\partial \xi}\tau_{xy} - \frac{\partial y}{\partial \xi}\tau_{xx}\Big)_{hs} \doteq \frac{1}{J_{hn}}\Big[\Big(\frac{\partial x}{\partial \xi}\Big)_{hn}\Big(\Big(-\frac{\partial x}{\partial \eta}u + \frac{\partial y}{\partial \eta}v\Big)_{nE} - \Big(-\frac{\partial x}{\partial \eta}u + \frac{\partial y}{\partial \eta}v\Big)_{n}\\ + \Big(\frac{\partial x}{\partial \xi}u - \frac{\partial y}{\partial \xi}v\Big)_{eN} - \Big(\frac{\partial x}{\partial \xi}u - \frac{\partial y}{\partial \xi}v\Big)_{e}\Big) - \frac{2}{3}\Big(\frac{\partial y}{\partial \xi}\Big)_{hn}\Big(\Big(2\frac{\partial y}{\partial \eta}u + \frac{\partial x}{\partial \eta}v\Big)_{nE}\\ - \Big(2\frac{\partial y}{\partial \eta}u + \frac{\partial x}{\partial \eta}v\Big)_{n} + \Big(2\frac{\partial y}{\partial \xi}u + \frac{\partial x}{\partial \eta}v\Big)_{eN} - \Big(2\frac{\partial y}{\partial \xi}u + \frac{\partial x}{\partial \eta}v)_{e}\Big)\Big].\end{aligned} \tag{A.2}$$

We consider now the terms τ_2^η and τ_2^ξ. It is worth to recall that they refer to the momentum equation for v, so the considered control volume is centred about the point n. Then

$$\tau_2^\eta\big|_n \doteq \Big(\frac{\partial x}{\partial \xi}\tau_{yy} - \frac{\partial y}{\partial \xi}\tau_{yx}\Big)_N - \Big(\frac{\partial x}{\partial \xi}\tau_{yy} - \frac{\partial y}{\partial \xi}\tau_{yx}\Big)_C, \tag{A.3}$$

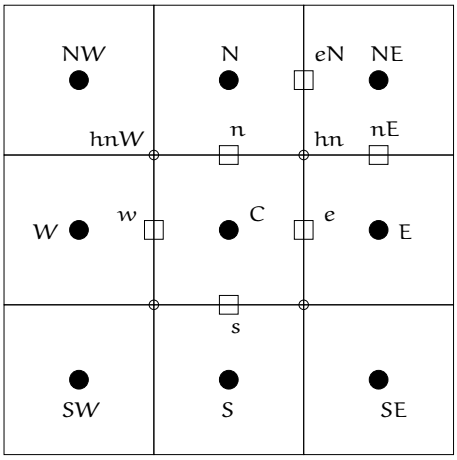


Figure A.1: Control points used for the computation of the viscous terms.

and

$$\begin{aligned}\Big(\frac{\partial x}{\partial \xi}\tau_{yy}-\frac{\partial y}{\partial \xi}\tau_{yx}\Big)_C \doteq \frac{1}{J_C}\Big[\frac{2}{3}\Big(\frac{\partial x}{\partial \xi}\Big)_C\Big(\Big(2\frac{\partial x}{\partial \eta}v+\frac{\partial y}{\partial \eta}u\Big)_e-\Big(2\frac{\partial x}{\partial \eta}v+\frac{\partial y}{\partial \eta}u\Big)_w\\ -\Big(2\frac{\partial x}{\partial \xi}v+\frac{\partial y}{\partial \xi}u\Big)_n+\Big(2\frac{\partial x}{\partial \xi}v+\frac{\partial y}{\partial \xi}u\Big)_s\Big)-\Big(\frac{\partial y}{\partial \xi}\Big)_C\Big(\Big(-\frac{\partial x}{\partial \eta}u+\frac{\partial y}{\partial \eta}v\Big)_e\\ -\Big(-\frac{\partial x}{\partial \eta}u+\frac{\partial y}{\partial \eta}v\Big)_w+\Big(\frac{\partial x}{\partial \xi}u-\frac{\partial x}{\partial \xi}v\Big)_n-\Big(\frac{\partial x}{\partial \xi}u-\frac{\partial x}{\partial \xi}v\Big)_s\Big)\Big].\end{aligned} \tag{A.4}$$

For the last term we have

$$\tau_2^{\xi}\big|_n \doteq \Big(\frac{\partial y}{\partial \eta}\tau_{yx}-\frac{\partial x}{\partial \eta}\tau_{yy}\Big)_{hn}-\Big(\frac{\partial y}{\partial \eta}\tau_{yx}-\frac{\partial x}{\partial \eta}\tau_{yy}\Big)_{hnW}, \tag{A.5}$$

with

$$\begin{aligned}\Big(\frac{\partial y}{\partial \eta}\tau_{yx}-\frac{\partial x}{\partial \eta}\tau_{yy}\Big)_{hn} \doteq \frac{1}{J_{hn}}\Big[\Big(\frac{\partial y}{\partial \eta}\Big)_{hn}\Big(\Big(-\frac{\partial x}{\partial \eta}u+\frac{\partial y}{\partial \eta}v\Big)_{nE}-\Big(-\frac{\partial x}{\partial \eta}u+\frac{\partial y}{\partial \eta}v\Big)_n\\ -\Big(\frac{\partial x}{\partial \xi}u-\frac{\partial y}{\partial \xi}v\Big)_{eN}+\Big(\frac{\partial x}{\partial \xi}u-\frac{\partial y}{\partial \xi}v\Big)_e\Big)-\frac{2}{3}\Big(\frac{\partial x}{\partial \eta}\Big)_{hn}\Big(\Big(2\frac{\partial x}{\partial \eta}v+\frac{\partial y}{\partial \eta}u\Big)_{nE}\\ -\Big(2\frac{\partial x}{\partial \eta}v+\frac{\partial y}{\partial \eta}u\Big)_n-\Big(2\frac{\partial x}{\partial \xi}v+\frac{\partial y}{\partial \xi}u\Big)_{eN}+\Big(2\frac{\partial x}{\partial \xi}v+\frac{\partial y}{\partial \xi}u\Big)_e\Big].\end{aligned} \tag{A.6}$$

APPENDIX **B**

Flame solver

In this appendix we present the details of the numerical methods used by the laminar flame solver. A more exhaustive exposition can be found in [30].

B.1 Block Gauss-Seidel method

Let us suppose that our domain is covered with N grid cells. We define the vector $\boldsymbol{\psi}_i$ as the combination of the vectors $\boldsymbol{\varphi}$ and $\mathbf{v}$ consisting of the combustion and the velocity unknowns, respectively, at a certain grid cell. After discretisation, system (6.50b)-(6.50d) can be written in the following symbolic way

$$\frac{\boldsymbol{\psi}_i^n - \boldsymbol{\psi}_i^{n-1}}{\Delta t} = \mathbf{A}_i^n(\boldsymbol{\psi}_i^n, \boldsymbol{\psi}_j^n), \qquad j \in \mathcal{N}(i), \qquad i = 1, \cdots, N. \tag{B.1}$$

Here, the set $\mathcal{N}(i)$ contains the indices of the grid points that belong to the discretisation stencil of i. Since we use staggered grids, the combustion and velocity variables are not computed in the same location. Therefore, we associate to each cell of the basis grid the staggered points laying just north and east. If the function $\mathbf{F}$ is defined as

$$\mathbf{F}_i(\zeta_i, \zeta_j) := \frac{\zeta_i - \boldsymbol{\psi}_i^{n-1}}{\Delta t} - \mathbf{A}_i^n(\zeta_i, \zeta_j), \qquad j \in \mathcal{N}(i), \qquad i = 1, \cdots, N, \tag{B.2}$$

system (B.1) can be written as

$$\mathbf{F}(\boldsymbol{\psi}^n) = 0. \tag{B.3}$$

Equation (B.3) is solved with the multigrid method, see e.g. [66]. This can be seen as a speed up of a simple solution method, here block Gauss-Seidel implemented in a *matrix free* way, which is used as a smoother. Block Gauss-Seidel consists in solving (B.3) sequentially using the modified Newton method: the function $\mathbf{F}$ is evaluated using the updated values ψ_j^n if the point j has already been visited, i.e. $j < i$, and using the old values if the equations in $j > i$. This implies that, when modifying $\boldsymbol{\psi}_i$, the equations in the previous grid points are not satisfied anymore, thus more than one sweep for all the grid points is necessary.

B.2 Broyden iteration method

The well known modified Newton method for the solution of (B.3) reads

$$\boldsymbol{\psi}_i^{k+1} = \boldsymbol{\psi}_i^k - J_{ij}^{-1}(\boldsymbol{\psi}^k)\mathbf{F}_j(\boldsymbol{\psi}^k), \tag{B.4}$$

where J is the Jacobi matrix of $\mathbf{F}$

$$J_{ij} := \frac{\partial \mathbf{F}_i}{\partial \boldsymbol{\psi}_j}. \tag{B.5}$$

The necessity to evaluate the Jacobi matrix is one of the weak points of the procedure. This operation is in fact very expensive, especially when dealing with large systems. Moreover, it can be even impossible when F consists of *library-type* functions, i.e. when no explicit formula is available. A very good alternative is the Broyden iteration method, see [41]. This consist in iteratively finding an approximation $B(\psi^k)$, also called the *Broyden matrix*, for the inverse of the Jacobi matrix such that

$$\boldsymbol{\psi}_i^{k+1} = \boldsymbol{\psi}_i^k - \lambda B_i^k \mathbf{F}_i(\boldsymbol{\psi}^k), \tag{B.6}$$

given an initial guess B_i^0. In (B.6) λ is a relaxation parameter while the updated Broyden matrix is given by $B_i^{k+1} = B_i^k + \delta B$, with

$$\delta B = B_i^k \frac{(\lambda - 1)\mathbf{F}_i(\boldsymbol{\psi}^k) + \mathbf{F}_i(\boldsymbol{\psi}^{k+1})}{\| \mathbf{F}_i(\boldsymbol{\psi}^k) - \mathbf{F}_i(\boldsymbol{\psi}^{k+1}) \|^2} (\mathbf{F}_i(\boldsymbol{\psi}^k) - \mathbf{F}_i(\boldsymbol{\psi}^{k+1}))^T. \tag{B.7}$$

The matrix update is obtained by requiring that

$$(B_i^k + \delta B)(\mathbf{F}_i(\boldsymbol{\psi}^k) - \mathbf{F}_i(\boldsymbol{\psi}^{k+1})) = \boldsymbol{\psi}_i^k - \boldsymbol{\psi}_i^{k+1}, \tag{B.8}$$

i.e. B_i^{k+1} is a better approximation of the inverse of the Jacobi matrix. There are several choices to construct the update δB for the Broyden Matrix. It can be shown that (B.7) is the solution of (B.6) which minimises $\| \delta B \|_2$.

B.3 Multigrid solver

The multigrid method used in the code is described in [66]. We notice that, after a few Gauss-Seidel iterations, the residual decreases very slowly. In fact, while the small scale fluctuations are easily damped, the large scale fluctuations are somewhat more persistent: this implies that the representation of the residual should be used on a much coarser grid. Therefore, we define a series of increasingly coarser grids $\Omega^M, \Omega^{M-1}, \cdots, \Omega^1$. Let us suppose that, after a relatively small number of iterations on the finest level Ω^M, a solution $\boldsymbol{\psi}^M$ of the system

$$\mathbf{F}(\boldsymbol{\psi}^M) = \mathbf{f}^M, \tag{B.9}$$

is obtained (the dependence on the time is here omitted for the sake of simplicity). Then, we can define the defect as

$$\mathbf{r}^M := \mathbf{f}^M - \mathbf{F}(\boldsymbol{\psi}^M). \tag{B.10}$$

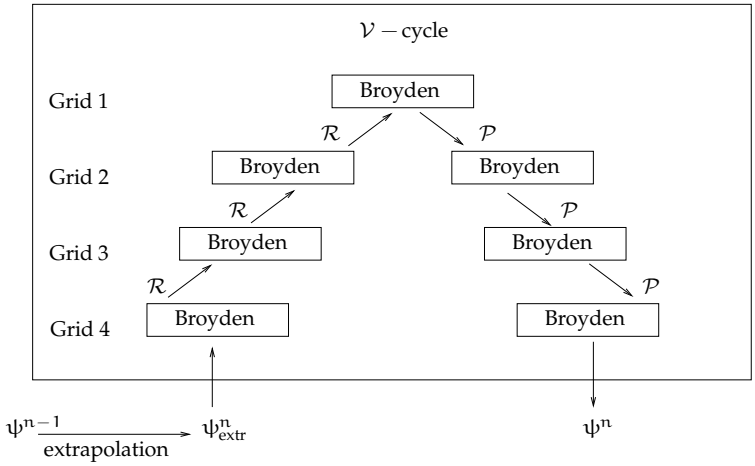


Figure B.1: $\mathcal{V}$-cycle.

If we define a restriction operator $\mathcal{R}^M : \mathcal{F}(\Omega^M) \to \mathcal{F}(\Omega^{M-1})$, we can define and solve a new problem on Ω^{M-1}

$$\mathbf{F}^{M-1}(\boldsymbol{\psi}^{M-1}) = \mathbf{F}^{M-1}\mathcal{R}^M[\boldsymbol{\psi}^M] - \sigma\mathcal{R}^M[\mathbf{r}^M], \tag{B.11}$$

where σ is a relaxation parameter. The so found approximation $\boldsymbol{\psi}^{M-1}$ is used to correct the solution on the finer grid

$$\boldsymbol{\psi}^M := \boldsymbol{\psi}^M + \mathcal{P}^{M-1}[\boldsymbol{\psi}^{M-1} - \mathcal{R}^M[\boldsymbol{\psi}^M]], \tag{B.12}$$

where $\mathcal{P}^{M-1} : \mathcal{F}(\Omega^{M-1}) \to \mathcal{F}(\Omega^M)$ is a prolongation operator.

In the laminar flame code, the multigrid method is implemented by using four different global grid levels. Both the restriction and the prolongation operators maps grid functions by using bi-linear interpolation. The so-called $\mathcal{V}$-cycle, see Figure B.1, is used to solve both the prediction and the correction steps of (6.50).

B.4 Least squares extrapolation in time

Although the multigrid $\mathcal{V}$-cycle is very efficient, convergence can still be improved by using *least squares extrapolation*. In fact, the information at the previous time steps can fruitfully help in finding a good approximation of $\boldsymbol{\psi}^n$ before entering the $\mathcal{V}$-cycle. We recursively define the search vectors as follows

$$\mathbf{s}_1^n := \psi^{n-1} - \psi^{n-2}, \qquad \mathbf{s}_2^n := \mathbf{s}_1^n - \mathbf{s}_1^{n-1}, \quad \cdots \quad \mathbf{s}_{N_k}^n := \mathbf{s}_{N_{k-1}}^n - \mathbf{s}_{N_{k-1}}^{n-1}. \tag{B.13}$$

Hence, nonlinear extrapolation yields

$$\boldsymbol{\psi}^n = \boldsymbol{\psi}^{n-1} + (\mathbf{s}_1^n, \cdots \mathbf{s}_k^n)\boldsymbol{\alpha} + \mathcal{O}(\Delta t^{N_k+1}), \tag{B.14}$$

where the vector $\boldsymbol{\alpha}$ is chosen in such a way that the residual is minimised in the two-norm. The approximation of $\boldsymbol{\alpha}$ is given by the least squares solution of the linearisation of problem (B.3). The residual function $\mathbf{F}$ from (B.3) can be used to compute the matrix M^n as

$$M^n = [\mathbf{F}(\boldsymbol{\psi}^{n-1} + \mathbf{s}_1^n) - \mathbf{F}(\boldsymbol{\psi}^{n-1}), \cdots, \mathbf{F}(\boldsymbol{\psi}^{n-1} + \mathbf{s}_{N_k}^n) - \mathbf{F}(\boldsymbol{\psi}^{n-1})]. \tag{B.15}$$

The vector $\boldsymbol{\alpha}$ is thus found as

$$\boldsymbol{\alpha} = ((M^n)^T M^n)^{-1} (M^n)^T \mathbf{F}(\boldsymbol{\psi}^{n-1}). \tag{B.16}$$

We must point out that, if N_k is not small enough, the method becomes very inefficient and suffers from round-off errors. According to the studies presented in [30], the choice $N_k = 3$ used by the laminar flame program has not shown these drawbacks.

Bibliography

[1] ABBOTT, J.P. and R.P. BRENT: *Fast local convergence with single and multistep methods for nonlinear equations.* J. Austral. Math. Soc., 19 (series B):173–199, 1975.

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

[3] ANTHONISSEN, M.G.H., R.M.M. MATTHEIJ and J.H.M TEN THIJE BOONKKAMP: *Convergence Results for the Local Defect Correction Method as an Iterative Process*. Numerische Mathematik, 95:401–425, 2003.

[4] ANTHONISSEN, M.J.H., 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.

[5] ASCHER, U.M., R.M.M. MATTHEIJ and R.D. RUSSEL: *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Prentice Hall, 1988.

[6] BENNET, B.A.V. and M.D. SMOOKE: *Local rectangular refinement with application to axisymmetric laminar flames*. Comb. Theory Modelling, 2:221–258, 1998.

[7] BENNET, B.A.V. and M.D. SMOOKE: *Local Rectangular Refinement with Application to Nonreacting and Reacting Fluid Flow Problems*. J. Comp. Phys., 151:684–727, 1999.

[8] BERESTYCKI, H. and B. LARROUTUROU: *A semilinear elliptic equation in a strip arising in a two-dimensional flame propagation model.* J. fur Reine und Angewandte Mathematiek, 396:14–40, 1989.

[9] BERESTYCKI, H. and B. LARROUTUROU: *Nonlinear partial differential equations and their application. College de France Seminar. Volume X.*, chapter Quelques aspects mathématiques de la propagation des flammes prémélangées, pages 65–129. H. Brezis and J.L. Lions editors. Longman Scientific and Technical, 1991.

[10] BERESTYCKI, H., B. LARROUTUROU and P.L. LIONS: *Multi-Dimensional Travelling-Wave Solution of a Flame Propagation Model*. Arch. Rat. Mech. Anal., 111:33–49, 1990.

[11] BOGGS, P.T.: *The solution of non-linear systems of equations by A-stable integration techniques.* SIAM J.Numer.Analysis, 8:767–785, 1971.

[12] BOUMA, P. H.: *Methane-Air Combustion on Ceramic Foam Surface Burners*. PhD thesis, Eindhoven University of Technology, 1997.

[13] BRAATEN, M. and W. SHYY: *A study of recirculating flow computation using body-fitted coordinates: consistency aspects and mesh skewness*. Numerical Heat Transfer, 9:559–574, 1986.

[14] BUCKMASTER, J.D. and G.S.S. LUDFORD: *Theory of Laminar Flames*. Cambrige University Press, 1982.

[15] COFFEY, T.S., C.T. KELLEY and D.E. KEYES: *Pseudo-transient continuation and differential-algebraic equations*. SIAM J. Sci. Comput., 25, No 2:553–569, 1998.

[16] DAVIES, C.W.: *An initial value Approach to the production of Discrete Orthogonal Coordinates*. J. Comp. Phys., 59:164–178, 1981.

[17] EGGELS, R.L.G.M.: *Modeling of Combustion Processes and NO Formation with Reduced Reaction Mechanism*. PhD thesis, Eindhoven University of Technology, 1996.

[18] EGGELS, R.L.G.M. and L.P.H. DE GOEY: *Post-Processing Method for Predicting NO Formation in One- and Two-Dimensional Premixed Methane/air Flames*. Combustion and Flame, 107:65–71, 1996.

[19] EGOLFOPOULOS, F., D. ZHU and C. LAW: *Experimental and numerical determination of laminar flame speeds: Mixtures of C_2-hydrocarbons with oxigen and nitrogen*. Twenty-Third symposium on combustion, 1990.

[20] FERKET, P.J.J.: *Numerical Treatment of Coupled Systems. Notes on Numerical Fluid Mechanics. Volume 51.*, chapter Coupling of a global coarse grid discretization and local fine discretization., pages 47–58. Hackbusch, W. and Wittum, G. editors. Braunschweig. Vieweg, 1995.

[21] FERKET, P.J.J. and A.A. REUSKEN: *Further Analysis of the Local Defect Correction Method*. Computing, 56:117–139, 1996.

[22] GRAZIADEI, M., R.M.M. MATTHEIJ and J.H.M. TEN THIJE BOONKKAMP: *Local defect correction with curvilinear grids: algorithm and application to laminar flames*. Submitted to Numerical Methods for Partial Differential Equations, 2003.

[23] GRAZIADEI, M., R.M.M. MATTHEIJ and J.H.M. TEN THIJE BOONKKAMP: *Local defect correction with slanting grids*. Numerical Methods for Partial Differential Equations, 20, Issue 1:1–17, 2004.

[24] GRAZIADEI, M. and J. H. M. TEN THIJE BOONKKAMP: *Local Defect Correction for Laminar Flames Simulation*. Submitted to Proceedings of the European Conference on Mathematics for Industry, 2004.

[25] GRIFFITHS, J.F. and J.A. BARNARD: *Flame and Combustion*. Chapman and Hall, 1995.

[26] HACKBUSCH, W.: *Local Defect Correction Method and Domain Decomposition Techniques*. Computing, Suppl. 5:89–113, 1984.

[27] HEIJER, C. DEN: *Iterative methods for solving non-linear equations when no good approximation to the solution is available*. Technical Report NW 46/77, Amsterdam, Mathematisch Centrum, Afdeling Numerieke Wiskunde, 1977.

[28] HEIJER, C. DEN: *Iterative solution of nonlinear equations by imbedding methods*. Technical Report NW 46/77, Amsterdam, Mathematisch Centrum, Afdeling Numerieke Wiskunde, 1977.

[29] HIRSCH, C.: *Numerical Computation of Internal and External Flows. Volume 1*. Chichester: Wiley, 1988-1990.

[30] HOF, B. VAN 'T: *Numerical Aspects of Laminar Flame Simulation*. PhD thesis, Eindhoven University of Technology, 2002.

[31] HOF, B. VAN 'T, J.H.M. TEN THIJE BOONKKAMP and R.M.M. MATTHEIJ: *Discretization of the stationary convection-diffusion-reaction equation*. Numer. Meth. Part. Diff. Eq., pages 608–625, 1998.

[32] HOF, B. VAN 'T, J.H.M. TEN THIJE BOONKKAMP and R.M.M. MATTHEIJ: *Pressure Correction for Laminar Combustion Simulation*. Comb. Sci. and Tech., 149:201–223, 1999.

[33] ISSA, R.I.: *Solution of the implicitly discretized fluid flow equations by operator-splitting*. J. Comp. Phys., 62:40–65, 1985.

[34] KELLEY, C.T. and D.E. KEYES: *Convergence analysis of pseudo-transient continuation*. SIAM J. Numer. Anal., 35, No. 2:508–523, 1998.

[35] KRAMER, M.E.: *Aspects of Solving Non-Linear Boundary Value Problems Numerically*. PhD thesis, Eindhoven University of Technology, 1992.

[36] LANGE, H.C. DE: *Modelling of premixed laminar flames*. PhD thesis, Eindhoven University of Technology, 1992.

[37] LANGE, H.C. DE and L.P.H DE GOEY: *Numerical flow modeling in a locally refined grid*. International Journal for Numerical Methods in Engineering, 37:497–515, 1994.

[38] LARROUTUROU, B.: *The equations of one-dimensional unsteady flame propagation: existence and uniqueness.* SIAM J. Math. Anal., 244:32–59, 1988.

[39] LARROUTUROU, B.: *Numerical Modeling in Combustion*, chapter Adaptive numerical simulation of premixed flame propagation, pages 177–230. Taylor and Francis, 1993.

[40] MALLENS, R.M.M.: *Stabilisation of Laminar Premixed Methane/Air Flames*. PhD thesis, Eindhoven University of Technology, 1996.

[41] MOHAMED, J. and J. WALSH: *Numerical algorithms*. Clarendon, Oxford, 1986.

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

[43] NEFEDOV, V. and R.M.M. MATTHEIJ: *Local defect correction with different grid types*. Technical Report RANA 01-11, Eindhoven University of Technology, Faculty of Mathematics and Computing Science, 2001.

[44] NONINO, C.: *A simple pressure stabilization for a simple-like equal-order FEM algorithm*. Numer. Heat Transfer, part B, 44:61–81, 2003.

[45] OIJEN, J. VAN: *Flamelet-Generated Manifolds: Development and Application to Premixed Laminar Flames.* PhD thesis, Eindhoven University of Technology, 2002.

[46] PATANKAR, S.V.: *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, New York, 1980.

[47] PETERS, N. and J. WARNATZ: *Numerical methods in laminar flames propagation, Notes on Numerical Fluid Mechanics; Vol. 6*. Braunschweig;Wiesbaden : Vieweg, 1982.

[48] ROOK, R.: *Acoustics in Burner-Stabilised Flames*. PhD thesis, Eindhoven University of Technology, 2001.

[49] SANI, R.L., P.M. GRESHO, R.L. LEE and D.F. GRIFFITHS: *The cause and cure (?) of the spurious pressure generated by certain FEM solutions of the incompressible Navier-Stokes equations: Part 1*. Int. J. for Num. Meth. in Fluids, 1:17–43, 1981.

[50] SANI, R.L., P.M. GRESHO, R.L. LEE and D.F. GRIFFITHS: *The cause and cure (?) of the spurious pressure generated by certain FEM solutions of the incompressible Navier-Stokes equations: Part 2*. Int. J. for Num. Meth. in Fluids, 1:171–204, 1981.

[51] SHYY, W. and M.E. BRAATEN: *Three dimensional analysis of the flow in a curved hydraulic turbine draft tube*. Int. J. for Num. Meth. in Fluids, 6:861–882, 1986.

[52] SHYY, W., S.S. TONG and S.M. CORREA: *Numerical recirculating flow calculation using a body-fitted coordinate system*. Numerical Heat Transfer, 8:99–113, 1985.

[53] SHYY, W. and T.C. VU: *On the Adoption of Velocity Variable and Grid System for Fluid Flow Computation in Curvilinear Coordinates*. J. Comp. Phys., 92:82–105, 1991.

[54] SIVASHINSKY, G.I.: *Instabilities, pattern formation and turbulence in flames*. Ann. Rev. Fluid Mech., 15:179–199, 1983.

[55] SMOOKE, M.D.: *Lecture Notes in Physics*, chapter Reduced Kinetics Mechanisms and Asymptotic Approximations for Methane-Air Flames, pages 90–96. Springer-Verlag, 1999.

[56] SOMERS, L.M.T.: *The Simulation of Flat Flames with Detailed and Reduced Chemical Models*. PhD thesis, Eindhoven University of Technology, 1994.

[57] SOMERS, L.M.T. and L.P.H. DE GOEY: *Analysis of a Systematical Reduction Technique*. In *Twenty-Fifth symposium on Combustion*. The Combustion Institute, Pittsburg, 1994.

[58] THIART, G.D.: *Finite difference scheme for the numerical solution of fluid flow and heat transfer problems on nonstaggered grids*. Numer. Heat Transfer, part B 17:41–62, 1990.

[59] THIART, G.D.: *Improved finite-difference scheme for the solution of convection-diffusion problems with the SIMPLEN algorithm*. Numer. Heat Transfer, part B 18:81–95, 1990.

[60] THIJE BOONKKAMP, J.H.M. TEN: *The conservation equations for reacting gas flow*. Technical Report 93-WSK-01, Eindhoven University of Technology, Faculty of Mathematics and Computing Science, 1993.

[61] THOMPSON, J.F., B.K. SONI and N.P. WEATHERILL: *Handbook of Grid Generation*. London: CRC Press LLC, 1999.

[62] THOMPSON, J.F., Z.U.A. WARSI and C.W. MASTIN: *Numerical Grid Generation (Foundations and Applications)*. Amsterdam: North Holland, 1985.

[63] VALDATI, B.A.: *Solution-adaptive gridding methods with application to combustion problems*. PhD thesis, Yale University, New Haven, CT, 1997.

[64] WAPPLER, J.U.: *Die lokale Defektkorrekturmethode zur adaptiven Diskretisierung elliptischer Differentialgleichungen mit finiten Elementen*. PhD thesis, Christin-Albrechts-Universität, Kiel, 1999.

[65] WARNATZ, J., U. MAAS and R.W. DIBBLE: *Combustion*. Springer, 1996.

[66] WESSELING, P.: *An Introduction to Multigrid Methods*. Wiley, Chichester, 1992.

[67] WIJNGAART, R.F. VAN DER: *Composite Grid Techniques and Adaptive Mesh Refinement in Computational Fluid Dynamics*. PhD thesis, Stanford University, 1990.

[68] WILLIAMS, FORMAN A.: *Combustion Theory, The Fundamental Theory of Chemically Reacting Flow Systems*. Addison-Wesley Publishing Company, Redwood City, 1985.

Index

Summary

In this research we face some of the problems related to the simulation of laminar flames. These, as well as many others BVPs representing natural phenomena (shock waves, semiconductor devices, etc.), are characterised by the presence of regions where gradients are quite large compared to those in the rest of the domain, where the solution can be considered relatively smooth. When solving such problems numerically, this solution behaviour requires a much finer grid in these *high activity* regions than in regions where the solution is fairly smooth. This is the case, for instance, of the equations that describe laminar flames: most of the activity is concentrated in the flame front, a narrow area where the temperature rises steeply and chemical reactions take place. In all these cases, the Local Defect Correction (LDC) method reveals to be a quite powerful tool. Roughly speaking, the method goes as follows. First, an approximation of the solution on a coarse grid in the entire domain is computed. Then, this is used to define a boundary value problem on a subdomain where a different grid is employed. The thus found solution on the latter subdomain is used to define a defect on the global domain, which, in turn, induces an updated global domain problem. This so-called local defect correction (LDC) method is used in an iterative way. It converges very fast: typically one iteration suffices. So far, the LDC method has been used in combination with rectangular fine grids only. Since this is not a theoretical requirement, we combine a global coarse grid with different types of fine grids, with the aim of minimizing the total number of grid points.

First, we start with a rectangular fine grid that is slanted over a certain angle with respect to the coarse one in order to better follow the shape of the high activity region. The advantages of using a slanting grid are mainly that the implementation is simple, i.e. the equations can be transformed just applying a rotation, and a signifivant reduction in the number of fine grid points can be achieved, as assessed by the complexity analysis. The only drawback is represented by the fact that the fine grid sticks out of the global domain. However, show that also this problem can be easily solved.

The next step is to combine LDC with curvilinear grids. Although the transformation of the equations is, in this case, more involved, the use of a curvilinear grid reveals to be necessary when the shape of the high activity region is more complex. A solution fitted fine grid that tightly follows the shape of the solution is built by using a trajectory method. This takes an existing nonorthogonal grid and transforms it into an orthogonal one. One set of lines of the original nonorthogonal grid is retained; the other is

displaced, through a marching process, such that the intersection between the two sets of lines is orthogonal. The set of lines that is retaind is chosen coincident with the iso-contours of the coarse grid solution. This allows to obtain a very accurate local solution with a very limited amount of grid points. Also in this case, a complexity analysis supports our results. Both the algorithm that use slanting and curvilinear grids are tested by computing the solution of some model problems.

Then, we use LDC in combination with curvilinear grids to solve two different 2D laminar flame problems. The first one is the thermo-diffusive model. It is based on the constant pressure and constant density assumptions, that allow to simplify the equations. Though being simple, the resulting model gives some insight in the structure of the flame. The shape of the flame front makes it suitable to be solved with our algorithm. Next, we study Bunsen flames. In this case the equations are much more complicated. The fine grid problem, as well as the coarse one, is solved with a pressure correction technique. To prevent pressure oscillations staggered grids are used. Nevertheless, when a curvilinear grid turns 90 degrees, loss of staggering effectiveness can occur. The remedy that we use here is to apply pressure smoothing.

The discretisation of the combustion equations gives rise to systems that are highly nonlinear and that pose several problems when being solved. In these cases the Newton method reveals to be not robust enough. This has led us to another part of our research. We consider embedding techniques to deal with nonlinear systems and we try to develop alternative algorithms that improve their robustness. Those methods are tested using benchmark problems found in literature.

Samenvatting

In dit onderzoek beschouwen we enkele van de vraagstukken met betrekking tot de simulatie van laminaire vlammen. Zoals vele andere randwaardeproblemen, die natuurlijke fenomenen beschrijven (schokgolven, halfgeleider-apparaten, enzovoort), worden deze gekenmerkt door de aanwezigheid van gebieden waar de gradiënten zeer groot zijn ten opzichte van de rest van het domein, waar de oplossing relatief glad is. Bij het numeriek oplossen van dergelijke problemen dient een veel fijner rooster gebruikt te worden binnen deze zones met hoge activiteit dan waar de oplossing redelijk glad is. Dit is bijvoorbeeld het geval bij de vergelijkingen die laminaire vlammen beschrijven. Het merendeel van de activiteit is geconcentreerd in het vlamfront, een smal gebied, waar de temperatuur scherp stijgt en scheikundige reacties plaatsvinden. In al deze gevallen blijkt de methode van lokaal tekort-correctie (Local Defect Correction, of kortweg LDC) een zeer krachtig gereedschap. Grofweg werkt de methode als volgt. Eerst wordt een benaderende oplossing op het grove rooster in het hele domein berekend. Deze wordt vervolgens gebruikt om een randwaardeprobleem op een subdomein te definiëren, waar een ander, fijner, rooster wordt ingezet. De oplossing op het subdomein wordt gebruikt om een defect op het globale domein te definiëren, wat dan weer leidt tot een aangepast probleem om het globale domein. LDC wordt iteratief gebruikt, maar het convergeert doorgaans erg snel: meestal is één iteratieslag al toereikend. Voorheen is LDC alleen met rechthoekige fijne roosters gebruikt. Omdat dit echter geen theoretische eis is, hebben we geprobeerd een globaal grof rooster te combineren met verschillende types fijne rooster met als doel het totale aantal roosterpunten te minimaliseren.

We zijn begonnen met een rechthoekig fijn rooster, dat een bepaald aantal graden is verdraaid ten opzichte van het globale rooster, om zo de vorm van de hoge activiteitszone beter te kunnen volgen. Een voordeel van het gebruik van zo'n schuin rooster is, dat het eenvoudig is te implementeren, dat wil zeggen, de vergelijkingen hoeven alleen aangepast te worden door een draaiing toe te passen. Een ander voordeel is, dat het aantal fijne roosterpunten aanzienlijk wordt teruggebracht, zoals wordt vastgesteld in een complexiteitsanalyse. Het enige nadeel wordt gegeven door het feit, dat het fijne rooster buiten het globale domein uitsteekt. We tonen echter aan, dat dit probleem eenvoudig kan worden opgelost.

De volgende stap was om LDC met curvilineaire roosters te combineren. Hoewel de transformatie van de vergelijkingen in dit geval meer werk vereist, blijkt een curvilin-

eair rooster noodzakelijk als de vorm van de hoge activiteitszone complexer is. Een aan de oplossing aangepast rooster wordt hier opgebouwd met een baanmethode. Deze neemt een bestaand niet-orthogonaal rooster en maakt dit orthogonaal. Eén lijnenset van het oorspronkelijke niet-orthogonale rooster wordt hierbij behouden, terwijl het andere wordt verplaatst door gebruik van een stapsgewijs proces, zodat de twee lijnensets uiteindelijk orthogonaal zijn. De lijnenset, die wordt behouden, wordt gekozen om samen te vallen met de isocontouren van de oplossing op het grove rooster. Hierdoor kan een zeer nauwkeurige oplossing verkregen worden met slechts een beperkt aantal roosterpunten. Ook in dit geval worden onze resultaten onderschreven door een complexiteitsanalyse. De beide algoritmes voor schuine en curvilineaire roosters zijn getest met enkele modelproblemen.

Daarna hebben we LDC tezamen met curvilineaire roosters toegepast om twee verschillende laminaire vlamproblemen op te lossen. Het eerste is het zogenaamde thermodiffusief model. Het is gebaseerd op aannames van constante druk en constante dichtheid, wat vereenvoudigingen in de vergelijkingen toelaat. Hoewel eenvoudig geeft het resulterende model ons inzicht in de structuur van de vlam. De vorm van het vlamfront maakt het geschikt om met onze methode op te lossen. Vervolgens hebben we Bunsenvlammen bestudeerd. In dit geval zijn de vergelijkingen veel gecompliceerder. Zowel het fijne als het grove rooster wordt met een drukcorrectie-techniek opgelost. Om oscillaties in de druk te verkomen worden alternerende roosters ingezet. Desalniettemin kan de effectiviteit van het alterneren verloren gaan, als het curvilineaire rooster een bocht van 90 graden maakt. De remedie die we er hier tegen gebruiken is de druk glad te strijken.

De discretisatie van de verbrandingsvergelijkingen leidt tot stelsels, die zeer niet-lineair zijn en een oplossing met verschillende problemen bemoeilijken. Hier blijkt de methode van Newton niet robuust genoeg. Dit heeft tot een ander onderdeel van ons onderzoek geleid. We kijken hierbij naar inbeddingstechnieken om met de niet-lineaire stelsels te behandelen en ontwikkelen daarbij alternatieve methoden om hun robuustheid te verbeteren. Deze methoden zijn getest met maatstaf-problemen die in de literatuur gevonden kunnen worden.

CVRRICVLVM VITÆ

The author of this thesis was born in Pagani, Italy, on April 28th 1970. After completing pre-university schooling at the Liceo Scientifico in Santa Maria Capua Vetere, Italy, in 1989 she moved to Rome, where she studied Aeronautical Engineering at the University 'La Sapienza'. During the last year of university, she worked as design engineer at Umbra Cuscinetti, (Foligno, Italy), where she developed her master thesis project. In November 1997, she moved to GE/Nuovo Pignone (Florence, Italy) to work as gas turbine engineer. She came to The Netherlands in September 2000 and started the PhD research discussed in this dissertation at the Scientific Computing Group of the Eindhoven University of Technology. Since October 2004 she works at the Mechanics, Design & Heat Technology group at Philips Lighting.