Institut für Energie- und Kraftwerkstechnik
Technische Universität Darmstadt

Large Eddy Simulation of a Counter Flow Configuration

DIPLOMARBEIT

Submitted for the "Diplom-Ingenieur" degree

Andreas Kempf
Technische Universität Darmstadt

Supervisor: Professor J. Y. Chen
Co-Supervisor: Dr. H. Forkel

Department of Mechanical Engineering
University of California, Berkeley

7 Oktober 1999

Eidesstattliche Erklärung

"Ich versichere hiermit, daß ich die vorliegende Arbeit selbstständig verfasst, keine anderen als die angegebenen Hilfsmittel verwendet und die Stellen, die anderen Werken im Wortlaut oder dem Sinne nach entnommen sind, mit Quellenangaben kenntlich gemacht habe."

Andreas Kempf

Abstract

A *Large Eddy Simulation* (LES) was applied to study the flow and mixture field in a counterflow burner configuration. Various new boundary conditions were implemented, a comparison of inflow conditions was performed. This is necessary since the inflow condition is of fundamental importance for any LES. To get information about the results quality, setups similar to those investigated by *Mastorakos* [4] and *Sardi* [5] were studied.

Contents

Table of Contents **1**

List of Figures **3**

1 Introduction **5**

2 An Introduction to LES **6**

3 The Counter-Flow Configuration **12**

3.1 Counterflow Burners in General 12

3.2 The Mastorakos/Sardi Burners 13

4 The Numerical Setup **14**

4.1 The FLOWSI Flow-Predictor 14

4.2 The Simplified Burner Investigated 15

4.3 The Grid 16

4.4 The Boundary Conditions 16

4.5 Post-processing 22

5 Coupling and Parallelisation **24**

5.1 Coupling for the Counter-Flow 24

5.2 A Simple Method of Coupling separate Flow Fields 25

5.3 Communication 26

6 Results **29**

6.1 Mastorakos' Case 29

6.2 Sardi's Case 37

6.3 Comparison of Sardi and Mastorakos 39

7 Conclusion and Future Work **41**

A Advanced Boundary Conditions **43**

A.1 Determination of Wall Reynolds Number 43

A.2 Inflow Condition for Pipe 44

A.3 Outflow Condition for Pipe 45

B Modified Subroutines **46**

B.1 Modifications for Pipe-Flow with Wall Law 46

B.2 Modifications for Counterflow 48

B.3 Modifications for Pipe-flow with Dimensional Values 48

C Coupling **50**

Bibliography **52**

List of Figures

2.1 Sketch of typical turbulent spectrum 8

3.1 Burner configuration 13

4.1 The computational domain 15
4.2 The domain's configuration 16
4.3 The numerical grid 17
4.4 The perforated plate 19
4.5 Degenerated circles 20
4.6 Coupling of domains 21

5.1 Data exchange between domains 25

6.1 Snapshot of velocity field 30
6.2 Development of the Mixture Fraction in Time 31
6.3 Axial velocities 32
6.4 Mean values of the mixture fraction 33
6.5 Fluctuation of mixture fraction 34
6.6 Fluctuations of axial velocity 35
6.7 PDFs of the mixture fraction 36
6.8 Mean value of the mixture fraction 37
6.9 Fluctuation of mixture fraction 37

6.10 PDF of the mixture fraction 38
6.11 Mixture Fraction in Time 39
6.12 Mixture Fraction in Space 40
6.13 Fluctuation of Mixture Fraction 40

Chapter 1

Introduction

With further development of technology real progress in engineering is only possible on problems recently considered too complicated to be studied. A field with difficult questions to be answered is combustion. Although of general use, knowledge about this technology is somehow superficial, leaving a great potential to improve efficiency and to reduce pollutants.

To acquire deeper knowledge, basic cases must be studied. Very interesting ones are so called "counter-flow burners", studied in this work.

These counter-flow burners are well suited to study and calibrate models for non-premixed flames, e.g. *flamelet*-models. The counterflow is very interesting since it is possible to simulate this arrangement with the most simple one-dimensional flow solvers using RANS[1] models, computing the flow-field on the axis. Thus, solutions can be acquired very fast, and even complicated chemistry can be simulated.

For the experimental part, the investigation of counter-flow burners requires the most advanced experimental techniques of Laser-Diagnostics: More conventional methods are not able to gain the relevant data.

This report deals with the simulation part. A *Large Eddy Simulation* was applied to compute both turbulent flow and mixing. This costly approach was chosen since it promises information unavailable from RANS-modelling. Furthermore, it promises good results without calibrating model-parameters to match experimental data.

[1] Reynolds averaged Navier Stokes

Chapter 2

An Introduction to LES

To predict the flow- and mixing-fields, the conservation equations for species and momentum must be solved.

The most fundamental principle of conservation is the conservation of mass. It is described by the continuity equation (Eq. 2.1) for compressible fluids:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0 \tag{2.1}$$

This equation[1] sets the sum of the density's (ρ) change in time and the mass flow's (ρu_j) divergence to be zero.

To conserve momentum, Eq. 2.2 must be fulfilled for $i = 1, 2, 3$:

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = \frac{\partial}{\partial x_j}\tau_{ij} - \frac{\partial p}{\partial x_i} + \rho g_i \tag{2.2}$$

This equation's left-hand side consists of an accumulation term (acceleration), describing the change of momentum in time; and an convection term, describing convective momentum-transport. The right-hand side is formed with the gradient of the Stress-Tensor τ_{ij}, the gradient of the pressure p and

[1] In this work, the Einstein-convention is used. Thus, $\phi_j \phi_j$ stands for $\sum_{j=1}^{3} \phi_j \phi_j$.

the volume-forces ρg_i (e.g. describing gravitational forces).
In case of Newtonian fluids (gases, water...) the stress tensor τ_{ij} is defined by:

$$\tau_{ij} = \rho\nu \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) - \frac{2}{3}\rho\nu \frac{\partial u_k}{\partial x_k}\delta_{ij} \tag{2.3}$$

Inserting the stress-tensor into the momentum equation results in the famous Navier-Stokes-Equations.

The description of mixing is performed by a transport equation for the mass fraction x_α. This item describes the ratio of the mass m_α of specie α to the mass of all species m. A transport-equation for the mass fraction reads (Forkel [6]):

$$\frac{\partial}{\partial t}(\rho x_\alpha) + \frac{\partial}{\partial x_j}(\rho x_\alpha u_j) = \frac{\partial}{\partial} \left(\rho D_\alpha \frac{\partial x_\alpha}{\partial x_j} \right) + S_{x_\alpha} \tag{2.4}$$

The first term on the left-hand side describes the local change of the mass of species α in time. The second term describes it's convective transport. The right-hand side describes the diffusive transport of this specie, due to diffusivity D. The final source-term, S_{x_α} describes production and destruction by chemical reactions.

These differential equations describe turbulent mixing entirely. Even the Kolmogorov scales would be predicted if an exact solution for these equations was possible. Unfortunately, only numerical approaches are known. These methods rely on discretisation in space. But turbulence happens on scales too small to be resolved by realistic numerical grids. Thus, another way to describe turbulence is necessary.

Very promising approaches are so called *Large Eddy Simulations*, used in this study. They rely on most of the turbulent kinetic energy existing on large scale structures. This can be seen in Fig. 2.1, where the turbulent kinetic energy is plot over the wavenumber k.

Those *Large Eddies* can be resolved without the use of too fine grids. Certainly, these grids are still too coarse to predict turbulence on *Kolmogorov-Scales*. So the finest turbulent structures are still modeled. On these small

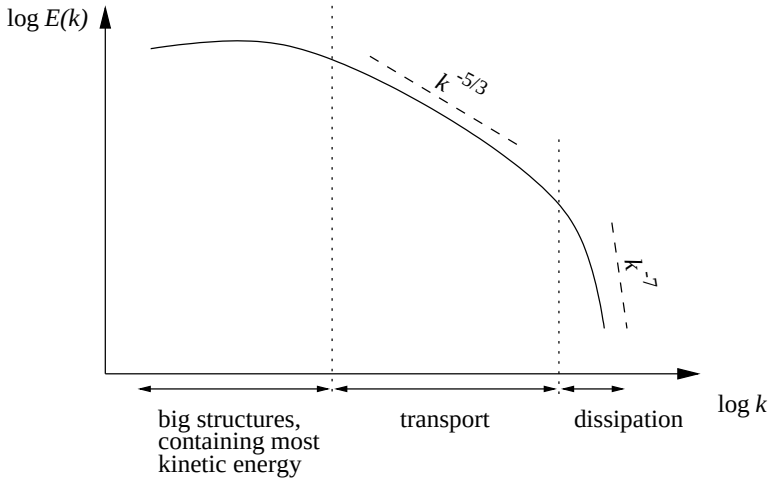

Figure 2.1: Sketch of typical turbulent spectrum

structures, dissipation and kinetic energy are distributed almost isotropic. This allows the use of very simple models to describe these fluctuations. For example, adapted zero-equation models work fine here.

To make the conservation equations describe only large scale turbulence, all fields are transformed by so called *"Filter Functions"* h. These *low-pass filters* remove all but the largest fluctuation, resulting in locally space averaged fields.

According to [2], the large scale structure of the relevant field $f(x_j, t)$ is obtained by averaging in space the product of the filter function $h(x_j - x'_j)$ (in local coordinates) and the field. Thus, the filtered field is defined by the following integral:

$$\overline{f(x_j,t)} = \int_{-\infty}^{\infty}\!\!\int\!\!\int h\left(x_j - x'_j\right) f\left(x'_j\right) \mathrm{d}x'\mathrm{d}y'\mathrm{d}z' \tag{2.5}$$

A typical filter for numerical simulation, using discretisation in space, is *Schumann's* filter. This filter is a rectangle filter with the filter width Δ_j equal to the width Δx_j of the grid cells. This equality will simplify the further proceeding.

A common rectangle filter (top hat filter) is defined by the following equation for $|x_j| \leq \frac{\Delta_j}{2}$:

$$h(x_j) = \prod_{j=1}^{3} 1/\Delta_j \tag{2.6}$$

For $|x_j| > \frac{\Delta_j}{2}$, the filter function $h(x_j)$ is null.

Using Schumann-filtering in space, the (2.5)-equation's integration interval can be narrowed. Then, integration is just done from $\phi - \Delta\phi/2$ to $\phi + \Delta\phi/2$ since h is null outside. In this document, $(\phi - \Delta\phi/2)$ is referred to as ϕ^-, accordant $(\phi + \Delta\phi/2)$ as ϕ^+.

$$\overline{f(x_j,t)} \;=\; \frac{1}{\Delta x\,\Delta y\,\Delta z}\int_{x^-}^{x^+}\int_{y^-}^{y^+}\int_{z^-}^{z^+} f\left(x'_j,t'\right)\,\mathrm{d}x'\,\mathrm{d}y'\,\mathrm{d}z' \tag{2.7}$$

Using this filtering technique, the simplified ($\rho = const.$, $g_i = 0$) momentum equation reads:

$$\underbrace{\frac{\partial}{\partial t}\,{}^{V}\overline{U_i}}_{\text{accumulation}} \;+\; \underbrace{\delta_j\,{}^{j}\overline{U_i U_j}}_{\text{convection}} \;=\; \underbrace{-\frac{1}{\rho}\,\delta_i\,{}^{i}\overline{P}}_{\substack{\text{pressure}\\ \text{gradient}}} \;+\; \underbrace{\delta_j\left(\nu\delta_j\,{}^{j}\overline{U_i}\right)}_{\text{diffusion}} \tag{2.8}$$

In equation (2.8), ${}^{V}\overline{U_i}$ is the volume averaged velocity component U_i.

The δ_x-operator of any scalar ϕ is similar to a central differencing scheme to discretise $\partial\phi/\partial x$. It is defined by:

$$\delta_x\phi = \left[\phi\left(x^+\right) - \phi\left(x^-\right)\right]/\Delta x \tag{2.9}$$

Finally, the surface averaged value ${}^{x}\overline{\phi}$ of a scalar ϕ is defined as:

$${}^{x}\overline{\phi}\left(x^+,y,z,t\right) = \frac{1}{\Delta y\Delta z}\int_{y^-}^{y^+}\int_{z^-}^{z^+} U\left(x^+,y',z',t\right)\,\mathrm{d}y'\,\mathrm{d}z' \tag{2.10}$$

According to [2], the momentum equation (2.8) is minute. Nevertheless, for the numeric solution the following assumptions are made:

$$ {}^{V}\overline{U}_i \approx {}^{i}\overline{U}_i \qquad {}^{V}\overline{P} \approx {}^{i}\overline{P} \tag{2.11} $$

Since this filtering doesn't introduce new information, the system of equations is not closed yet. Defining the fluctuation $u_i' = U_i - {}^{j}\overline{U_i}$, the convective term ${}^{V}\overline{U_i U_j}$ is not known entirely:

$$ {}^{j}\overline{U_i U_j} = {}^{j}\overline{U_i}\,{}^{j}\overline{U_j} + \tau_{ij}^{SGS} \tag{2.12} $$

Using the assumptions (2.11), the Schumann-filtered incompressible Navier-Stokes-equations read:

$$ \underbrace{\frac{\partial}{\partial t}\,{}^{i}\overline{U_i}}_{\text{accumulation}} + \underbrace{\delta_j\left({}^{j}\overline{U_i}\,{}^{j}\overline{U_j}\right)}_{\text{convection}} = \underbrace{-\frac{1}{\rho}\,\delta_i\,{}^{V}\overline{P}}_{\substack{\text{pressure}\\ \text{gradient}}} + \underbrace{\delta_j\left(\nu\delta_j\,{}^{i}\overline{U_i} - \tau_{ij}^{SGS}\right)}_{\text{(turbulent) diffusion}} \tag{2.13} $$

This equation describes the flow field together with the continuity equation (2.14).

$$ \delta_j\,{}^{j}\overline{U_j} = 0 \tag{2.14} $$

Still to be modeled are the small scale fluctuations τ_{ij}^{SGS}. This is done by so called sub-grid stress models. A typical representative is the one developed by *Smagorinsky* in 1965. His zero-equation model uses a modified *Boussinesq* approach to describe the non closured term τ_{ij}^{SGS}:

$$ -\left(\tau_{ij}^{SGS} - \frac{1}{3}\tau_{kk}^{SGS}\right) = 2\nu_t\overline{S_{ij}} \tag{2.15} $$

The tensor of deformation velocity $\overline{S_{ij}}$, and the turbulent viscosity ν_t are modeled by the following equations:

$$\overline{S_{ij}} = \frac{1}{2}\left(\frac{\partial \overline{U_i}}{\partial x_j} + \frac{\partial \overline{U_j}}{\partial x_i}\right)$$

$$\nu_t = c^2 \overline{\Delta}^2 \sqrt{\overline{S_{mn}}\,\overline{S_{mn}}}$$

The constant $c \approx 0.2$ slightly varies from case to case, $\overline{\Delta}$ describes the filter-width.

Although this is only a very simple way to model small scale turbulence, this approach is used with good success.

Chapter 3

The Counter-Flow Configuration

This chapter describes the counter-flow configuration investigated in this study. First, some introduction to counter-flow burners is given. Then, the investigated case is presented. Finally, the simplifications for the numerical study are mentioned.

3.1 Counterflow Burners in General

This study examines the flow- and scalar fields of so called "counter-flow burners". This configuration is well suited to study combustion, for both its simplicity and versatility. It enables the researcher to study a wealth of interesting problems with different fuel and it features many parameters which can easily be varied – without too big changes neither in experimental nor in numerical setups.

For example, the counterflow setup is an interesting validation case for flamelet models. As well, it is suited to study boundary conditions – crucial with any LES: The area of interest (mixing layer) and the inflow are so close together that the turbulence forced on the inflow has major influence on the results.

3.2 The Mastorakos/Sardi Burners

The burners studied in this work are those investigated by Mastorakos and Sardi ([4], [5]):

Two opposed nozzles of diameter D are arranged in line, only separated by a distance H. Fig. 3.1 shows a sketch of this setup.

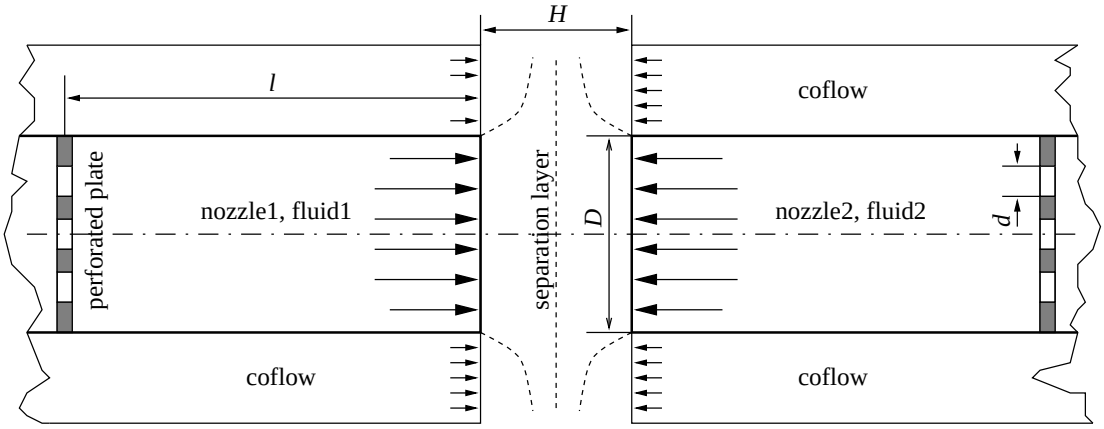

Figure 3.1: Burner configuration

To study non-premixed flames, one nozzle is set to spend fuel, the other one plain air. As well, premixed flames or non-reactive mixing can be investigated.

To excite predictable turbulent oscillations in the nozzle, a perforated plate is inserted in the pipe, short (l) upstream the nozzle. This circular plate is perforated with holes of diameter d, resulting in a permeability of 0.45. The following table opposes the geometry of Mastorakos' burner to Sardi's. The other values are bulk velocity $\overline{U}$, Reynolds number Re, friction number λ, friction velocity u^* and finally, the Reynolds number Re^* based on u^*.

case	$\overline{U}$	D	d	H	l	Re	λ	u^*	Re^*
	m/s	mm	mm	mm	mm	$1 \cdot 10^3$	$1 \cdot 10^{-3}$	m/s	1
Mast.	1.48	25.4	4	20	55	2.64	44.0	0.110	157
Sardi	3.84	30.0	4	30	50	7.50	32.4	0.244	415

(using air: $\eta = 1,84 \cdot 10^{-5}\mathrm{Ns/m^2}$; $\rho = 1.29\mathrm{kg/m^3}$)

Chapter 4

The Numerical Setup

After the physical configuration has been described, this chapter deals with the applied numerics. First, the methods used in the *flowsi* flow- and chemistry code are described. Then the numerical grid and the boundary conditions are presented.

4.1 The FLOWSI Flow-Predictor

For this study, *flowsi* was used. It was developed by Schmitt [8] for DNS and LES. New capabilities were added by Unger [9], Weinberger [10] and Forkel [6].

Flowsi solves the equations of conservation for mass, momentum and species on structured cartesian (or cylindric) grids with equidistant nodes. This limitation to very simple geometries is necessary to reduce computational costs and errors, rendering possible LES or DNS on workstations. To fulfil continuity, a pressure correction scheme for incompressible fluids is applied on staggered grids. For time integration, a Runge Kutta scheme was chosen. The explicit algorithm's time step width is computed by setting the flowfield's maximum courant number to 0.7[1].

To model sub-grid turbulence, a (dynamic[2]) Smagorinsky model is imple-

[1]This courant number is rather high, confirming the qualities of the *flowsi* code

[2]Germano enhanced model: Model constants are determined automatically. This is done by using two different filter widths and determining the model constant such that the underlying model is consistent with the different resolved fluctuation on the two levels

mented.

All in all, *flowsi* is a very convenient tool for the computation of benchmark flows. It is not able to predict flows in complicated geometries.

4.2 The Simplified Burner Investigated

To simulate reality, the actual setup must be transformed into a virtual one. This modeling includes simplifications which should not have any effects on the results.

For example, only a *finite* flow-field can be computed. The part investigated is the cylindric domain between the nozzles. It is presented in Fig. 4.1. It's radius was set to 1.6 times the radius of Mastorakos' pipe. This means that the outflow is close to the counterflow field. Though, the area of interest is the axis, because hardly any experimental data is available for other regions. Thus, a small domain is used to minimise CPU-time.

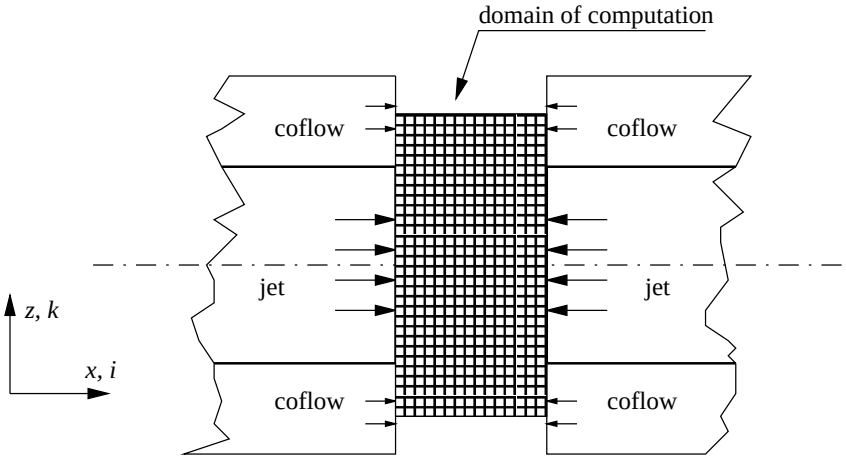

Figure 4.1: The computational domain

Both nozzles and Co-flow are modeled using an inflow-condition on the abutting face of the cylinder: On the inner section of this surface (see Fig. 4.2), velocities corresponding the flow on the nozzle are forced. These velocities do fluctuate to model the turbulent flow out of the pipe. The approach

of filtering.

to determine these fluctuations has great influence on the results. The approaches applied are described in 4.4.

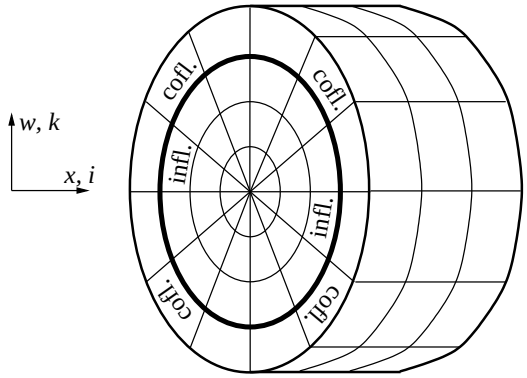

Figure 4.2: The domain's configuration

On the outer ring of the abutting surface, an laminar inflow is forced. In experiments, this co-flow is intended to minimise exterior influences on the flow-field. Of course, this is not necessary for numerical studies. Nevertheless, it is simulated to get results comparable to those acquired in experiments.

On the domain's outer surface, a simple outflow condition was applied.

4.3 The Grid

To discretise equations, a numerical grid is needed. Since the mean fields are axis-symmetric, a cylindrical grid is straight forward. Figure 4.3 shows an illustration.

During development, this Grid consisted of 65 cells in axial direction, corresponding to 25 and 64 cells in radial and tangential direction.

4.4 The Boundary Conditions

This section describes the applied boundary conditions. Since they represent a major link between physics and numerical study, they are essential for any

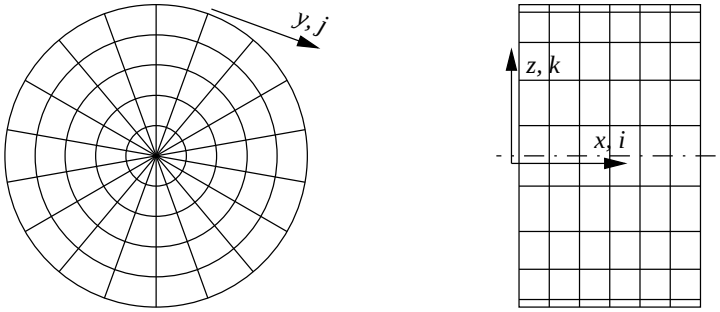

Figure 4.3: The numerical grid

simulation. Thus, great care has to be taken.

The formulation of realistic boundary conditions is *the* unsolved problem of LES.
A typical problem concerning boundary conditions is dealing with walls. Fortunately, this is not necessary here since the considered geometry shows no walls at all!
Another problem are inflow conditions. During this study, different approaches were developed to improve results and predictability. The more advanced approaches were developed by just improving on the previous ones. On the following pages, all four cases are presented, ordered by degree of sophistication.

Finally, the outflow conditions are presented.

4.4.1 Inflow using Random Noise

The most simple condition is copying the local mean $\overline{\phi}$ of a scalar ϕ to its according inflow cell. Nevertheless, this neglects unknown turbulent fluctuation: Only laminar inflow is forced.

To improve on this, turbulent fluctuations can be modeled by just adding random noise. This noise is generated by a random number generator; its level is known from the variance, $\overline{\phi'\phi'}$.

The drawback of this approach is the lack of any information about length-scales, energy spectra and cross correlations ($\overline{\phi_i'\phi_j'}$ for $i \neq j$). Indeed, only two values are known for each scalar (mean and variance). The superiority of

LES compared to RANS-models[3] is hardly exploited this way. Nevertheless, in many cases there is no alternative to adding noise, rendering this an approach of common use in LES or DNS.

To determine reasonable mean values $\overline{\phi}$ and fluctuation levels $\overline{\phi'\phi'}$, data from measurements was used. Since (to the authors knowledge) no data exists for nozzles, it was fallen back on pipe-flows[4]. No real outflow is simulated, representing a restriction to this simulation.

All in all, this boundary condition proved to be unsatisfying – more complex ones were to be considered.

4.4.2 Inflow using Pipe-flow

To resolve the problem of unreasonable cross correlations and spectra, another approach was considered:
The simulation of a fully developed turbulent pipe flow results in turbulence similar to that in the nozzle. This turbulent velocity field is copied to the inflow of the counterflow domain, delivering a strongly improved description of the inflow:

First of all, the flow field in a pipe of infinitesimal length was calculated. This is done by applying periodic boundary conditions to a finite pipe segment. To get a full spectrum of turbulence, this pipe has to be longer than the integral length-scale. Thus, its length must be far greater than its diameter.

For each time-step, the data of a pipe's slice was written to a file which is read by the program computing the counter flow.

4.4.3 Inflow with Modeled Perforated Plate

Using the pipe-flow condition delivers realistic turbulence. Though, it is just realistic for an infinitesimal pipe. In reality, Mastorakos and Sardi used perforated plates short of the nozzles to generate predictable turbulence. I.e. the fluctuations in the nozzle are dominated by the plate, even wake effects may occur.

An infinitesimal pipe can not describe this flow field. To improve on this, a

[3]Reynolds Averaged Navier Stokes
[4]The same approach as chosen by Forkel to simulate reactive turbulent jet flow [6].

new setup was created. Its grid and geometry are just the same as in the case of the infinitesimal pipe, only the boundary conditions were replaced: On the first side, an inflow condition (Dirichlet velocity, Neumann pressure) describing the velocity profile out of the plate is used. On the outflow, a Neumann condition is applied to the velocity field, with a Dirichlet condition describing the pressure.

Here again, the data on the last plain (outflow plain) is written to a file and used as inflow condition for the counter-flow.

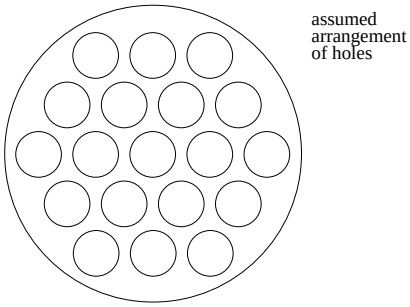

Figure 4.4: The perforated plate

To describe the flow through the perforated plate, a complicated velocity profile is forced onto the inflow boundary[5]:

First, each cell's distance to the closest hole is determined. If it is smaller than the holes radius, the cell is considered to be in a hole. Thus, the inflow velocity is copied onto this cell. Otherwise, the cell is on a wall, the velocity is set null. To arouse turbulence, some noise is added on cells located in holes.

As mentioned, a hole is no more represented as circular but rather as an assembly of quads (see Fig. 4.5). The effects of this distortion are not really known. As well, there is no reasonable velocity profile known to be applied to the holes (actually, top hat is used). Nevertheless, this approach should be valid since the focus is on the outflow of the pipe – where the information for the burner's nozzle is acquired.

The perforated plate consists of holes of 2 mm in radius, so far apart that a 45 percent transparency is realized. Since there is no information about

[5]The grid couldn't be warped to fit the holes since *flowsi* is not able to deal with distorted cells. Only Cartesian and cylindrical grids are allowed.

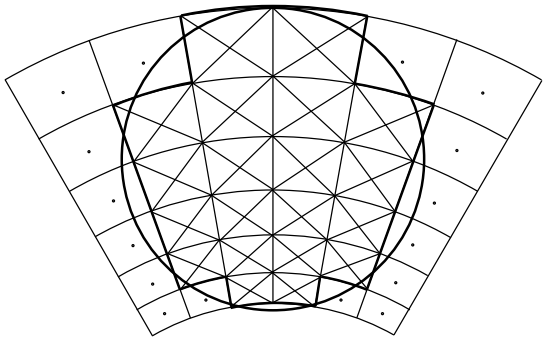

Figure 4.5: Degenerated circles

the hole's arrangement, the simple case of "closest package" of circles was assumed. Thus, the perforated plate looks like illustrated in Fig. 4.4.

To create appropriate inflow profiles, a fortran function `Sieb` (German word for *sieve*, see Appendix A) was written. This routine computes the distance of a cell's center to the very next hole. A simple velocity profile is applied to return a value proportional to the local inflow velocity. On walls `Sieb` will just return null. Thus, this function can not only return velocities. It can be used as well to check if a cell is located in a hole or not.

The grid represents a no-slip wall – resolving the boundary layer in laminar cases only. There is no modeling of the turbulent boundary layer near the plate. Nevertheless, this should not be too big a problem since the Reynolds Number in the hole is rather small. Further more, the area of interest is just the outflow of the pipe – far away of the inflow.
Thus, this simplification should not cause too big errors. Although there is no proof, this has to be assumed since there is no realistic way for modeling a turbulent boundary layer on the perforated plate.

4.4.4 Inflow replaced by Coupling

All previous approaches unidirectionally copy the velocities of a Neumann outflow to the counter configuration's inflow. Thus, the counter flow has no influence on the fields inside the inflow pipes. This is not realistic: The flow out of a nozzle strongly differs from that on an outflow boundary or inside a pipe.

To come free of this limitation, the inflow condition must be removed. This is possible only by creating one big flow-field, consisting of two inflow pipes and the counter flow domain. This can be done by coupling several *flowsi*-processes. Figure 4.6 illustrates this coupling which will be described in detail later on (see chapter 5).

This approach promises to describe the flow field in a very realistic way.

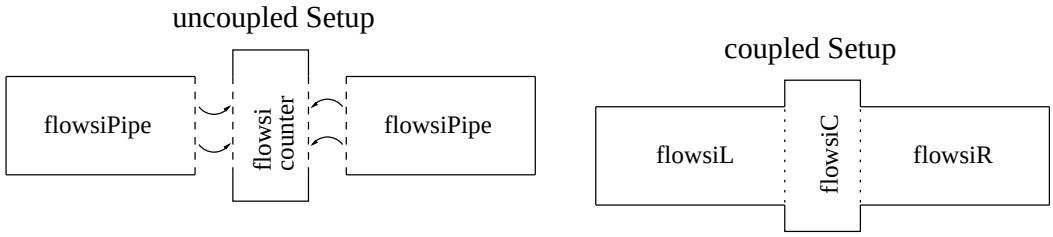

Figure 4.6: Coupling of domains

4.4.5 Outflow Conditions

Describing the characteristics of the outflow is far less complicated than those of the inflow.

For pressure, a Dirichlet condition is applied, forcing a reference value (i.e. 0) on the outflow surface. For all other scalars, a Neumann condition is used to make the radial derivatives disappear. Thus, there is only convective transport over the outflow.

Using these boundary conditions, re-flow can occur (i.e. fluid entering the domain over the outflow boundary). This results in instabilities. Nevertheless, a Neumann velocity condition can not inhibit this.

To avoid any re-flow, the condition applied to the velocity normal to the boundary was altered. If mass-flow is positive, the Neumann condition is used. Otherwise, the velocity is reset null. This stabilising condition is referred to as *"Neumann with Cut-Off"*.

"Cutting off" negative outflow velocities alters the results. Nevertheless, with the focus on the axis, this can be done. Furthermore, reflow seems not to be too realistic considering the flow field.

4.5 Post-processing

A numerical simulation produces huge amounts of data. In this case, a set of over 4 billion items is acquired. This data must be condensed to relevant information. Procedures performing this task are called *"post-processing"*.

Coming to *Large Eddy Simulations* or *Direct Numerical Simulations* of turbulent flows, a big part of post-processing is the calculation of mean values (*Reynolds* or *Favre* averaged) as well as the determination of relevant statistical correlations. These procedures render the acquired data into a useful form.

As well, it is very interesting to study the ''probability density function" (PDF) of the mixture fraction at a single point.

Finally, the data must be visualised. The following techniques were applied:

4.5.1 Visualisation in Two Dimensions of Space

Especially during the development phase, it is important to visualise velocity and pressure fields. This gives a better understanding of the physics and the numerics happening in the simulation. To generate iso-colour plots of scalars and vector-plots of velocities, the free *plotmtv* program was used. It reads data in a simple format and delivers reasonable output. A new routine was written to create the *mtv*-files showing the instantaneous velocity field.

4.5.2 Visualisation in One Dimension of Space

For comparing the computational results with experimental data, best suited are x/y plots. For this study, *gnuplot* was used.
The values along the counter-flow's centreline were plotted over the x-location to be compared to papers of Sardi and Mastorakos. All plots are centred at the stagnation point.

4.5.3 Visualisation of Time Dependency

To get a better understanding of the development in time, the values on the center point were monitored[6]. This data is used as well to compute the mixture fraction's PDF.

Finally, the monitored data gives early hints about the quality of the solution – simplifying the decision whether to kill non-promising runs or not.

[6]This point is not necessarily located on the mixing layer.

Chapter 5

Coupling and Parallelisation

It appeared desirable to compute the real outflow of the nozzle and not only use a pipe-flow as inflow condition. Thus, a way to compute the flow in the whole flow-field, including the pipes spending fuel and oxidant, had to be found.

Certainly, the LES of such a big domain requires great amounts of CPU-power. To reduce the time between setting up calculations and converged mean fields, multi processing becomes necessary. Thus, this coupling should be combined with a parallelisation as well.

To achieve this without spending too much time on coding, a very simple approach was developed. This chapter describes the solution found.

5.1 Coupling for the Counter-Flow

In previous computations, the data on the burner's inflow was acquired by *independent* computations of the pipes spending fuel and oxidant. The outflow values were written to a file and used as inflow condition for the burner. In this case, it is sufficient to just compute the inflow values once, considering the setup being almost symmetrical. Thus, two *flowsi* processes were necessary; "flowsiPipe" computing the inflows, "flowsiBurner" simulating the burner. Figure 4.6 symbolises this.

Opposed to unidirectional copying, full coupling enables pipes and burner to freely interfere. I.e. a fluctuation on the left pipe's inflow will influence

both the burner and the right pipe. Thus, three processes are running, two dealing with the pipes, one simulating the burner.

This coupling is realized by exchanging data between the three processes.

5.2 A Simple Method of Coupling separate Flow Fields

To couple separate flow-domains, a method is needed which introduces information from one domain into the solving procedure for the other domain. This is done by an approach which is very common for block-structured grids.

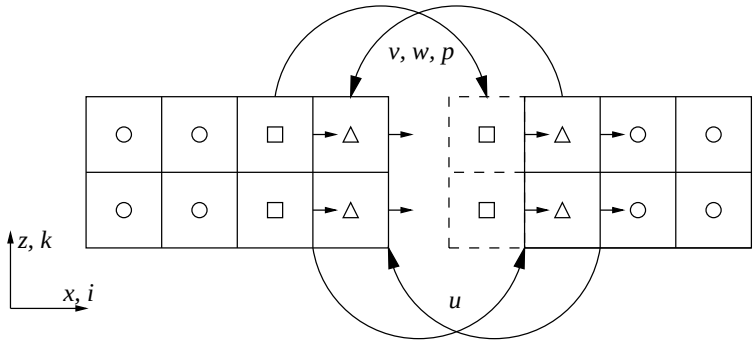

Figure 5.1: Data exchange between domains

To realize this data exchange, the routines describing the boundary conditions were altered: A new type of boundary condition was implemented. This "exchange" condition is quite similar to the existing periodic conditions – differing only by the fact that data is exchanged between two different blocks instead of two sides of the same block. Fig. 5.1 shows the way data is exchanged: Values from the last cell of the flow domain are copied to a "ghost-point" outside the other domain. (These boundary values are not computed. For example, when using a no-slip condition, the velocities on the ghost-point are set zero. When using a Neumann condition, the values of the last cells are copied to this ghost-point, resulting in a zero gradient on the outflow.)
I.e. the last relevant slice of the first domain is copied to a "ghost-slice" on the second domain – and vice versa.

This data exchange between the flow-domains is executed whenever a routine describing boundary conditions for velocities, pressure or scalars is called.

It is sufficient to exchange data only once per time-step, although the problem is rather elliptic. The reason for this is hidden in the CFL-condition:

To get a stable numerical scheme, the time-step width must be so small that no scalar is transported further than one cell per time-step. To assure this, *flowsi* adjusts it's time-step width to fulfil this limit (The Courant-Number is set 0.7). Thus, any relevant influence is transported from domain to domain only by exchanging the values of a single slice[1].

Though, this is only half the truth. This approach only resulted in a stable scheme when a strong under-relaxation[2] was applied (0.1 instead of 1.0), resulting in flow-fields only loosely coupled.

To solve this problem, C. P. Chou suggested the exchange of more than one slice, assuring that gradients (and higher derivatives) are transfered as well. For this study, three slices had to be exchanged to establish a coupling that is stable with no under-relaxation. Thus, the overlap between the domains is six cells.

5.3 Communication

After having found a way to introduce the other domains data into the solving procedure, a communication must be set up. To avoid errors, it was relied on a simple approach for data exchange via files: Each time *"process1"* calls a boundary-routine, it writes the relevant data to a file *"out1"*. Then it reads *"process2"*'s output (*"out2"*), containing the input data for *"process1"*. Finally, *"process1"* removes *"out2"*.

[1] This is not true for the pressure since an approach for incompressible fluids was used. Nevertheless, the mean pressure is almost constant.

[2] This means that instead of exchanging the values, the new value of a cell is formed by calculating an weighted average between it's old value and the value that *should* be the new value!

5.3.1 Avoiding Errors

To avoid reading non existent files, each process must wait until it's input file exists – created by the other process. As well, each process writing to a new file has to wait until this file was deleted by the other process. If the file doesn't exist (or was not deleted) yet, the process will loop, checking if the file exists (or was deleted) yet.

This proceeding would result in a working communication, if all files were written entirely before the scheduler passes the CPU to another process. Actually, a process may start reading a file before writing was finished – causing read-errors. To avoid this problem, *"process1"* will create a *"OK1"* file of size 0, it's existence signalling *"process2"* to start reading now. With this "handshake", a stable communication is set up.

Finally, care was taken to avoid any "deadlocks".

Nevertheless, this communication is still inefficient, with "busy" waiting being just the worst of problems:

5.3.2 Reducing the Impact of Busy Waiting

Very often, a situation will occur when a process waits to read data, although the file does not exist yet. Thus, the processor has to wait until this file exists. Especially on machines with more processes than processors running, this is fatal: Not bad enough one processor is rendered useless by having to wait for results, this very processor might be assigned to the waiting process, permanently checking if the file was written yet. Meanwhile, the process generating the output might be sleeping, since the scheduler assigned the CPU to the *busy waiting* process instead of to the calculating one!

This waste of time gets worse with each process waiting busy. The coupling of only two domains can well result in 90 percent of time spent on busy waiting on a single-processor machine.

A simple solution for this problem is suggested in [11]: Whenever a process is waiting, it is made sleep for at least one micro second, giving the scheduler a chance to reassign the CPU to another process. Thus, the other process can finish computing and writing. A process waiting like this consumes less than 1 percent of the systems power.

For this study, the simple C-routine `wait_()` was linked to the code to make a process sleep for (at least) a micro-second:

```
#include<unistd.h>

wait_()
{
  usleep(1);
}
```

The use of this `usleep` command greatly improves the coupled codes performance – speed ups of factor ten have been observed!

5.3.3 Further Reducing Communication Time

A simple way to reduce the communication time is using a ram-disk instead of a real hard-drive:

Part of the physical RAM is addressed just like a hard-disk, combining an easy to use file-system with the great bandwidth of system RAM. [3] Thus, the data exchange is not much slower than the use of shared memory or common message passing.

To further reduce the communication time, an analysis of the code could deliver information which communication is not necessary at all or which data could be exchanged together, reducing the time spent on waiting for files and creating them. Nevertheless, this was not realized since this analysis had taken too much time for a "simple investigation" of a counter-flow burner.

[3] This is just the opposite of hard-disk-paging – reducing the available amount of RAM to speed up the system.

Chapter 6

Results

This section presents the acquired results. First, of all, a comparison of the available boundary conditions is performed by means of Mastorakos' case. Then, some results for Sardi's case are presented. Finally, certain features of the results for Sardi and Mastorakos are discussed - considering experimental data as well.

The first image, Fig. 6.1, presents a snapshot of the velocity-field. This is well suited to get a basic understanding of the flow field. The plot clearly shows some fluctuation on the center plain, right in the mixing layer.

6.1 Mastorakos' Case

Mastorakos Case is the more difficult to simulate since its Reynolds number ($Re = 2,640\,;\, Re^* = 157$) is too low to assume fully developed turbulence. Thus, this case is the more suited one to estimate the boundary condition's quality.

The first plot (Fig. 6.2) shows the mixture fractions development in time. The data was gained by saving the values on a monitoring point for each time-step. This very point is located in the center of the counter-flow, almost in the stagnation point.

The mixture fraction is plotted in the interval between 0.15 s and 0.3 s. The startup phase is not presented as it strongly depends on initial conditions – which are not relevant for burner-like configurations.

These plots give a good impression of turbulent fluctuation in the flowfield's center.

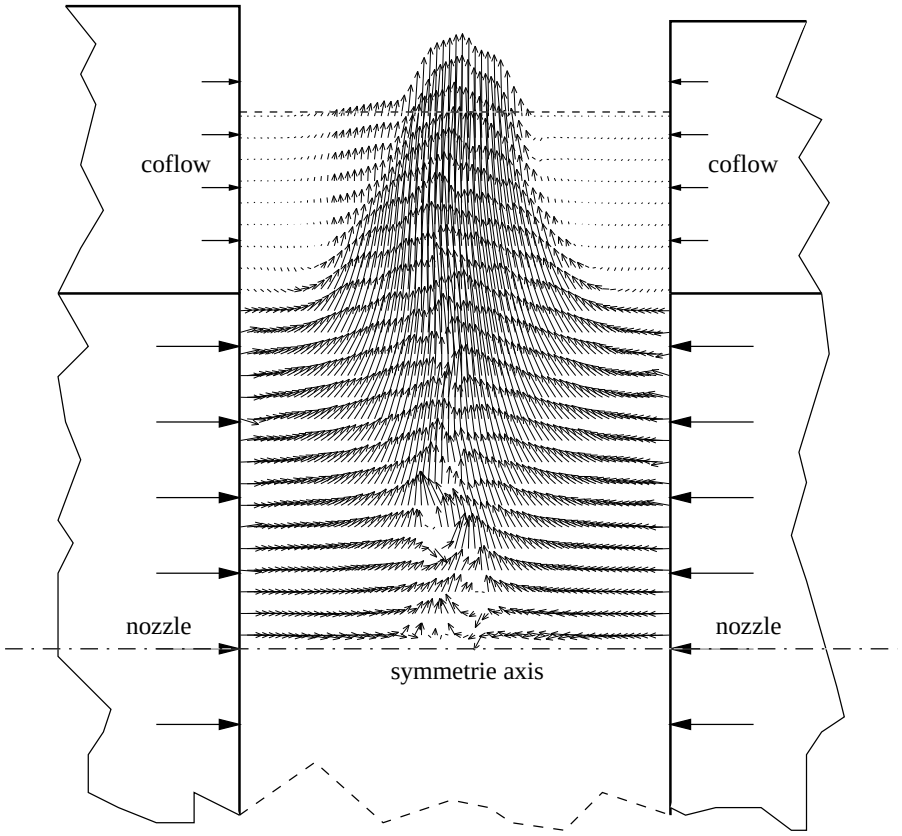

Figure 6.1: Snapshot of velocity field

For the simple random noise case, one finds a very smooth curve having developed. This shows that only large structures exist in the center of the domain. Opposed to this, with pipe or plate conditions, far smaller structures are predicted, resulting in strong high-frequent fluctuation. An attempt to explain this difference will be made later.

The axial velocities on the domains axis are presented in Fig. 6.3. For all

boundary conditions, the velocities were normalised with the velocity on the center of each outflow, resulting in a maximal non-dimensional velocity of 1. This was necessary since Mastorakos [4] apparently did the same – without giving any information about the velocity he used for normalisation.

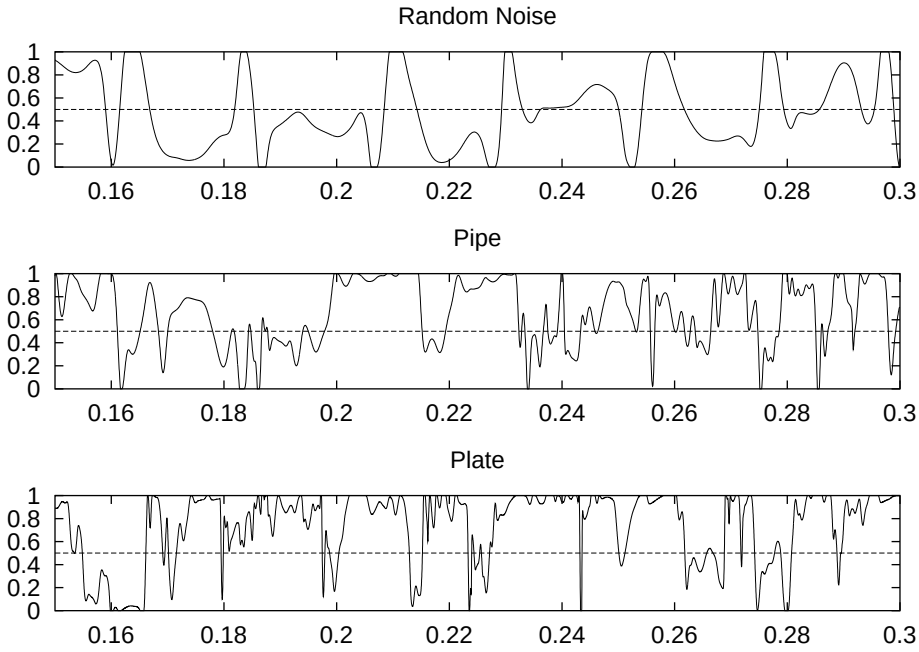

Figure 6.2: Development of the Mixture Fraction in Time

In this case, the noise condition best matches the experimental data. The most complicated condition, computed by simulating a perforated plate, results in the strongest deviations of the measurements. Nevertheless, all results are close to the experimental data.

It strikes that the measured results seem to be asymmetric. This can be explained by the difference in the fluid's density, with one pipe being heated. This is done to find the mixture fraction by simply measuring the temperature.

The next image (Fig. 6.4) shows the mean mixture fraction plotted along the axis. Although the velocity field seems to be computed better when using the simple noise condition, the mixture fraction is predicted best with inflow data acquired by simulating the flow through the plate.

The noise condition results in too thin mixing layers, confirming that too

few turbulence is predicted for this case. The pipe condition's results are better, although insufficient mixing is predicted as well. Only the perforated plate seems to force realistic turbulence.

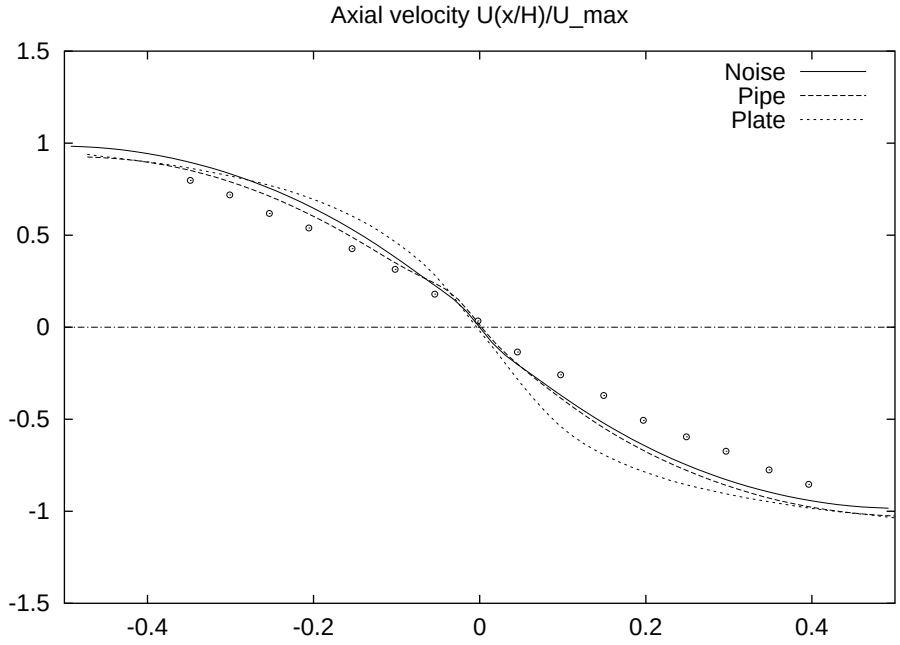

Figure 6.3: Axial velocities

Figure 6.5 presents the fluctuations of the mixture fraction. Its RMS [1] is plotted along the axis

One finds that in all cases, the peak value of the fluctuations, located in the stagnation point, is overestimated by at least 15 percent. Using the complex plate condition, the highest peak values are observed. With the random noise condition, the lowest values are computed, matching best to the measured peak.

Nevertheless, the curve's shape looks best when the plate-condition is applied. The pipe-condition delivers a too slim curve and the noise condition just results in a peak. This is consistent with the too thin mixing layers predicted when less sophisticated boundary conditions are used.

Figure 6.6 shows the axial velocities fluctuation. The RMS of this velocity is presented along the axis.

[1] Root Mean Square, $\mathrm{RMS}(\phi) = \sqrt{\overline{\phi'\phi'}}$

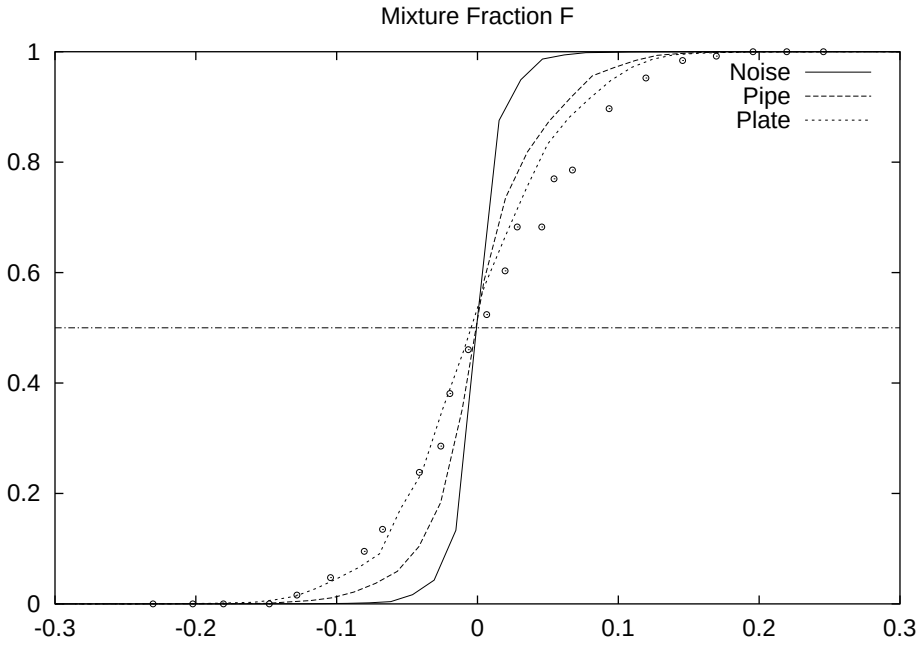

Figure 6.4: Mean values of the mixture fraction

This plot is particularly interesting, giving a possible explanation for too weak mixing in cases, when only noise is applied:

In the noise case, random noise is added to the inflow velocities. These fluctuations do not represent realistic turbulence. They disturb the flow field and excite oscillations on certain instable scales. The turbulent kinetic energy added on stable scales is just dissipated. Since noise adds too much turbulent energy on small scales (dissipation) and too few on larger scales (production and transportation), the rate of dissipation at the boundary is too high. Thus, only few turbulent kinetic energy can reach the domain's center. This becomes obvious in the plot for the noise case: Close to the inflows, the fluctuation level drops rapidly.
This is different when using the pipe-condition. Since a realistic spectrum is applied, the dissipation remains on a reasonably low rate.
Using the perforated plate, the level of turbulence on the inflow is even higher. This is due to the perforated plate producing stronger turbulence than an infinitesimal pipe.

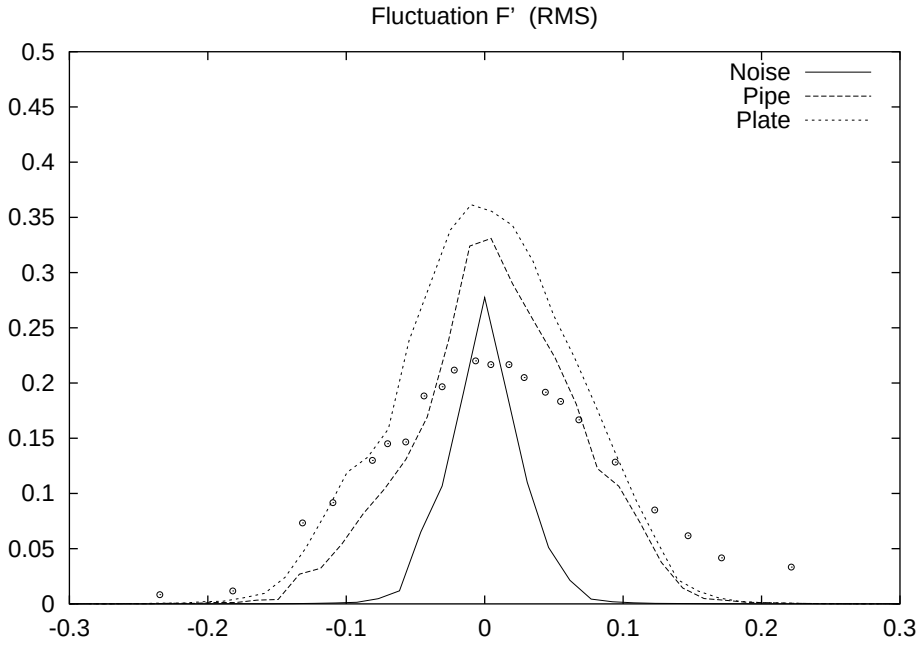

Figure 6.5: Fluctuation of mixture fraction

Finally, the PDFs[2] for the mixture fraction are presented. Figure 6.7 proves that the noise condition is not sufficient to compute good results. It strongly overestimates the probability for the mean values. On the other hand, the PDF predicted by means of the perforated plate fits best. The bi-modality is well resolved. Though, the computed PDFs are wider than the measured one: The probability of pure mixtures is predicted far higher than measured. This explains as well why the mixture fraction's fluctuation level (see Fig. 6.5) is overestimated for all boundary conditions.

This comparison was done to find the best suited boundary conditions. When developing the boundary conditions, it was hoped that the more complicated approaches were able to better describe reality, leading to improved results. This proved to be true, the results got better with the evolution of the boundary condition:

When only applying noise, the mixing layer is predicted too thin due to the unrealistic turbulent spectrum applied. – Using a pipe-flow features real turbulence, but not the turbulence existing behind a perforated plate. –

[2]Probability Density Function

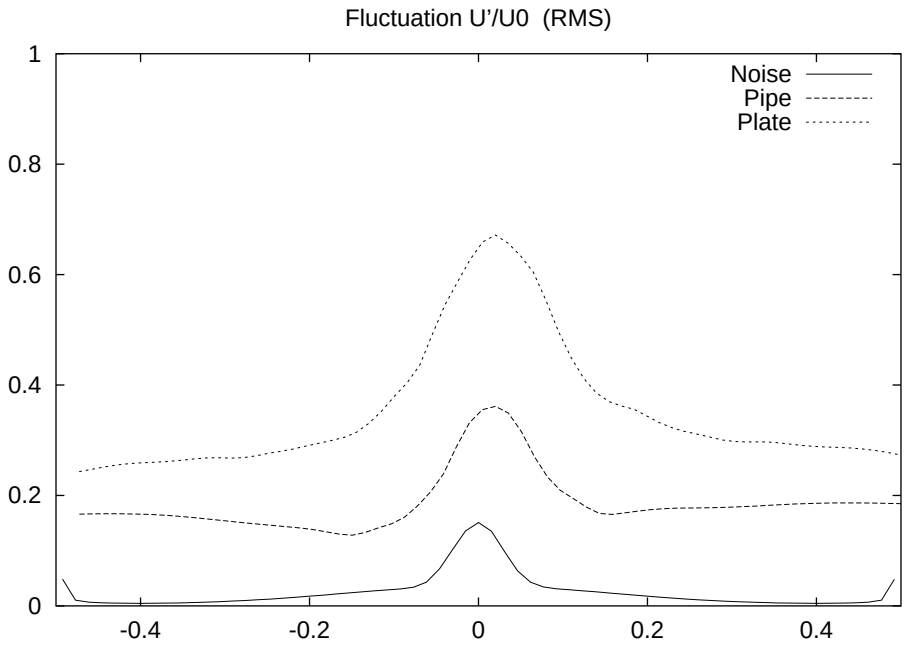

Figure 6.6: Fluctuations of axial velocity

Finally, the perforated plate was simulated, giving a good description of the turbulence entering the computational domain.

In this context, it shall be mentioned that no parameter or constant was altered to achieve these results. Only the boundary conditions were adapted to describe Mastorakos' counter-flow configuration. The computation was done with the same code previously used to simulate pipe-flows or jets – with good success.

Thus, the LES approach implemented in *flowsi* seems to offer good predictability!

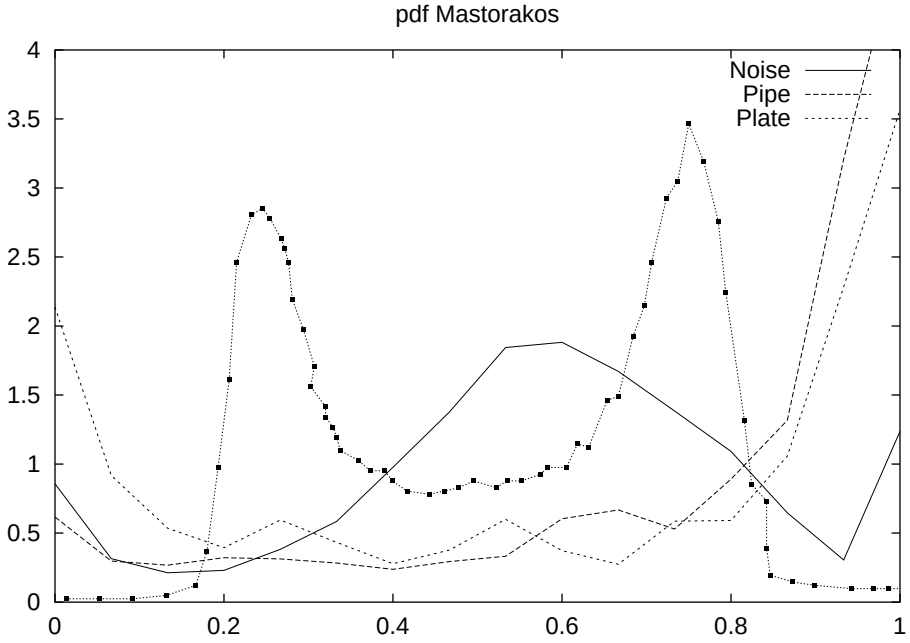

Figure 6.7: PDFs of the mixture fraction

6.2 Sardi's Case

It was found that simulating the flow through the perforated plate is the best available description for the inflow. Thus, only results achieved with this setup are presented here.

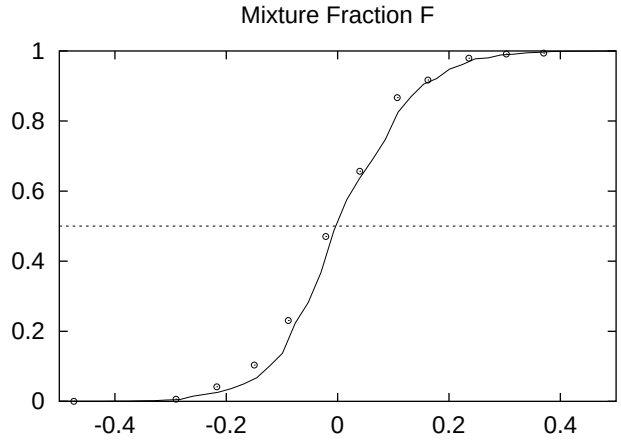

Figure 6.8: Mean value of the mixture fraction

First, the computed mixture fraction is compared to experimental data in Fig. 6.8. One clearly finds the good match between that data. The biggest difference results from a small left-shift of the experiment's mixing layer.

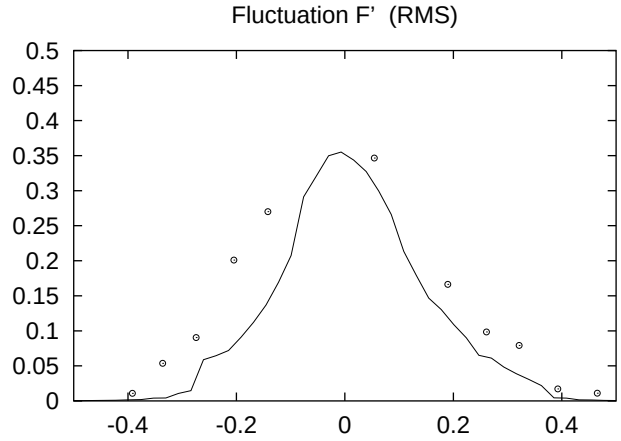

Figure 6.9: Fluctuation of mixture fraction

The next plot (Fig. 6.9) shows the mixture fractions RMS. Here again, the simulation conforms to reality.

Finally, the PDF computed for the flow field's center is presented (Fig. 6.10). Unfortunately, the monitoring point seems not to be located on the mean stagnation point, making the PDF asymmetric. This possible since only few time steps (23,000) were computed[3]. Figure 6.2 clearly shows that for reasonable statistics, a longer time must be regarded.

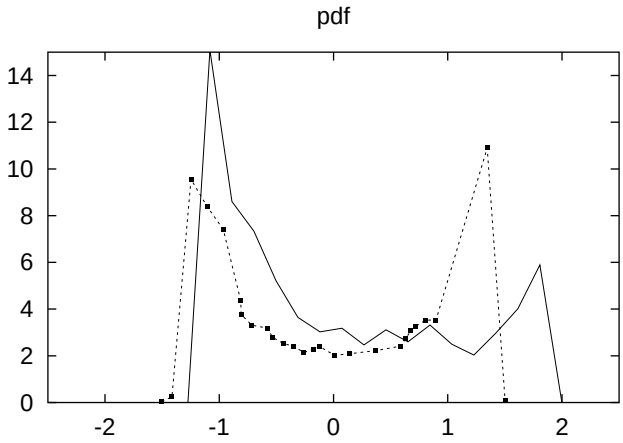

Figure 6.10: PDF of the mixture fraction

Although the computed PDF looks similar to the measured one, a real comparison is impossible: In her paper [5], Sardi only presents PDFs normalised with variance and mean value. Thus, there is no information about neither mean value nor variance, rendering any reasonable comparison impossible. Even worse, the integrals over her PDFs are no more unity. To make the PDFs comparable, the predicted PDF was scaled to match the one provided by Sardi.

[3]The reason for not computing more time-steps is the same as always: Saving CPU-time!

6.3 Comparison of Sardi and Mastorakos

This section compares the setups of Sardi and Mastorakos – considering both experimental and numerical[4] data. Although very similar configurations were investigated, some results strongly vary.

Mastorakos studied a low-Reynolds case, the pipe's Reynolds-number just being over-critical (2,640). Sardi, on the other hand, uses almost three times this much ($Re_{Sardi} = 7,500$).

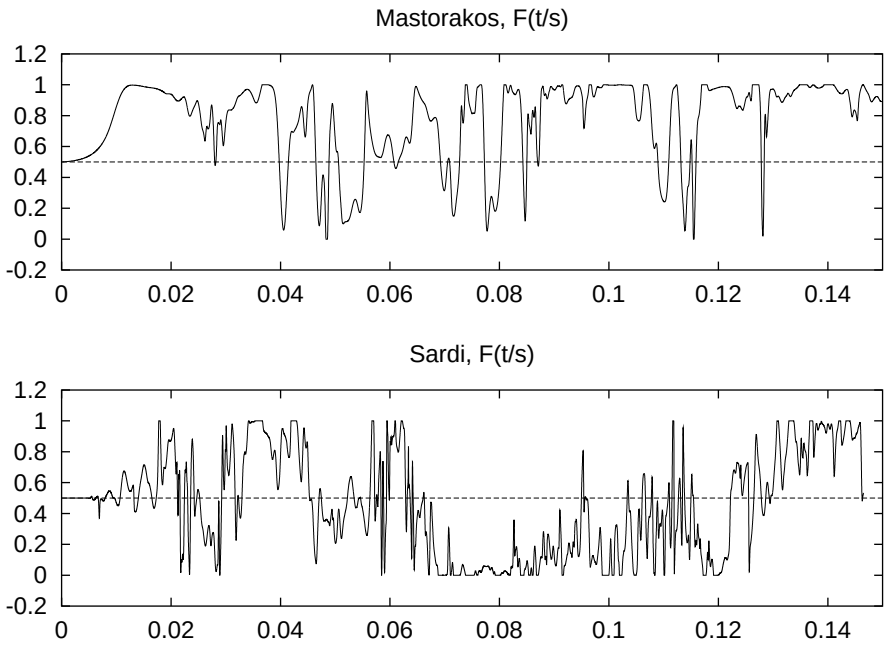

Figure 6.11: Mixture Fraction in Time

Figure 6.11 compares the transient development of mixture fraction for Sardi and Mastorakos. It is only presented to 0.15 s for in Sardi's case, a smaller time-step width is necessary due to a more restrictious CFL-limit. For Sardi's case, one finds the amplitudes of higher frequencies to be much bigger than for Mastorakos.

The next plot, Fig. 6.3 presents the mixture fractions. The calculations and measurements deliver very similar data.

[4]For this comparison, only data computed with the simulation of the perforated plate is used.

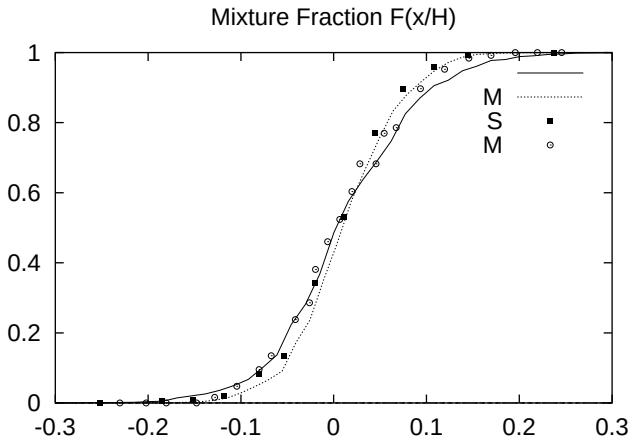

Figure 6.12: Mixture Fraction in Space

Finally, the plots for the mixture fraction's fluctuation are shown (Fig.6.13). According theory, all curves should be identical. There should not be deviations between the results of Sardi or Mastorakos. The computational results (for both Sardi's and Mastorakos' cases) are almost identical to the experimental data gained by Sardi. On the other hand, Mastorakos measurements resulted in values approximately 30 percent lower. This is astonishing since classical theory *and* LES-data predict the results for both setups to be identical.

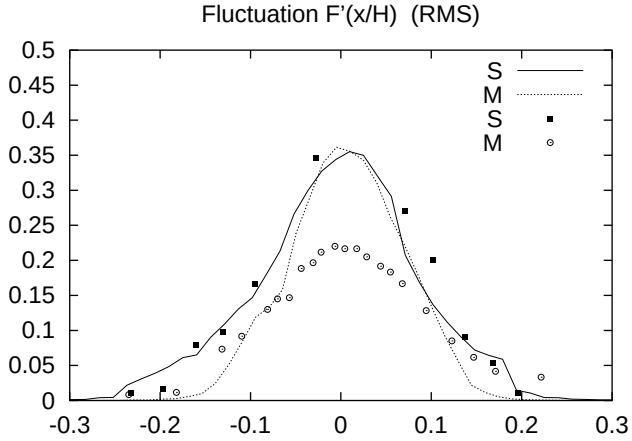

Figure 6.13: Fluctuation of Mixture Fraction

Chapter 7

Conclusion and Future Work

In this work, a 3d-LES was applied to study flow phenomena in the mixing layer of a counter-flow arrangement:

The *flowsi* code was adapted to this geometry. New boundary conditions were developed to improve the prediction. Finally, an approach for both parallelisation and coupling of flow-domains was developed and implemented.

The results achieved are in good agreement to the experimental data acquired by Sardi [5] and Mastorakos [4].

The differences resulting from modification of boundary conditions show the biggest problem of LES: Finding realistic turbulent boundary conditions.
Nevertheless, the results proof that with good boundary conditions, an LES can deliver satisfying results. With the perforated-plate condition applied, results could probably be improved just using finer grids.

It shall be repeated that these results were achieved without modifying any model-parameters. No changes in the flow solver were necessary. Except of the boundary conditions, the same code was used to simulate pipe and jet flows – with good success. This confirms that an LES (with reasonable boundary conditions) can offer good predictability!
When using the perforated plate condition, no experimental data on turbulence is necessary to determine the boundary conditions. Thus, these results can be obtained by "pure CFD"!

Although promising results were achieved, a lot of research is to be done on counter-flow burners (besides chemical reactive flows). Considering results presented previously, there is quite some potential for improving the mixture

fractions PDF. Nevertheless, this is only possible with more experimental data available. Especially in Sardi's paper, the normalisation of the mixture fractions PDF removes any information on mean values or variances.
To improve the computed PDFs, possible approaches are the use of finer grids, longer time for averaging, and most interesting, simulating a coupled flow-field. Of these suggestions, only the last is able to improve on the flow out of the nozzle. Considering the improvements resulting from better boundary conditions, full coupling seems to be the way to go!

Appendix A

Advanced Boundary Conditions

The computation of the inflow conditions is not part of the main simulation. Nevertheless, it is vital. Thus, the treatment of the inflow pipes has to be discussed.

A.1 Determination of Wall Reynolds Number

Since all pipe flows are similar, there is only one free parameter – the Reynolds number Re. Since the *flowsi* code does pipe computations in non dimensional values, the Reynolds number Re^* based on the *friction velocity* u^* is to be used.

According to Spurk [7], it can be determined by the following implicit equation:

$$\frac{1}{\sqrt{\lambda}} = 2.03 \lg\left(Re\sqrt{\lambda}\right) - 0.8 \tag{A.1}$$

The so called friction number $\lambda = 8\frac{u_*^2}{\overline{U}^2}$ compares the friction velocity u^* to the pipes bulk velocity $\overline{U}$.

But certainly, these equations are only a simplified description. The real dependencies are far more complicated. Thus, more realistic Re^* were found by doing a parameter study. The number actually used is referred to as Re^*_u in table 3.2.

A.2 Inflow Condition for Pipe

To model the flow through the plate, a complicated velocity profile was forced onto the pipes inflow.

To create appropriate inflow profiles, a fortran function `Sieb` (German word for *sieve*) was written. This routine computes the distance of a cell's center to the very next hole. A simple velocity profile is applied to return a value proportional to the local inflow velocity. On walls `Sieb` will just return null. Thus, this function can not only return velocities. It can be used as well to check if a cell is located in a hole or not.

The function `Sieb` is defined as follows:

```
      REAL FUNCTION Sieb(j0, k0, jma, kma)
************************************************************************
*     Andreas Kempf, 12.Juli.1999                                      *
************************************************************************
      INTEGER      j0, k0, jma, kma, l1, l2
      REAL         x10, x20, xl1, xl2, r, phi, Dist, nDist
      REAL         PPi, PRp, PRh, PD
      PARAMETER    (PPi = 3.1415927, PRp = 12.7,
     &              PRh = 2., PD = 5.025)
*            PPi ParameterPI
*            PRp ParameterRadiusPipe
*            PRh ParameterRadiasHoles
*            PD  ParameterDistance (of holes)


* determine cell-position in cartesian coordinates (x10,x20)
      phi = 2.*PPi*REAL(j0-1)/REAL(jma)
      r = PRp*REAL(k0-1)/REAL(kma)-(0.5*PRp/REAL(kma))
      x10 = r*cos(phi)
      x20 = r*sin(phi)

* Determine distance to hole
*     Dist just bigger than any hole diameter!
      Dist = PRp
```

```
      DO 2 l1 = -4, 4
         DO 1 l2 = -3, 3
*           position of hole
            xl1 = REAL(l1)*PD + 0.5*REAL(l2)*PD
            xl2 = REAL(l2)*PD * sin(PPi/3.0)
*           distance to hole
            nDist = ((x10-xl1)**2 + (x20-xl2)**2) **0.5
            Dist = MIN(Dist, nDist)
 1       CONTINUE
 2    CONTINUE

* Determine Sieb as a function of "Dist"
      IF (Dist.LE.(PRh/2.)) THEN
         Sieb = 1.
      ELSEIF ((Dist.LE.PRh).AND.(Dist.GE.(PRh/2.))) THEN
         Sieb = 1. - (Dist - 0.5*PRh)
      ELSE
         Sieb = 0.
      ENDIF

      RETURN
*************** END OF 'Sieb' ***************************************
      END
```

A.3 Outflow Condition for Pipe

As outflow condition, a simple Neumann condition was applied for all scalars but pressure, which is just forced zero.

Appendix B

Modified Subroutines

This chapter gives an overview on all relevant changes to the existing code. They were necessary to adapt *flowsi* to the new geometry, to enable a coupling between different jobs, and to adjust the evaluation routines to the data relevant for this setup. The following sections only describe the most important changes.

B.1 Modifications for Pipe-Flow with Wall Law

To deal with the turbulence created by the perforated plate, a new setup was created. This setup bases on a pipe-configuration with periodic boundary – describing fully developed turbulent pipe-flow. To make this setup usable for these simulations, the periodic boundary conditions were replaced. On the inflow, a Dirichlet condition forces proper velocities, a zero gradient condition is applied for the pressure. On the outflow, a Neumann condition is applied to the velocities, a Dirichlet condition to the pressure. Therefore, the following changes were necessary:

B.1.1 Initialisation

When a Dirichlet inflow condition is applied, the mean values on the corresponding cells are set during initialisation. They remain untouched later. It

is in the `IniPip` routine where the inflow profile simulating the perforated plate has to be set. This is done by calling `Sieb`, returning the proper values for each cell.

```
      DO 2 j = 1, Jmap2
         DO 1 k = 1, Kmap2
            Uinflw(j, k) = C*Sieb(j, k, Jmap2, Kmap2)
 1       CONTINUE
 2    CONTINUE
```

B.1.2 Boundary Treatment

The new boundary condition type in x-direction was newly implemented. Nevertheless, the condition corresponds to the "`(Lsbx .EQ. 3)`" setting, so that many subroutines didn't need a change for the new value of `Lsbx`. Only many `IF`-Statements were altered to call the code segment with `(Lsbx .EQ. 3)` as well.

The pressure boundary condition was changed to a Neumann condition at the inflow and to a Dirichlet condition on the outflow. This is done by the following lines of `BDPall`:

```
         DO 160, k = Kpfi, Kpla
            DO 150, j = Jpfi, Jpla
               P(j, k, Ipfim1) = P(j, k, Ipfi)
               P(j, k, Iplap1) = 0.0
 150       CONTINUE
 160    CONTINUE
```

The inflow velocity condition was changed to force the proper profile onto the first x-plain. This happens in the routine `BDu1x`. On the outflow plain, a Neumann velocity condition with cut-off (no reflow) is applied:

```
         DO 2, k = 1, Kmap2
            DO 1, j = 1, Jmap2
               U(j, k, Iulap1) = MA(XU(j, k, Iula), 0.0)
 1          CONTINUE
 2       CONTINUE
```

B.1.3 Main Cycle

Here again, some `IF`-Statements had to be changed to make code segments still being executed when `Lsbx` is set to "3" instead of "1".

B.2 Modifications for Counterflow

The most serious modifications were necessary to create a setup for the counterflow. To describe it's flow-field, Hendrik Forkel's jet setup was chosen as a basis for further development. This task included replacing the outflow boundary by an inflow on the right hand side of the cylinder as well as implementing a Neumann condition for the velocities on the cylinder's shell. The necessary changes are described below.

B.2.1 Treatment of Boundarys

This routine contains the most important changes again. Since the former outflow of the computational domain was changed into a second inflow, many changes were necessary on this plain. This means switching to Dirichlet conditions for scalars and velocities (in `BDSclx`, `BDuvwx`, `BDu1x` and `BDu2x`). For the pressure, a Neumann condition became necessary (`BDPall`).

B.2.2 Evaluation

Some additional data was written and some changes were made to assure the evaluation code (developed for jets) will not crash when dealing with the modified setup.

B.3 Modifications for Pipe-flow with Dimensional Values

Realizing that simple copying of outflow data to the inflow might not be sufficient, the challenge of computing the flow in the hole configuration was accepted. Since it is not possible to couple the non-dimensional pipe-flow

code to the counterflow code, a new pipe-setup must be developed. It bases on Hendrik Forkel's Jet case. The inflow condition is adapted to represent the flow through the perforated plate. Instead of an outflow on the surface of the cylinder, a no slip condition is applied. Thus, the wall is no more represented by a wall function. On the other hand, this is not necessarily a disadvantage: The Reynolds number in the nozzle is low (2,700 for Mastorakos A) and the piece of the pipe between the plate and the nozzle is short. Thus, the computation happens in an area of transition – where even a wall function is not capable to describe the boundary layer.

These changes are not described any further since they are very similar to those applied in previous sections.

Appendix C

Coupling

The data transfer is performed in the routines describing the boundary conditions. When such a routine is called, data is written to a file and read from another one. This results in over one-hundred file-creations per time step. Nevertheless, the simplicity of this approach made it "all in all efficient".

The actual data transfer is done by the subroutine `UTFsCp`. (The routine presented here is the one used for the left inflow-pipe.)

This routine checks if no old file exists and writes it's data. Then it waits until an input file exists, reads the data and deletes the file.

The additional `*OK`-files are necessary to assure that data files were written *entirely* before another *flowsi* attempts to read them.

```
      SUBROUTINE UTFsCp(n, in, out)
*****************************************************************
***   Do periodic boundary-conditions between different flowsi-jobs  ***
***-------------------------------------------------------------***
***   Author:           Andreas Kempf                           ***
***   Developed:        29.07.1999                              ***
***   Last change:      12.09.1999                              ***
*****************************************************************
*-------------- arguments ----------------------------------------*
      INTEGER           n
      REAL              in(*), out(*)
*-------------- local variables ----------------------------------*
      INTEGER           index
      LOGICAL           Lexist
      REAL              dummy
```

```
*-------------- called functions/subroutines --------------------------*

*-------------- common blocks -----------------------------------------*
      INCLUDE           'pnetin.inc'
      INCLUDE           'cinflow.inc'

*************** BEGIN OF CODE ***********************************************

* -- write out to L ----- ---------------------------------------------
 10   CONTINUE
      INQUIRE (FILE='../LOK', EXIST=Lexist)
      IF (.NOT.Lexist) THEN
         OPEN (21, ERR=101, FORM='UNFORMATTED',
     &        FILE='../L', STATUS='NEW')
*
         DO 20 index=1, n
            WRITE (21, ERR=102) out(index)
 20      CONTINUE
         CLOSE (21, ERR=103)
         OPEN (31, FILE='../LOK', STATUS='NEW')
         CLOSE (31)
      ELSE
         CALL wait()
         goto 10
      ENDIF
      print *, 'L written, LOK created'

* -- read in from CL ------------------------------------------------
 30   CONTINUE
      INQUIRE (FILE='../CLOK', EXIST=Lexist)
      IF (Lexist) THEN
         OPEN (21, ERR=104, FORM='UNFORMATTED',
     &       FILE='../CL', STATUS='OLD')
         DO 40 index=1, n
            READ (21, ERR=105) dummy
*           underrelaxation:
            in(index) = 0.9*in(index) + (1 - 0.9)*dummy
 40      CONTINUE
         CLOSE (21, ERR=106, STATUS='DELETE')
         OPEN (31, FILE='../CLOK', STATUS='OLD')
         CLOSE (31, STATUS='DELETE')
      ELSE
         CALL wait()
         goto 30
      ENDIF
      print *, 'CL read, CLOK, CL removed'
```

```
      RETURN

* == Error messages ================================================
 101  CONTINUE
      print *, 'Error opening output-file'
      STOP
 102  CONTINUE
      print *, 'Error writing output-file'
      STOP
 103  CONTINUE
      print *, 'Error closing output-file'
      STOP
 104  CONTINUE
      print *, 'Error opening input-file'
      STOP
 105  CONTINUE
      print *, 'Error reading input-file'
      STOP
 106  CONTINUE
      print *, 'Error closing input-file'
      STOP
*************** END OF 'UTFsCp' ***************************************
      END
```

Bibliography

[1] B. Wagner, *Einfuehrung in die numerischen Methoden der Aerodynamik*, Skript zur Vorlesung an der TU-Darmstadt, 1996

[2] J. Janicka, *Turbulenz*, Skript zur Vorlesung an der TU-Darmstadt, 1998

[3] J. Janicka, *Verbrennungstechnik*, Skript zur Vorlesung an der TU-Darmstadt, 1998

[4] E. Mastorakos, A. Taylor, J. Whitelaw, *Mixing in Turbulent Opposed Jet Flows*, Ninth Symplosium on "Turbulent Shear Flows", Kyoto, Japan, 1993

[5] K. Sardi, A. Taylor, J. Whitelaw, *Conditional scalar dissipation statistics in a turbulent counterflow*, Journal on Fluid Mechanics, 1998, vol. 361, pp. 1-24

[6] H. Forkel, *Ueber die Grobstruktursimulation turbulenter Wasserstoff-Diffussionsflammen*, Dissertation, Fachbereich Maschinenbau, Technische Universität Darmstadt, 1999

[7] J. Spurk, *Stroemungslehre: Einfuehrung in die Theorie der Stroemungen*, Berlin: Springer 1996

[8] L. Schmitt, *Numerische Simulation turbulenter Grenzschichten*, (Large-Eddy-Simulation) Teil 1, Bericht 82/2, Lehrstuhl für Strömungsmechanik, Technische Universität München, 1982

[9] F. Unger, *Numerische Simulation turbulenter Rohrströmungen*, Dissertation, Lehrstuhl fuer Fluidmechanik, Technische Universität München, 1993

[10] C. Weinberger, *Large-Eddy-Simulation von reagierenden und nicht-reagierenden turbulenten ebenen Freistrahlen*, Dissertation, Fachbereich Maschinenbau, Technische Hochschule Darmstadt, Darmstadt, 1997

[11] Hank Dietz, *The Linux Parallel Processing Howto*, available on the "World Wide Web" at:
`yara.ecn.purdue.edu/~pplinux/PPHOWTO/pphowto.html#toc1`