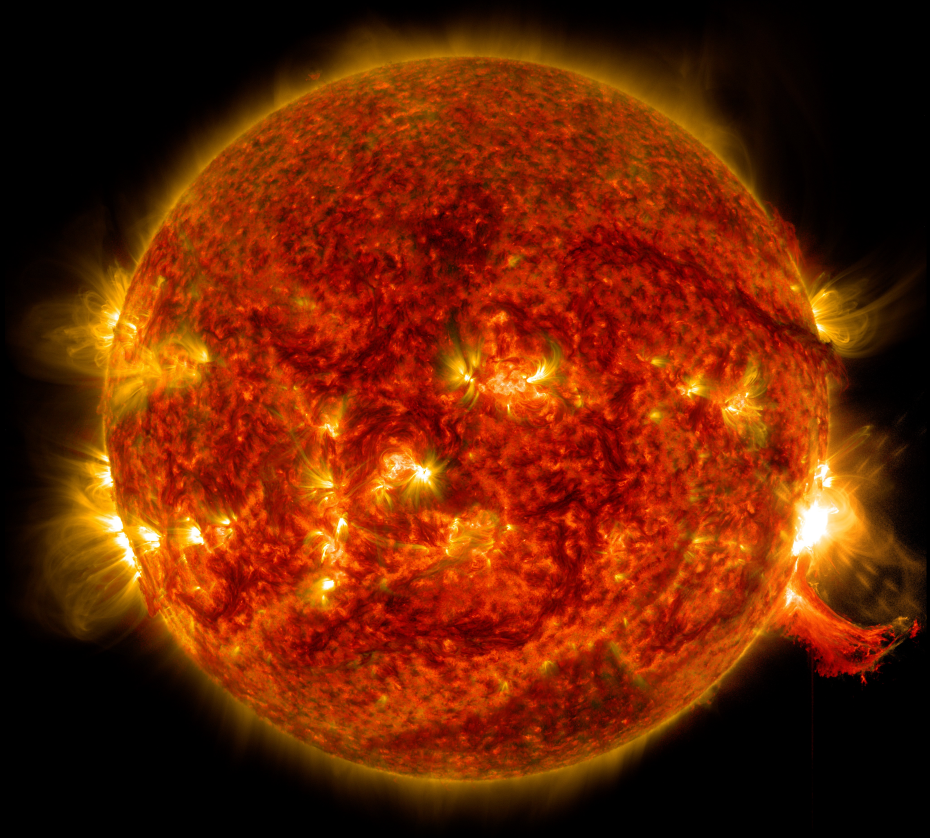
Boundary layer flashback prediction for low emissions full hydrogen gas turbine burners using flow simulation

O.H. Björnsson



Technische Universiteit Delft

TU Delft

Boundary layer flashback prediction for low emissions full hydrogen gas turbine burners using flow simulation

by

O.H. Björnsson

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 13, 2019 at 13:30.

Student number:	4743903	
Project duration:	December 2018 – September 2019	
Thesis committee:	Ir. L. Altenburg,	TU Delft
	Prof.dr.ir. S.A. Klein,	TU Delft, Supervisor & Chairman
	Dr. D. Lahaye,	TU Delft
	Dr.ir. R. Pecnik,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Cover image: The sun, our primary source of energy, fusing hydrogen. Captured by NASA's Solar Dynamics Observatory on Oct. 2, 2014.

Abstract

Flame flashback from the designated flame holding volume is an intrinsic problem in the design of lean premixed combustion systems. Due to higher flame speeds and lower quenching distances, hydrogen-rich fuel mixtures are especially prone to boundary layer flashback (BLF), where mixture flow speeds are low compared to the bulk flow. Research at TU Munich (Eichler 2011, Baumgartner 2014) has resulted in new insights into the mechanism of BLF, revealing a strong coupling between the flame and the flow field, and challenging the established critical gradient model. This led to the development of a new BLF model (Hoferichter 2017) for flames confined in a horizontal channel burner, validated for atmospheric conditions and built on the observation that BLF is triggered by flow separation at the flame front. A previous TU Delft student (Tober 2019) has suggested two modifications to this model, with one being to include the effect of Lewis number instabilities leading to the formation of cellular flame structures increasing the turbulent flame speed of lean hydrogen-air mixtures. With these modifications the model is now also validated for preheated mixtures.

In this thesis, the model is further investigated with the goal of applying it to more complex burner designs. A new way to apply Stratford's turbulent boundary layer separation criterion (originally derived for boundary layers growing on airfoils) for flame induced flow separation is proposed and validated. This "generalized" criterion results in more realistic values for the computed pressure difference over the turbulent flame front. The effect of flame stretch and the Markstein length on the laminar flame speed and subsequently on flashback limits is then investigated and found to be of secondary importance compared to the Lewis number correction mentioned above. Using an unstretched laminar flame speed in the turbulent flame speed closure reduces the complexity of the model and results in better predictions for very lean mixtures.

The BLF model is then modified to use CFD results for inert burner flows as input. This is validated for flames confined in horizontal channels and then applied to flames confined in diverging burners with underlying adverse pressure gradients. First, a comparison of turbulence models is made with regards to their performance for diffuser flow. Then an automatic method to customize the generalized separation criterion by fitting the mean velocity profile in the diffuser is implemented in code. This captures the effect of flow retardation in the diffuser and the shape of the velocity profile on the flashback limits. Including the underlying adverse pressure gradient in the flame backpressure expression further increases the flashback propensity by increasing the critical gradient. However, to fully reproduce the increased flashback tendency observed in the diffuser experiments, the turbulent flame speed needs to be positively tuned. This indicates that the increased flashback propensity could be due to differences in the time-resolved near-wall turbulence in the presence of an adverse pressure gradient.

Finally, the BLF model is discussed in the context of recently published numerical studies on the influence of the operating pressure on BLF (Endres & Sattelmayer 2019). These simulations suggest flashback propensity increases with increased pressure, even when the magnitude of flow separation is reduced. If this is confirmed, future modelling efforts for validation at gas turbine relevant pressures should focus on the interplay between flow separation and flame quenching at the wall.

Acknowledgements

First and foremost I would like to thank my supervisor Sikke Klein for his patient guidance and advice during this project and for providing me with the opportunity to present my work to interested parties. I would like to thank Rene Pecnik for partaking in the weekly meetings and providing his expert knowledge on the subject of CFD and turbulent flows. I also extend my sincere gratitude to Domenico Lahaye and Luuk Altenburg for reviewing my thesis and being part of the thesis committee.

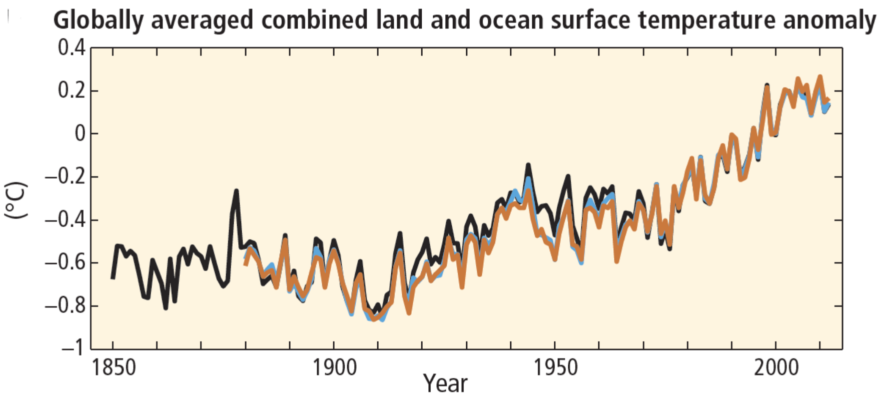
I am very grateful for the unwavering support of my family and friends in Iceland, and for the companionship of the friends I made in Delft. I am especially thankful to my fiancée Unnur, for her encouragement and for supporting my decision to leave and study abroad.

O.H. Björnsson
Reykjavík, September 2019

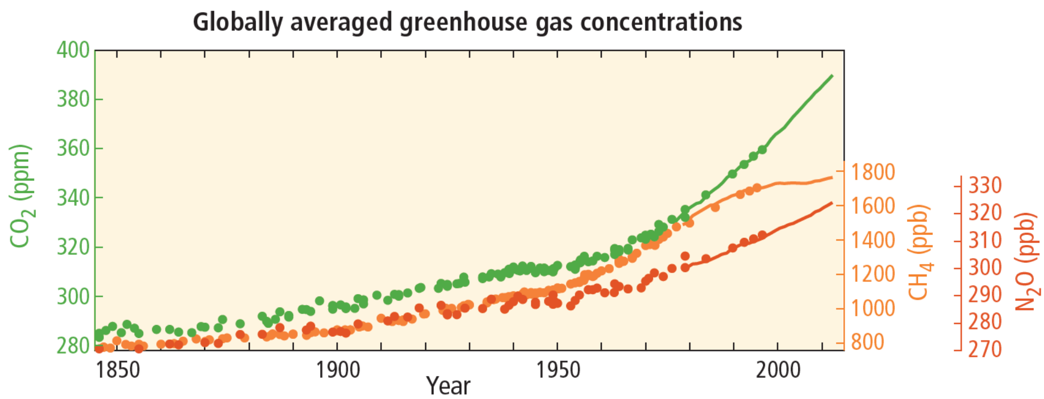
Contents

1
Introduction

According to the last synthesis report of the Intergovernmental Panel on Climate Change (IPCC), the period between 1983 and 2012 was very likely the warmest 30-year period of the last 1400 years in the Northern Hemisphere. The warming of the atmosphere is happening as levels of greenhouse gases (CO_2, methane and nitrous oxide) are higher than they have been for at least the last 800 millennia, due to strong increases in anthropogenic emissions in the industrial era. Figure 1.1 shows observations of the changing global climate. There is almost certainly a causal relationship between greenhouse gas emissions and global warming [48].



(a) Averaged temperatures relative to the average between 1986 and 2005.



(b) Greenhouse gas concentrations determined from ice core data (dots) and from direct atmospheric measurements (lines).

Figure 1.1: Observations of the changing global climate. Source: IPCC [48].

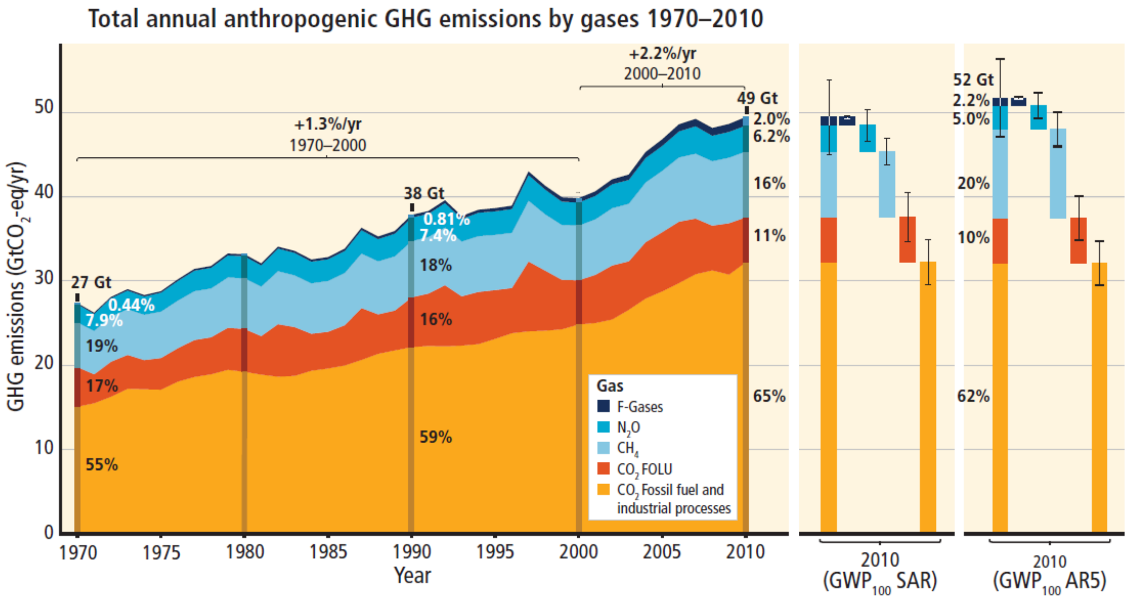
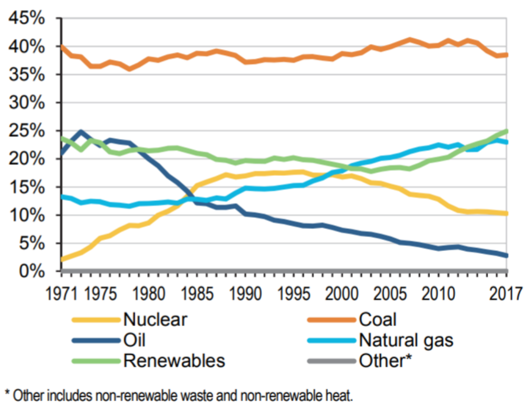


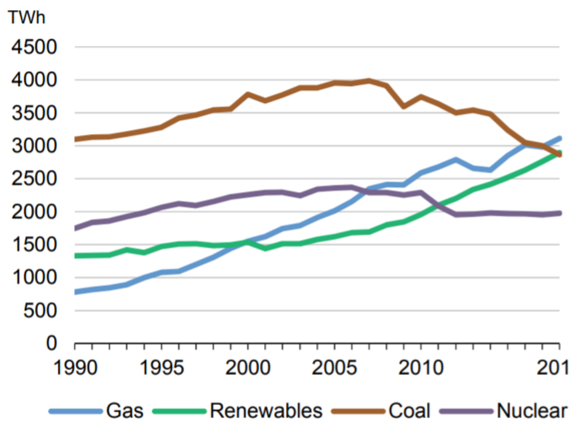
Figure 1.2: Total annual anthropogenic greenhouse gas (GHG) emissions from 1970 to 2010 in gigatonnes of CO_2 equivalent per year. Fossil fuels and industrial processes are the main source of CO_2 emissions. Forestry and other land use (FOLU) also contributes. Source: IPCC [48].

Figure 1.2 shows that most of the CO_2 released comes from the combustion of fossil fuels (coal, oil and natural gas) and industrial processes. Oil products are mostly used for transportation purposes while coal and natural gas are heavily relied upon for local electricity generation, as shown in Fig. 1.3a. Coal has satisfied around 40% of the worlds electricity needs for decades. In the OECD countries however, coal fired power plants are being phased out as seen in Fig. 1.3b. Natural gas and renewables are taking over as the main energy sources for electricity generation.

Natural gas power plants emit less CO_2 and less pollutants compared to coal or oil fired plants [2]. In natural gas power plants, gas turbines are used to drive electric generators. A gas turbine consists of a compressor, a combustor and a turbine. The natural gas is burned in the combustor to heat up incoming pressurized air, which then expands while doing work on the turbine blades. The heat in the exhaust gases can be used to cogenerate hot water or steam. In combined cycle gas turbines (CCGT's), the steam is expanded in a steam turbine to produce additional power resulting in energy efficiencies above 60% [3]. Figure 1.4 shows the schematics of a combined cycle gas turbine.



(a) Electricity generation share by source worldwide.



(b) Electricity generation by source in the OECD countries.

Figure 1.3: The sources of electric energy worldwide and in the OECD countries for 2019. Source: International Energy Agency [4].

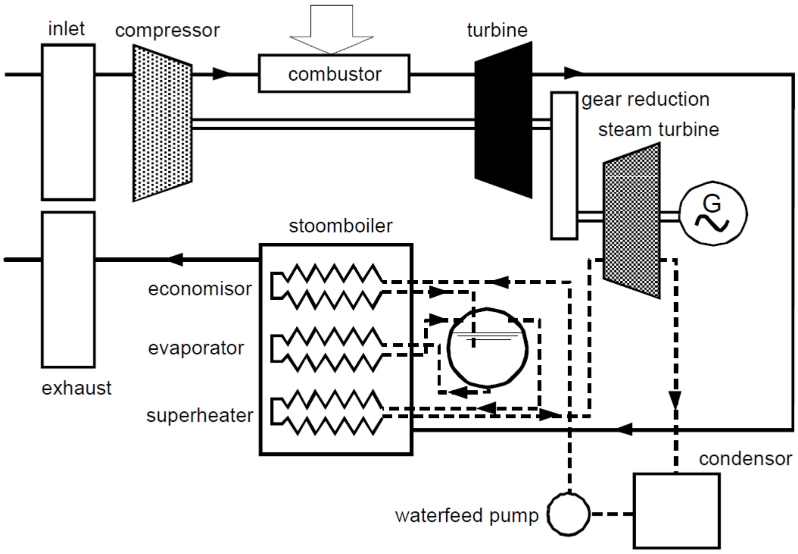


Figure 1.4: A combined cycle gas turbine used for high efficiency electricity generation. Gas is burned in the combustor, delivering heat to the power cycle which is then converted to mechanical power in the two turbines. Source: van Buijtenen, Visser [62].

While switching to natural gas from coal and oil does decrease the carbon emissions, there are ways to limit carbon emissions directly. Carbon Capture and Storage (CCS) techniques can be employed to capture the CO_2 either before or after combustion [1]. The captured CO_2 is then stored, e.g. underground or by mineral carbonation [45]. Pre-combustion capture can be achieved with natural gas reforming or coal gasification. The result is hydrogen-rich fuels. Post-combustion capture involves separating the CO_2 from the flue gas using membranes or adsorption technologies.

Apart from the emissions issues, fossil fuels are also a limited resource. Renewable energy sources are needed to satisfy the ever increasing energy demand and to drive future economic growth sustainably. Sunshine and wind are abundant energy sources that are already being harnessed cost effectively. However, their intermittent nature calls for smart solutions to store the energy when there is excess supply and to generate electricity from the stored energy during periods of excess demand. Hydrogen is a possible energy storage medium for this application. Hydrogen can be produced via electrolysis, a technology where electric power is harnessed to split water molecules into their constituents. The stored hydrogen can then be used on demand to generate electricity in a fuel cell. It can also be burned in a gas turbine retrofitted with hydrogen burners. This could be an attractive business case as it requires relatively little additional investments to the existing power plant, and gas turbines are well suited to balance the intermittent power supply from solar and wind since they offer fast start-up times, high efficiency and high turn-down ratios [25].

Stable burning of 100% hydrogen or hydrogen-rich mixtures is not straightforward. Hydrogen burns much quicker than standard fuels like natural gas, has a higher flame temperature and is more resistant to quenching [24]. These properties also contribute to a higher chance of so-called flame flashback, discussed in the next section.

1.1. Flame flashback

Modern gas turbines used for electricity generation primarily use lean premixed combustion in order to lower the heavily temperature dependent NO_x production and emissions. The excess air absorbs heat from the combustion which lowers the flame temperature. However, premixing of fuel and oxidizer opens up the possibility of flame flashback, where the flame propagates from the desired flame holding volume into the premixing section. Flashback is typically initiated in regions of low flow velocity relative to the consumption speed of the flame [30]. The occurrence of flashback necessitates a shutdown of the gas turbine and possibly causes damages to components that are not designed for high temperatures.

Flashback in gas turbine combustors can be caused by one of the four following mechanisms [6, 14, 30]:

- **Core flow flashback:** Happens when the turbulent burning velocity exceeds the core flow velocity. However, in industrial gas turbine burners, the bulk flow velocity in the premixer well exceeds the turbulent burning velocity such that the flame would instead be blown out if it were not stabilized.
- **Combustion instability induced flashback:** Large oscillations in the mixture flow, caused by instabilities due to e.g. heat release or pressure variations, lead to flame flashback.
- **Combustion induced vortex breakdown (CIVB):** Occurs in swirl stabilized burners. Swirling flow is used to create a recirculation area at the burner inlet in the core flow, stabilizing the flame. Under special circumstances, this recirculation area can propagate upstream into the premixing section which facilitates flame flashback in the core .
- **Boundary layer flashback (BLF):** At the premixer walls the flow velocity monotonously decreases to zero due to the no-slip condition and the viscosity of the fluid. Flashback can be initiated in the low speed boundary layer given that the flame is not quenched.

The unique properties of hydrogen (high flame speed, low quenching distances) cause premixed hydrogen-air mixtures to be especially prone to BLF [25]. Understanding the phenomena is therefore very important for the safe design of high-hydrogen burners [59]. The next section provides a brief introduction on recent advances in boundary layer flashback research.

1.1.1. Boundary layer flashback

Experimental research on BLF started already in 1943 when Lewis and von Elbe [34] were the first to perform systematic experiments on BLF, studying premixed laminar methane-air jet flames at atmospheric conditions. They developed a simple model which focuses on the wall gradient of streamwise velocity at flashback. This is the so-called critical velocity gradient model which has long been the accepted standard description of BLF. Figure 1.5 illustrates the concept.

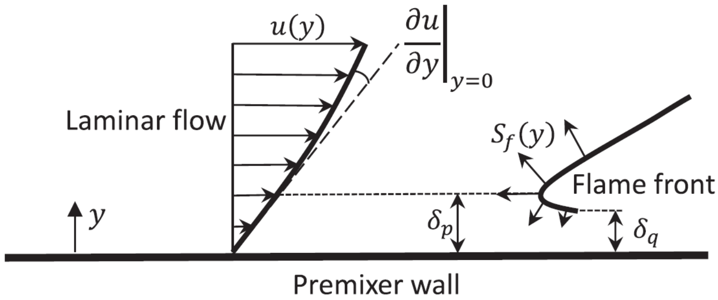


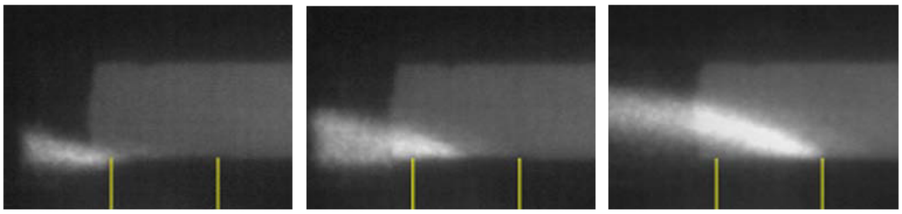
Figure 1.5: The critical gradient concept. Flashback is initiated if the local flame speed in the boundary layer exceeds the local flow speed. Flow and flame are assumed to be uncoupled. Source: Kalantari [30].

The near-wall velocity profile is assumed to be linear. The laminar flame front is quenched at a wall distance of δ_q and at the onset of flashback the flame speed is equal to the local flow speed at a distance δ_p from the wall, called the penetration distance. The flame speed at this point is assumed to be close to the laminar flame speed S_l of the fuel-oxidizer mixture. Flashback is then expected for:

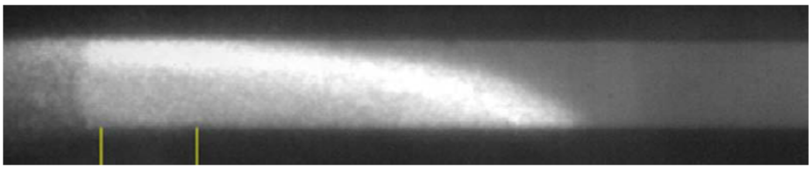
$$\frac{\partial u}{\partial y} < \frac{S_l}{\delta_p}$$

and $g_c = \frac{S_l}{\delta_p}$ is called the critical velocity gradient. Although the model was developed for laminar flames it has also been used for turbulent flames.

In 2011 research on boundary layer flashback started at TU Munich, sparked by recent interest in pre-combustion Carbon Capture and Storage and low-NO_x hydrogen-rich burner design. Eichler [14] studied the phenomena both experimentally and numerically. He focused on confined flames where the flame is stabilized inside a metal duct. The stabilization is achieved on a ceramic tile inserted flush with the duct wall, acting as a thermal insulator. Figure 1.6 shows the process of flame stabilization and flashback.



(a) Wake stabilization. **(b)** Upstream creep on ceramic tile. **(c)** Stabilization on ceramic tile.



(d) Flame at flashback.

Figure 1.6: Flame stabilization and flashback for H2-air mixtures in a 0° channel. Side view. Source: Eichler's PhD thesis [14].

Eichler found that this flame is much more prone to BLF than unconfined jet flames and will therefore give conservative flashback limits. He also found that the established critical gradient model fails to describe confined flame flashback for different degrees of mixture preheating and adverse pressure gradients. The critical gradient model assumes that the mixture flow and the flame are uncoupled. In reality the presence of the flame will affect the incoming flow. Eichler described that flow separation due to the backpressure effect of the flame gives rise to a flow recirculation area in which the flame can propagate upstream. DNS studies by Gruber [22] confirmed this new found mechanism of BLF.

Baumgartner [6] then studied jet flames and found that the critical gradient model also does not adequately describe the flashback mechanism for unconfined flames. He proposed an improved flashback model for this configuration. Hoferichter later [24–26] proposed models for both unconfined and confined flashback. Her confined BLF model is a semi-analytical model validated for ambient conditions and based on one experimental fitting parameter in the turbulent flame speed closure. It is built on the assumption that BLF is triggered by flow separation in front of the flame.

At TU Delft, Tober [59] validated the confined flashback model from Hoferichter for preheated hydrogen mixtures by including the effect of thermo-diffusive flame instabilities on the turbulent flame speed, and the effect of anisotropic turbulence on the flame stretch rate. The model is currently only validated for channels and tubes and needs to be extended to different geometries. That requires flow simulation for boundary layer flow in non-standard geometries where empirical expressions are not available, along with describing the effect of underlying pressure gradients on the flashback mechanism. This is the main topic of the thesis.

A BLF model applicable to any burner geometry could aid in the design against flashback in new industrial burners such as the FlameSheet™ burner depicted in Fig. 1.7.

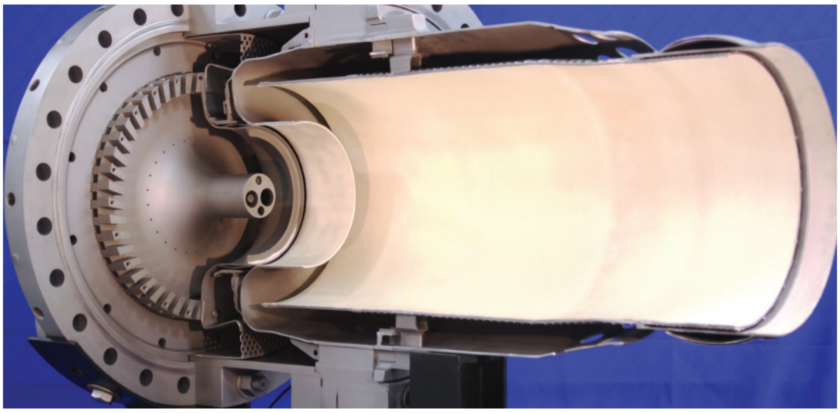


Figure 1.7: The PSM FlameSheet™ combustor. Source: PSM.com

It was originally developed by Power Systems Manufacturing (PSM) to reduce emissions and to handle lower load conditions with improved flame stability [56]. It is essentially a combustor within a combustor with two strong recirculation regions as shown in Fig. 1.8. It is currently being further developed for combustion of hydrogen-rich fuels [60].

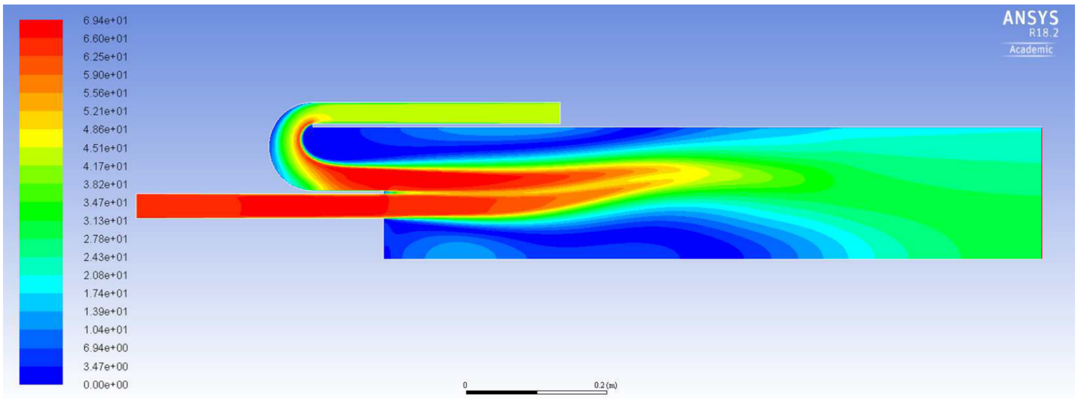


Figure 1.8: Velocity magnitude contours in m/s as predicted by CFD for the PSM FlameSheet™ combustor. The two large low velocity regions are flow recirculation flame stabilization zones. Source: Dankelman, Draskic, Ho and Muslem [13].

1.2. Research questions

This thesis aims to answer four research questions which are presented below.

In order to extend the BLF model to varying burner geometries, the effect of the underlying pressure gradient needs to be described both qualitatively and quantitatively. The model is currently only validated for flames confined in horizontal channels with favourable (negative) pressure gradients. Based on the observation of higher critical gradients for flames confined in diverging channels, the following question needs to be answered:

- *Why does an adverse pressure gradient increase confined flame flashback propensity?*

In the BLF model, Stratford's turbulent boundary layer separation criterion is applied to predict flame induced flow separation. Since this criterion was originally derived for boundary layers growing on aerofoils, it's applicability here is not obvious:

- *How should Stratford's turbulent boundary layer separation criterion be applied to predict flame backpressure induced flow separation in fully developed channel flow?*

Applying the BLF model to new burner geometries requires flow information, such as the mean velocity in the boundary layer. The empirical expression which are currently used in the model are only available for standard flows, such as channel and tube flow. Computational Fluid Dynamics can be used instead to simulate flow:

- *Can the BLF model, by coupling to CFD software, be extended to predict flashback limits in new burner concepts?*

The final question is related to the application of the CFD coupled BLF model to flames confined in diffusers. The underlying adverse pressure gradient in diffuser flow will act to retard the fluid, and will possibly change the nature of the turbulence. It could also increase the backpressure effect of the flame if the two adverse pressure gradients can be superimposed. This raises the following question:

- *Can the effect of an underlying adverse pressure gradient in diffuser geometries be separated from the effect of flame backpressure in the prediction of turbulent flow separation and flashback?*

1.3. Thesis outline

In chapter 2 basic concepts related to fluid flow and premixed combustion are discussed. Chapter 3 includes detailed discussions on confined BLF research, focusing on the confined BLF model and the experiments and assumptions it is built on. Boundary layer separation theory is also discussed as it forms an integral part of the model. Improvements to the BLF model made at the TU Delft are then explained.

A generalized form of Stratford's turbulent boundary layer separation criterion, applicable to the flame induced flow separation, is introduced and validated in chapter 4. The role of the flame stretch rate and Markstein length in computing the laminar flame speed is also discussed, focusing on the validity of the model at low equivalence ratios. In chapter 5, the BLF model is coupled to CFD and validated for flames confined in horizontal channels. It is then applied to flames confined in diffusers using the generalized separation criterion from chapter 4.

Finally, chapter 6 includes concluding remarks and recommendations for future research. A discussion on research containing new important insights but published after the bulk of this thesis had already been written is also included in chapter 6.1.

2

Flow and combustion basics

In this chapter the basics of fluid flow and combustion are discussed.

2.1. Fluid flow

Differential equations can be derived to describe mass conservation and momentum balance for every infinitesimal fluid element in the flow field. The so-called continuity equation describes mass conservation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

where $\mathbf{u}$ is the velocity vector $\mathbf{u}(\mathbf{x}, t) = (u, v, w)$. For two-dimensional flows and if density ρ is assumed to be constant such that the fluid can not compress or expand, the equation reduces to:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.2}$$

or $\nabla \cdot \mathbf{u} = 0$, the flow field is *divergence free*. Flow of unreacted mixture in burners can be approximated as incompressible since the flow velocities are low relative to the speed of sound [5, 6].

The Navier-Stokes Equations (NSE), a set of coupled partial differential equations for velocity and pressure, describe the fluid momentum balance on an infinitesimal scale. Equation 2.3 is the incompressible, constant viscosity version of the Navier-Stokes [57]:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla \mathbf{p} + \mu \nabla^2 \mathbf{u} + \mathbf{f} \tag{2.3}$$

The NSE includes an unsteady term, a non-linear advection/inertial term (bulk transport), a pressure term with $\mathbf{p}$ for pressure, a momentum diffusion term due to the viscosity of the fluid and a body force $\mathbf{f}$. For two dimensional flow, equation 2.3 represents two equations for x and y-momentum, respectively:

$$\rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = -\frac{\partial p}{\partial x} + \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f_{\mathrm{x}} \tag{2.4}$$

$$\rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = -\frac{\partial p}{\partial y} + \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + f_{\mathrm{y}} \tag{2.5}$$

Turbulent flow is characterized by a large Reynolds number:

$$\mathrm{Re} = \frac{\rho u L}{\mu} \tag{2.6}$$

such that the inertial term in the NSE dominates over the viscous term. Solving the NSE directly for turbulent flows is impractical for most industrial applications, so the NSE are typically averaged using the so-called

Reynolds decomposition where the flow variables are divided into a mean part and a fluctuating part as done here for the u velocity component:

$$u(\mathbf{x},t) = \overline{u}(\mathbf{x}) + u'(\mathbf{x},t) \tag{2.7}$$

Averaging the continuity equation and the NSE results in the Reynolds-averaged Navier-Stokes (RANS) equations, given in tensor notation as [19]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho u_i\right) = 0 \tag{2.8}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial u_l}{\partial x_l}\right)\right] + \frac{\partial}{\partial x_j}\left(-\rho\overline{u_i' u_j'}\right) \tag{2.9}$$

The effect of turbulence is included with additional terms $-\rho\overline{u_i' u_j'}$ called Reynolds stresses which have to be modelled to close the set of equations. Different turbulence models are used to close the equations. Many of them employ the Boussinesq hypothesis, where the momentum transfer of turbulent eddies is described using an effective eddy viscosity μ_t:

$$-\rho\overline{u_i' u_j'} = \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\left(\rho k + \mu_t\frac{\partial u_k}{\partial x_k}\right)\delta_{ij} \tag{2.10}$$

One disadvantage to this approach is that the turbulence is assumed to be isotropic. To have anisotropic turbulence, the Reynolds Stress Model (RSM) can be used where a transport equation is solved for each Reynolds stress [19].

2.1.1. Boundary layer flow

Figure 2.1 illustrates a boundary layer growing on a flat plate.

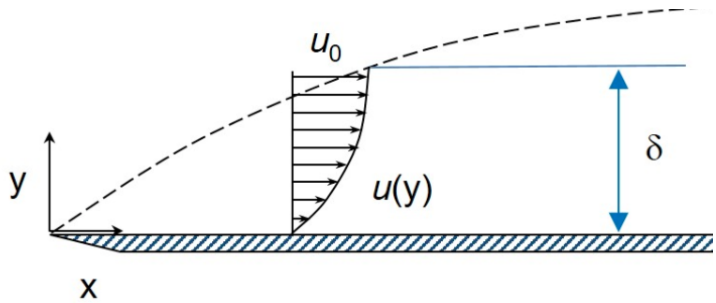


Figure 2.1: Illustration of a boundary layer growing on a flat plate. Source: Wikipedia/Boundary Layer

Ludwig Prandtl was the first to introduce a hypothesis of thin boundary layers where viscous effects were strong in otherwise highly inertial flows [33]. He showed that drag on objects inserted in low viscosity flow could be accounted for by introducing a no-slip condition at the surface. By non-dimensionalizing the Navier-Stokes equations and assuming that the boundary layer thickness δ is small compared to the characteristic length scale of the flow, he came up with the governing equations for boundary layers. The governing equations for continuity and streamwise x-momentum in case of incompressible flow are [33]:

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{2.11}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial y^2}\right) \tag{2.12}$$

assuming steady incompressible flow with constant density and viscosity. The momentum equation for the y-velocity component is simply:

$$\frac{\partial p}{\partial y} = 0 \tag{2.13}$$

meaning the pressure in the boundary layer is independent of y and takes the value of the pressure in the inviscid flow at the edge of the boundary layer. Blasius [9] presented accurate similarity solutions for flow over a flat plate. The thickness of the boundary layer δ can be shown to scale as (U_0 is the far-field velocity) [33]:

$$\delta(x) \sim \sqrt{\frac{\nu x}{U_0}}$$

by considering the time scale of viscous diffusion $t = \delta^2/\nu$. There is no universal definition for the boundary layer thickness since there is no obvious point on the y-axis where the boundary layer ends and the outside flow begins. Two useful definitions for the boundary layer thickness are the displacement thickness δ^* and momentum thickness θ, respectively:

$$\delta^*(x) = \int_0^\infty \left(1 - \frac{u}{U_0}\right) \mathrm{dy} \approx 1.72\sqrt{\frac{\nu x}{U_0}} \tag{2.14}$$

$$\theta(x) = \int_0^\infty \frac{u}{U_0}\left(1 - \frac{u}{U_0}\right) \mathrm{dy} \approx 0.665\sqrt{\frac{\nu x}{U_0}} \tag{2.15}$$

The displacement thickness is equal to the distance that an external potential flow would have to be displaced vertically to equal the same loss of flow rate as caused by the boundary layer. The momentum thickness is equal to the distance that the same potential flow would have to be displaced to equal the same loss of momentum as caused by the boundary layer.

Turbulent boundary layers are typically described with the following dimensionless quantities [24]:

$$y^+ = \frac{\rho u_\tau(x) y}{\mu} \tag{2.16}$$

and

$$u^+ = \frac{\overline{u}(x,y)}{u_\tau(x)} \tag{2.17}$$

where $u_\tau = \sqrt{\tau_w/\rho}$ is the friction velocity based on the wall shear stress τ_w. The boundary layer is then divided into the following regions based based on the value of y^+ [6, 24]:

- **Viscous sublayer** ($y^+ \leqslant 5$) Viscosity dominates with:

$$u^+ = y^+ \tag{2.18}$$

- **Buffer layer** ($5 < y^+ < 30$): Transition region.
- **Logarithmic region** ($30 \leqslant y^+ \leqslant 350$): Viscosity and turbulence both play a role. The logarithmic law-of-the-wall describes the flow:

$$u^+ = \frac{1}{K}\ln y^+ + 5.0 \tag{2.19}$$

  where K = 0.41 is the von Kármán constant.

2.2. Premixed Combustion

Lean premixed combustion is the standard method to minimize NO_x formation and emissions in modern gas turbine power generation [8]. Excess air in the fuel-air mixture effectively cools the flame lowering the heavily temperature dependant NO_x formation. The mixture equivalence ratio ϕ is defined as the actual fuel-air ratio divided by the stoichiometric fuel-air ratio. A stoichiometric mixture is indicated by $\phi = 1$. If $\phi < 1$ the mixture is called lean and rich mixtures have $\phi > 1$.

2.2.1. Laminar flames

Premixed mixtures burn with a mixture dependent laminar flame speed $S_{l,0}$, defined as the propagation speed of a one dimensional planar adiabatic flame relative to the mixture flow. Figure 2.2 shows the planar flame front and how it can be divided into a preheat zone and a reaction zone.

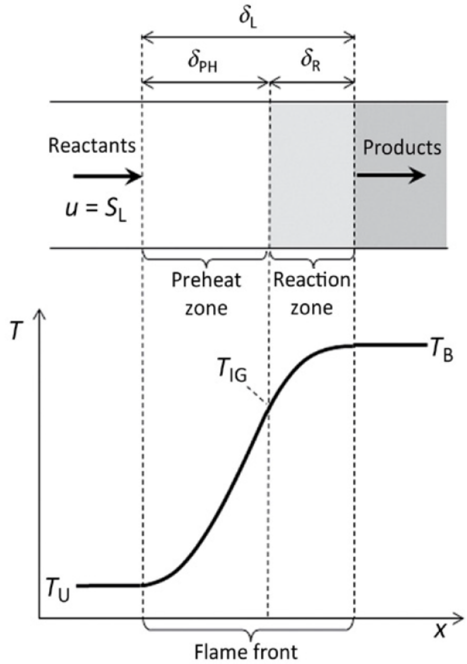


Figure 2.2: A one dimensional planar flame front. Source: Benim and Syed [8].

Heat is generated in the reaction zone where products are formed from reactants. Heat diffuses from the reaction zone into the preheat zone and reactants diffuse in the opposite direction. The balance of thermal α and mass D diffusion is described with the Lewis number:

$$\mathrm{Le} = \frac{\alpha}{D} \tag{2.20}$$

Unbalanced thermal and mass diffusion can increase or decrease the laminar flame speed of stretched flames. The flame stretch rate is defined as the normalized time rate of change of the flame front area [24]:

$$\kappa = \frac{1}{A}\frac{dA}{dt} \tag{2.21}$$

It is caused by two different phenomena, flow strain and flame curvature:

$$\kappa = \kappa_{\mathrm{strain}} + \kappa_{\mathrm{curv}} \tag{2.22}$$

Poinsot and Veynante [46] use Fig. 2.3 to illustrate typical configurations used to study different types of flame stretch. Flame stretch due to flow strain could be compared to how a thin rubber sheet can be stretched by pulling it's corners. Positive stretch is observed when two points on a flame front move away from each other. This only happens when there is a velocity gradient in the plane tangent to the flame front [46]. An example of flame stretch due to curvature is a change of radius and hence area of a spherical flame surface as the flame front propagates relative to a stationary fuel-air mixture (similar to a combustion bomb experiment).

The effect of flame stretch on laminar flame speed can be quantified by computing the stretched laminar flame speed:

$$S_{l,s} = S_{l,0} - L_M \kappa \tag{2.23}$$

as presented by Markstein [36]. L_M is the mixture dependent so-called Markstein length which determines the sensitivity of the flame speed to flame stretch. It depends on parameters such as mixture equivalence ratio and the Lewis number [8]. This relation is built on asymptotic theory and is therefore only strictly valid for $\mathrm{Ka} = \kappa \delta_F / S_{l,0} < 1$, i.e. weakly stretched flames.

The laminar flame speed $S_{l,0}$ can be calculated using e.g. the chemical simulation software Cantera [21] using an appropriate reaction mechanism.

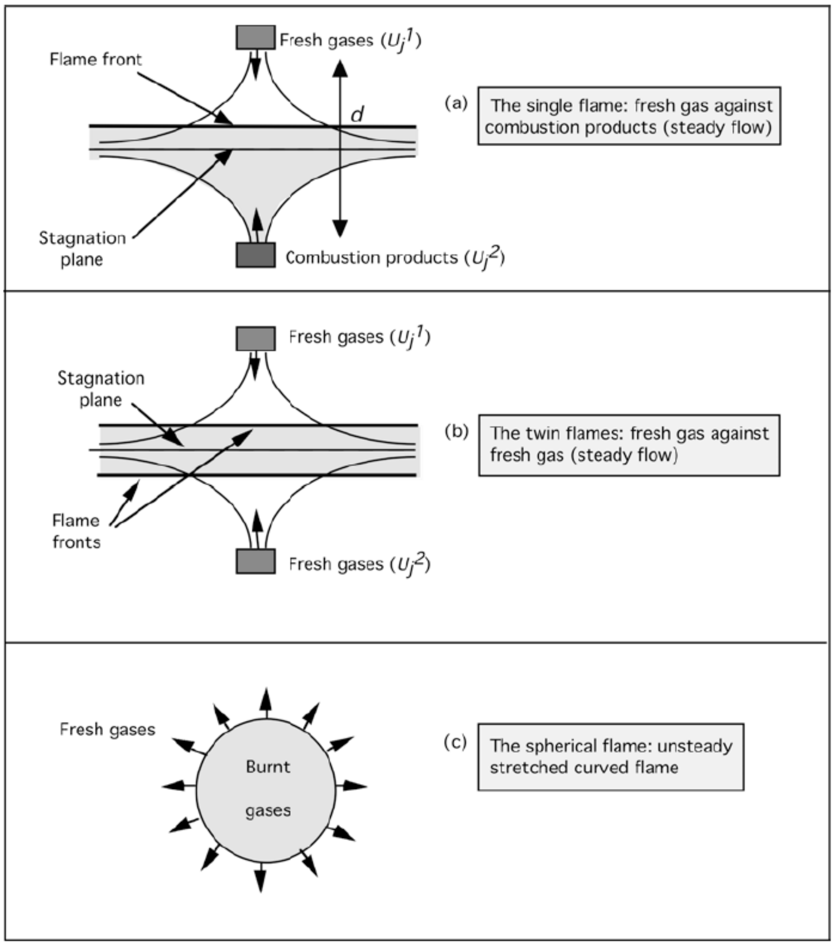


Figure 2.3: Examples of typical configurations used to study flame stretch in laminar premixed flames. (a) and (b) show flames that are stretched due to flow strain and (c) will stretch due to flame curvature. Source: Poinsot and Veynante [46].

2.2.2. Turbulent flames

Turbulence in the flow field will curve and wrinkle the flame front, increase the burning area and increase fuel consumption speeds. In industrial applications such as gas turbines, turbulent flames are used to increase the rate of heat release. Turbulent flames can be characterized depending on the scale and intensity of the turbulent eddies versus the flame thickness and laminar flame speed of the mixture. Peters [43] identified five different burning regimes of turbulent flames shown in Fig. 2.4.

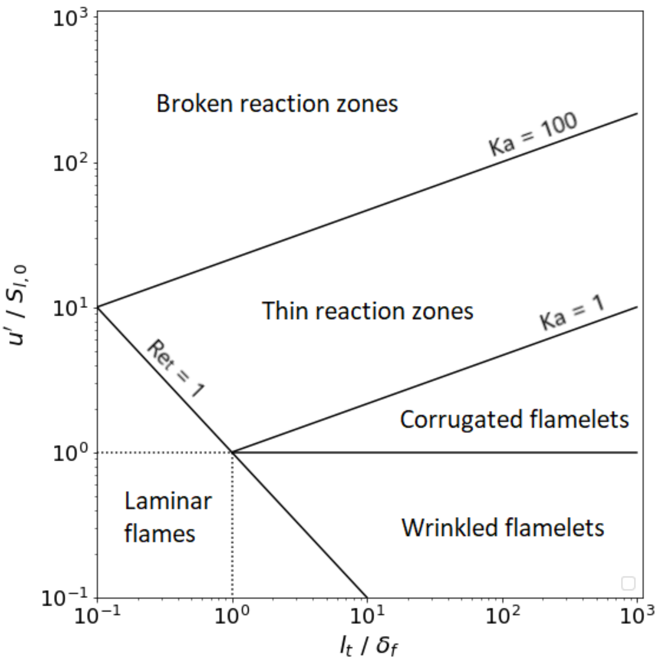


Figure 2.4: Modified turbulent combustion diagram proposed by Peters [43]. Based on figures in Poinsot and Baumgartner [6, 46].

Similar diagrams have been proposed by different authors [46]. Isotropic turbulence is assumed and characterized by the fluctuation velocity u' and the length scale of the largest turbulent eddies. The length scale ranges from the smallest eddies to the largest, i.e. Kolmogorov scale η_k to the integral scale l_t. The time scale of turbulence can be defined as:

$$t_t = \frac{l_t}{u'} \tag{2.24}$$

or

$$t_k = \frac{\eta_k}{u'(\eta_k)} \tag{2.25}$$

for the largest and smallest eddies respectively. The chemical time scale can similarly be defined as:

$$t_c = \frac{\delta_f}{S_{l,0}} \tag{2.26}$$

The Damköhler number compares the time scales of the largest turbulent eddies and the flame:

$$\mathrm{Da} = \frac{t_t}{t_c} = \frac{l_t S_{l,0}}{u' \delta_f} \tag{2.27}$$

while the Karlovitz number compares the chemical time scale and the time scale of the smallest turbulent eddies [46]:

$$\mathrm{Ka} = \frac{t_c}{t_k} = \frac{u'(\eta_k)\delta_f}{\eta_k S_{l,0}} = \left(\frac{l_t}{\delta_f}\right)^{-\frac{1}{2}} \left(\frac{u'}{S_{l,0}}\right)^{\frac{3}{2}} \tag{2.28}$$

If the chemistry is fast compared to the smallest eddies (Ka < 1), the effect of the turbulent fluctuations is to wrinkle the flame front. The flame front is locally behaving like a laminar flame. Corrugated flamelets are strongly wrinkled flames due to the fluctuation velocity u' being higher than the laminar flame speed $S_{l,0}$. In the thin reaction zone, the smallest eddies can penetrate the preheat zone but not the reaction zone. This causes mixing in the preheat layer increasing flame speed. If Ka ≥ 100 both the preheat- and the reaction zone are penetrated by turbulence. This is characterized by broken reaction zones.

The wrinkled flames, corrugated flames and flames with thin reaction zones can be modeled with the flamelet assumption, which assumes a locally laminar flame [6, 46]. Then a turbulent burning velocity S_t can be defined by relating the increase in flame speed to the increase in flame front area [44]:

$$S_t = S_{l,0}\frac{A_t}{A} \tag{2.29}$$

This was first proposed by Damköhler in 1940. Based on this idea he further derived:

$$\frac{S_t}{S_{l,0}} = 1 + C\left(\frac{u'}{S_{l,0}}\right)^n \tag{2.30}$$

where C should depend on the length scale ratio of the largest turbulent eddies and the flame, l_t/δ_f. The exponent n takes a value between 0.5 and 1. The turbulent flame speed is equal to the laminar one if $u' = 0$. Approximately 40 other correlations for turbulent flame speed have been proposed. Burke et al. [11] compared 16 correlations for hydrocarbon fuels at elevated pressures against a large experimental data set. The experimental setups used to build the correlations differ in e.g. the magnitude of turbulence, flame holding, heat transfer etc. The correlations were judged on their ability to predict flame speed trends for various fuels, turbulence conditions and pressures. Burke found that the following correlation by Muppala [38] performed best overall:

$$\frac{S_t}{S_{l,0}} = 1 + \frac{C}{\mathrm{Le}}\mathrm{Re}_t^{0.25}\left(\frac{u'}{S_{l,0}}\right)^{0.3}\left(\frac{P}{0.1\mathrm{MPa}}\right)^{0.2} \tag{2.31}$$

with the turbulent Reynolds number defined as:

$$\mathrm{Re}_t = \frac{u' l_t}{S_{l,0}\delta_f} \tag{2.32}$$

This relation is similar to Eq. (2.30) apart from the added pressure- and Lewis number dependency.

3

Boundary layer flashback of confined flames

Eichler [14] showed that the most conservative flashback limits correspond to confined flames stabilized on a ceramic tile inserted flush with the burner wall (see Fig. 1.6). In this chapter research on confined boundary layer flashback (BLF) carried out at TU Munich will be discussed in detail along with recent work at TU Delft to improve the confined BLF model.

3.1. Experiments on confined flame flashback in turbulent boundary layers

Eichler [14] worked on flame flashback in wall boundary layers during his PhD period from 2007-2011 at Lehrstuhl für Thermodynamik, TU Munich. Using lean to stoichiometric premixed CH_4-air and H_2-air mixtures he carried out flashback experiments for different types of flame anchoring, varying adverse pressure gradients and varied preheating. The complete performance specifications are given in Table 3.1, on the next page.

Fig. 3.1 shows a sketch of the measurement section. A rectangular measurement section with a high as-

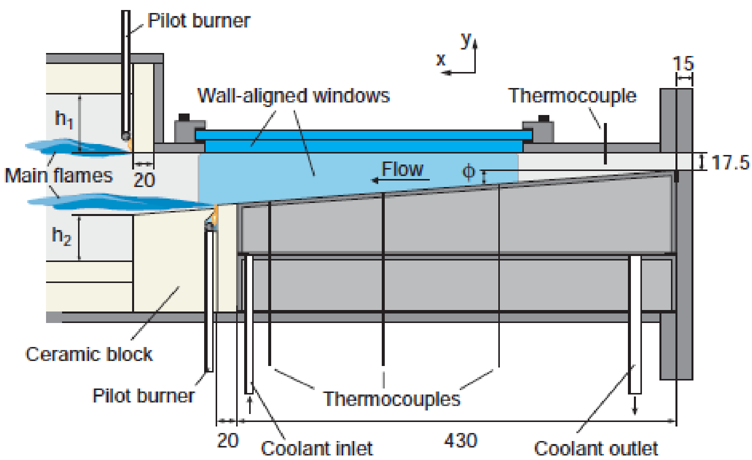


Figure 3.1: A sketch of the experimental measurement section. The lower wall section is interchangeable. 0°, 2° and 4° ramps were used during experiments to vary the pressure gradient. The dimensioning is done in millimeters. Source: Eichler's PhD thesis [14].

pect ratio is used. Transparent panels allow for simultaneous optical measurements from the sides and top. The wall temperature is controlled since it influences near-wall flashback propensity. A fully developed turbulent premixed mixture flows from right to left through the measurement section down an interchangeable ramp allowing varying opening angles. An adverse pressure gradient is realized by changing the opening angle from 0° to 2° or 4°. The ramp is cooled from below using air jets. Downstream of the ramp a ceramic tile is fitted flush with the ramp. It serves as thermal insulation and as a thermal flame holder. The flame is stabilized in the wake of a small backwards facing step and ignited with the help of a pilot burner. To avoid flashback along the side walls, air is blown along the corners. Flashback only occurs along the lower wall in the center region where a quasi two-dimensional flow field is assumed.

Table 3.1: Performance specification for the experimental rig used by Eichler. Taken directly from Eichler's PhD thesis [14].

Criterion	*Target*	*Reason/Comment*
Mixture preparation		
Components	CH_4, H_2, air	Influence of fuel properties, possibility to observe laminar and turbulent flashback.
Equivalence ratio	$0 \le \Phi \le 1$	Lean premixed gas turbine combustion as technology standard.
Mixing process	Fully premixed	Comparability with premixed flame theory.
Inlet conditions		
Reynolds number	$\mathcal{O}(10^3)$ up to $\mathcal{O}(10^5)$	Laminar and turbulent flow.
Mixture temperature	up to 400° C	Conditions in gas turbine combustors.
Static pressure	atmospheric	Compromise to reduce rig complexity and cost.
Measurement section		
Cross section	rectangular	Optical measurements in the near-wall region. Distinction from literature.
Flame holding	confined	Distinction from literature.
Optical access	three sides	Simultaneous optical measurements in two planes.
Channel aspect ratio	high	Exclusion of sidewall influence, 2D time-mean boundary layer.
Global pressure gradient	zero or adverse	Channel and diffuser geometries.
Wall temperature	controllable	High impact on flashback propensity.

3.1.1. Critical gradient results

Eichler's experiments produced flashback limits of H_2-air and CH_4-air mixtures with different types of flame anchoring, varying adverse pressure gradients and various degrees of preheating. Fig. 1.6 in chapter 1 showed flame stabilization and the flashback event in a 0° channel.

Confined flashback limits for H_2-air mixtures in a 0° channel are shown in Fig 3.2, obtained in terms of a critical gradient for a given equivalence ratio.

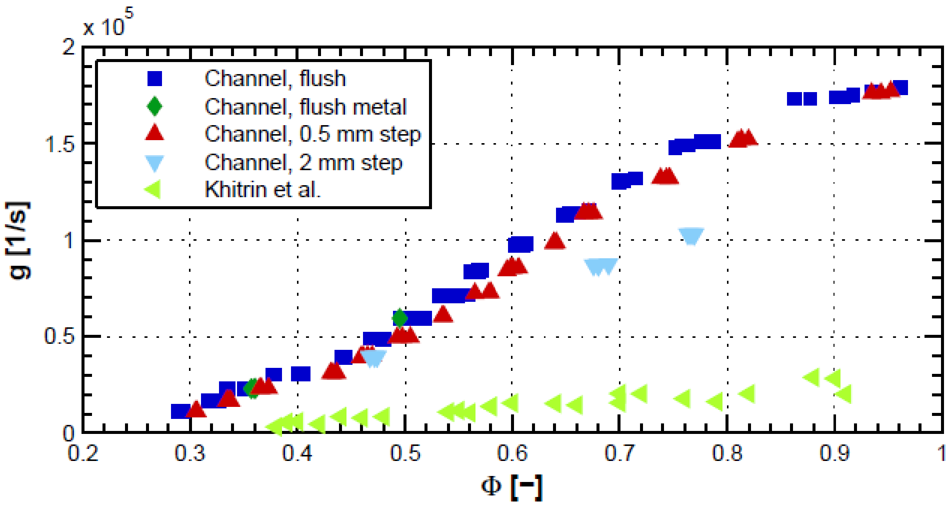


Figure 3.2: Turbulent wall flashback limits for H2-air mixtures in a 0° channel. Results for unconfined flames from Khitrin et al. are also plotted for comparison.

Results from Eichler's channel are plotted for various heights of the backwards facing step used for flame anchoring. Increasing the step height lowers flashback propensity somewhat. Results for unconfined tube flames from Khitrin et al. are also plotted for comparison. Eichler's confined flame is more prone to flashback since the critical gradients in his experiments are higher. This is an important finding since it implies that unconfined flame flashback limits in literature can not be used for safe design against flashback in case the flame accidentally enters the premixing section [14].

Eichler also demonstrated that 0° tube burners show the same increased flashback propensity for confined flame holding and that the tube and channel cases have similar flashback limits. This suggests a universal increased flashback propensity in confined versus unconfined burners, even though the mean velocity profiles in the boundary layer should be similar at the duct exit and in the confine. Eichler concludes that the critical gradient model is unsatisfactory since it can not explain this difference.

Results for flashback experiments involving preheated mixtures are given in Fig. 3.3.

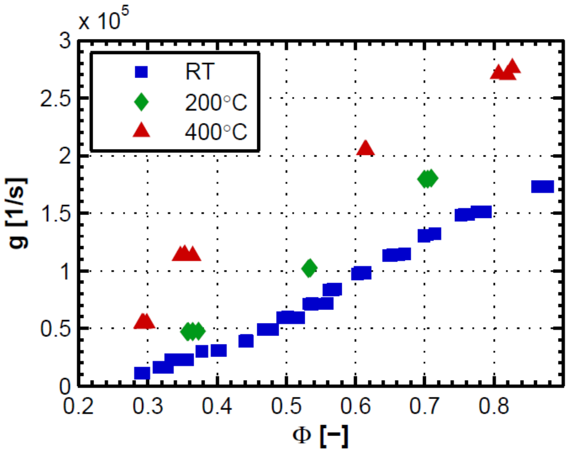
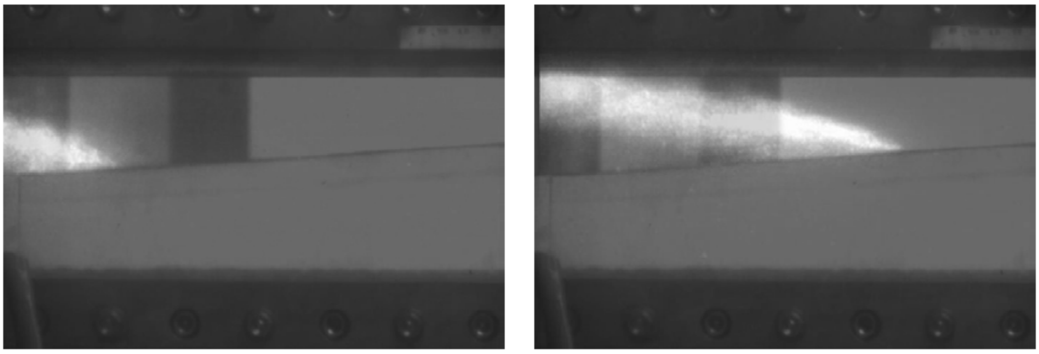


Figure 3.3: Turbulent wall flashback limits for preheated H_2-air mixtures in a 0° channel. RT stands for room temperature (20 °C). Source: Eichler's PhD thesis [14].

These were done only for the 0° channel and for the flush configuration of the flame anchoring ceramic tile. The lower wall of the channel was also preheated to the temperature of the mixture. Compared to the room temperature results, flashback limits are higher for the preheated cases and increase with temperature.

Flashback was also observed for atmospheric H_2-air mixtures in 2° and 4° planar asymmetric diffusers. Figure 3.4 shows the flashback event in a 4° diffuser.



(a) Low equivalence ratio. (b) High equivalence ratio.

Figure 3.4: Flashback observed in a 4° planar asymmetric diffuser. Source: Eichler's PhD thesis [14].

A key difference compared to the channel experiments is that the flame does not flashback continuously through the whole section. After flashback is initiated the flame front propagates upstream a finite distance before it starts to oscillate around a mean axial position with a frequency of a few Hertz. The critical gradient is derived for the mean axial position of the flame front using numerical simulation results for wall shear.

Figure 3.5 on the next page shows the flashback limits compared to the 0° channel results. It is evident that critical gradients for the diffusers, with underlying adverse pressure gradients, are higher than in the channel and tube. An increase of the adverse pressure gradient further increases the critical gradient based on the fact that 4° results lie above 2° results. At very lean equivalence ratios around $\Phi \approx 0.35$, the 2° and 0° critical gradients are similar, but Eichler suggests an underprediction of numerical wall shear in the diffusers may be at fault and that in reality the two could start deviating at lower equivalence ratios.

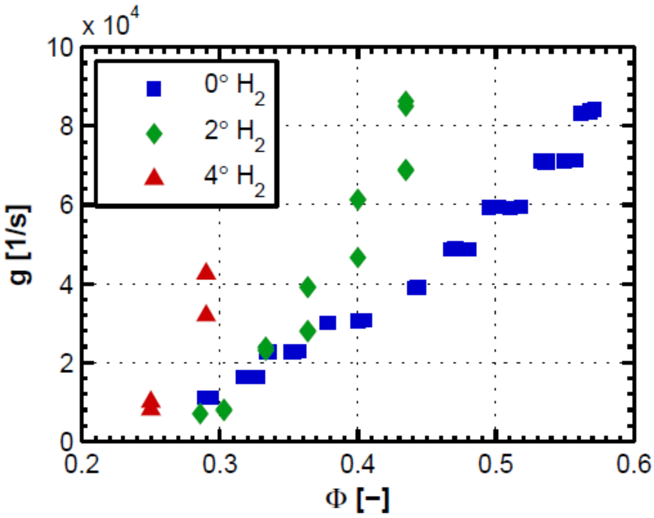


Figure 3.5: Turbulent wall flashback limits for H_2-air mixtures in 2° and 4° channels. Source: Eichler's PhD thesis [14].

Eichler mentions that in some cases flashback was observed at the same equivalence ratio for two different air massflows. In those cases, the critical gradients for high mass flows lie above their low mass flow counterparts. For higher mass flows, the flame front is oscillating further downstream in the diffuser. Eichler concludes that the downstream boundary layer is more susceptible to flashback, i.e. that for the same gradient at the wall, a boundary layer further downstream will flashback more easily and therefore at lower equivalence ratios.

The physical reason for the increased propensity of flashback for flames in adverse pressure gradients is yet unclear. Figure 3.6 shows the boundary layer in the 4° diffuser from LDA measurements plotted against the canonical Spalding profile for 0° channel flow.

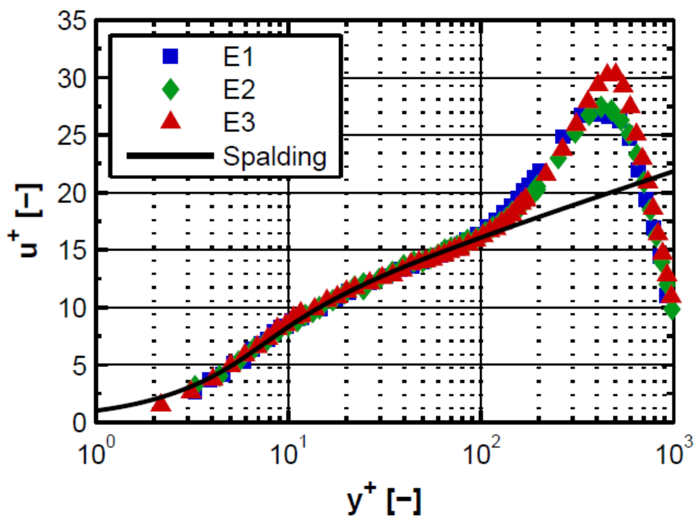
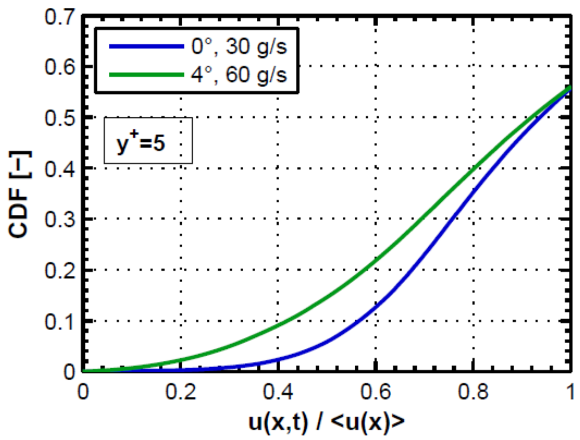


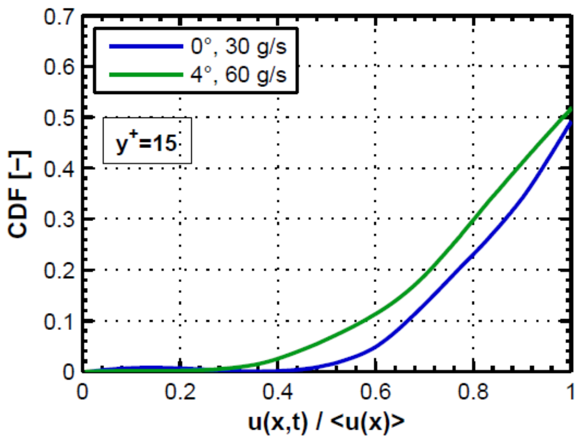
Figure 3.6: The dimensionless mean velocity profile of air in Eichler's 4° diffuser. LDA measurements. Mass flow: 60 g/s. Source: Eichler's PhD thesis [14].

The 4° results show an increased mean velocity in the wake region but the near wall flow below $y^+ = 100$ is identical. The mean velocity is normalized with the friction velocity $u_\tau = \sqrt{\tau_w/\rho}$ so the comparison is equivalent to comparing two profiles with the same wall gradient (du/dy). This is an important finding because it shows that the increased propensity for flashback in the diffuser can not be caused by a difference in the near-wall mean velocity profile. This is further evidence that the critical gradient model is unsatisfactory. Eichler even concluded that the increased wake velocity should decrease the effective backpressure of the stabilized flame and therefore decrease flashback susceptibility, not increase it. He reasoned that since the stabilized flame reaches into the main flow region, the streamlines of the flow are deflected upwards away from the flame due to the local flame backpressure effect. The stronger the curvature of the deflected streamlines, the stronger the backpressure effect. The presence of a top wall will decrease this curvature, decreasing the effective backpressure and cause a higher mean velocity in the wake region. This effect could also explain the difference in flashback propensity between upstream and downstream positions in the diffuser, since the duct height increases throughout the diffuser [14].

Since the higher flashback propensity can neither be explained by a difference in the mean velocity profile nor a difference in the effective backpressure, Eichler investigated if there were differences to be found in the instantaneous time resolved velocity profile. He found evidence of an increased frequency of low velocity streaks in the near-wall diffuser flow. Figure 3.7 shows a comparison of cumulative distribution functions of the near-wall normalized instantaneous velocity in the 4° diffuser and 0° channel flow. The CDF's show that low speed streaks are more frequent in the 4° diffuser which could explain the increased susceptibility for flashback.



(a) CDF at $y^+ = 5$ at E2.



(b) CDF at $y^+ = 15$ at E2.

Figure 3.7: Cumulative distribution functions of near-wall instantaneous velocity normalized with the mean velocity in a 4° diffuser versus 0° channel.

3.1.2. Turbulent combustion regime

Eichler presented estimations for the turbulent combustion regimes of the flames at flashback. Figure 3.8 shows results of his estimations according to Peters' regime diagram.

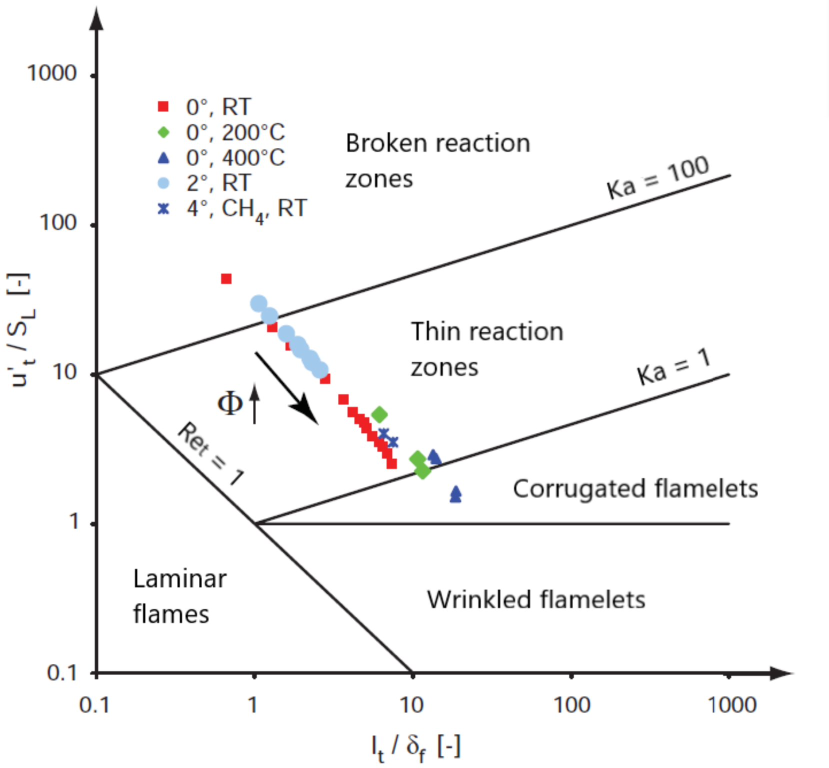


Figure 3.8: Estimated turbulent combustion regimes of flames during flashback in Eichler's experiments. The data is mainly for H_2-air mixtures except for one CH_4 experiment since flashback was only observed for CH_4 mixtures in the 4° diffuser. Source: Eichler's PhD thesis [14].

According to this figure the flames are mainly lying in the thin reaction zone regime. The leanest mixtures in the 0° and 2° ducts stretch into the broken reaction zone which could be explained by the laminar flame speed tending to zero according to Eichler. The richest preheated channel points lie close to or inside the corrugated flamelet regime which was consistent with the observed smoothness of the flames at flashback. The turbulent length scale l_t used is the diameter of the quasi-streamwise vortices in the inner layer of the turbulent boundary layer, given as $d+ = \nu / u_\tau \approx 30$ by Eichler. The velocity fluctuations u' are estimated from LDA measurements and taken at the maxima found at $y^+ \approx 15$. The flame thickness δ_f is estimated using a relation from Peters [43]:

$$\delta_f = \frac{(k/c_p)_{il}}{\rho S_L} \tag{3.1}$$

with k as thermal conductivity, c_p as the heat capacity at constant pressure and ρ as the unburned mixture mass density. The 'il' means properties are taken at the temperature of the inner layer of the flame where fuel is consumed. This relation is for a one-dimensional, unstretched adiabatic flame. The near-wall flames (noted by Eichler as being in the buffer- and log layer of the boundary layer structure) are however three-dimensional, stretched and diabatic so Fig. 3.8 should be viewed with that in mind [14].

3.1.3. Near-wall flame propagation studies and a new physical model for wall flashback

Eichler also studied the details of flame propagation in the near-wall region. The macroscopic structure of the turbulent flames during flashback can be seen in the top row of Fig. 3.9.

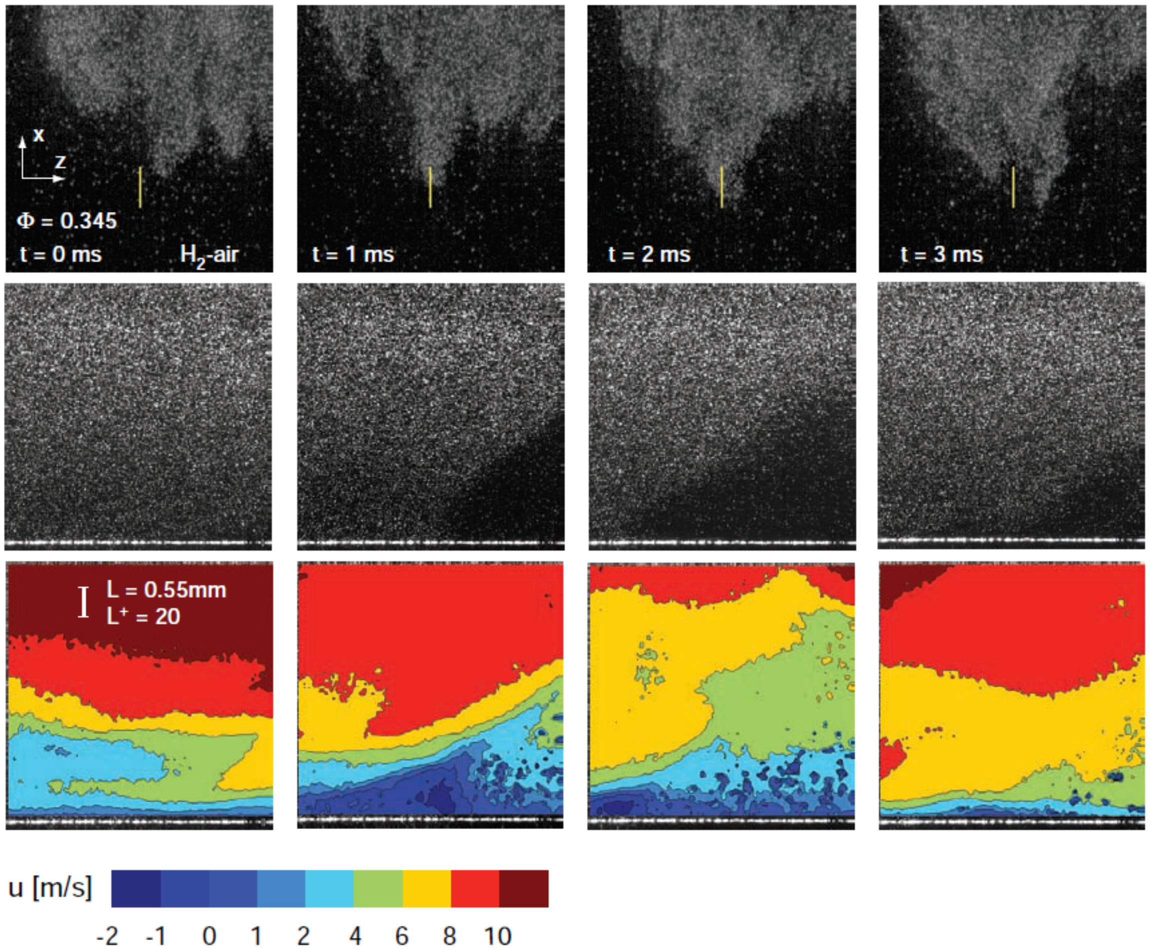


Figure 3.9: Axial velocity contours in front of a turbulent H_2-air flame in a 0° channel showing a distinct backflow region in front of the flame.

The turbulent flames show wrinkled chaotic flame fronts with flame cusps. A flame cusp can travel both upstream or downstream, and change shape by thickening laterally and subsequently breaking up into new cusps. The middle and bottom rows give a microscopic side-view picture of what is happening in front of the flame cusps.

The bottom contour images show axial velocity contours superimposed on the x-y side view shown in the middle row of images. The top row of images are macroscopic top-down views of the flashback event with the side view measurement section shown in bright yellow. Figure 3.9 shows how a backflow region is formed right in front of the upstream propagating flame cusp. The maximum negative velocities are found close to the tip of the cusp and the whole backflow region is attached to the wall. This backflow event was observed in all cases of upstream flame cusp propagation. Figure 3.10 shows the same series of events in the 4° diffuser for turbulent H_2-air mixtures.

The streamwise and vertical span of the backflow region is similar for the 2° and 0° cases, but greater in the 4° case. The backflow regions observed mean that the flow is separating from the wall in front of the flame. Eichler also studied laminar flashback in similar fashion and found the same separation in front of the flame cusps, although the flow is less chaotic. The backflow region extends further upstream in the laminar case.

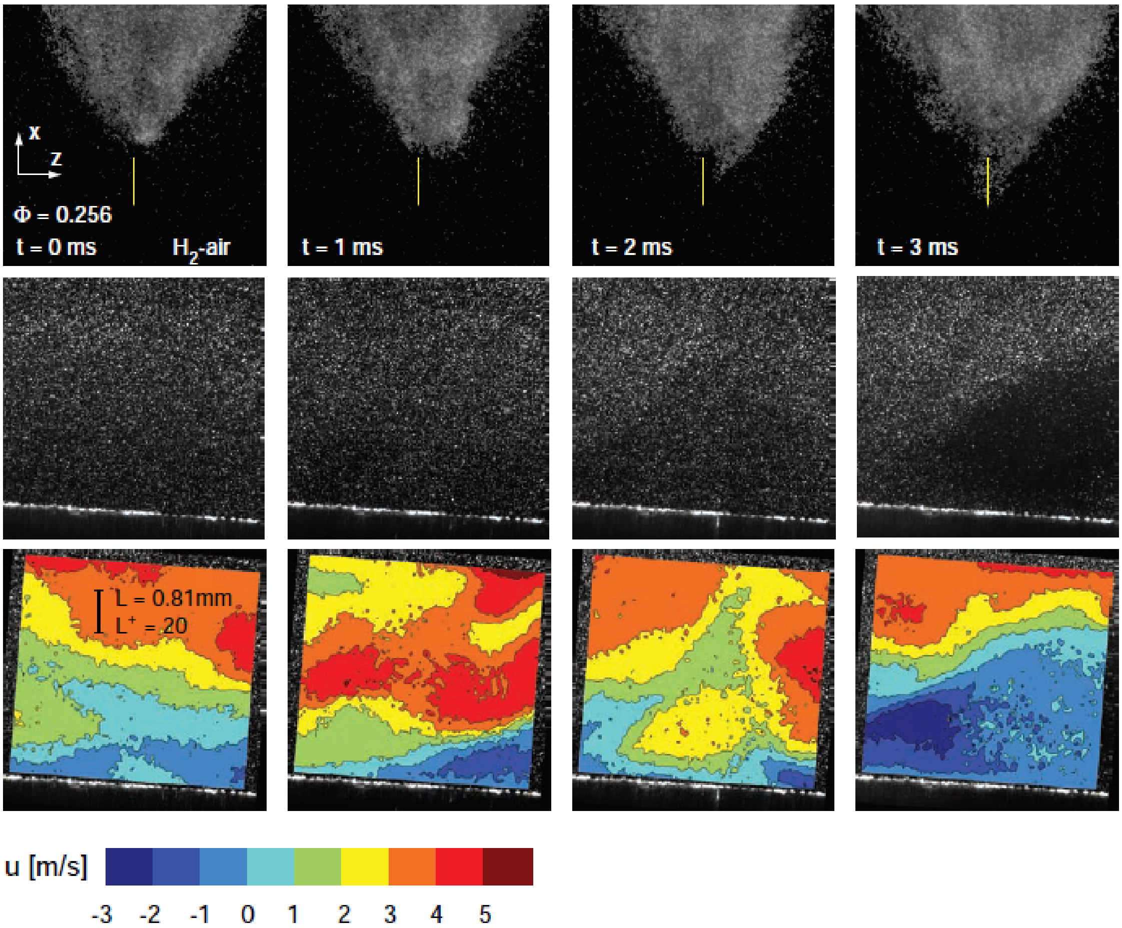


Figure 3.10: Axial velocity contours in front of a turbulent H2-air flame in a 4° diffuser showing a distinct backflow region in front of the flame.

Based on observations like those in Figures 3.9 and 3.10 and the results given in section 3.1.1 in terms of the critical gradient at flashback, Eichler concludes that there is a strong coupling between the flame front and the oncoming flow of reactants. The critical gradient model of Lewis and von Elbe assumes that they are uncoupled. Eichler calls this new observation a "recirculation-assisted [upstream] flame motion during wall flashback" and suggests it is a universal mechanism of flame flashback. Flame flashback is therefore caused by a local separation of the boundary layer in front of the flame and it's upstream propagation is assisted by local upstream flow in the recirculation zone. This applies both to laminar and turbulent flame flashback.

Gruber et al. [22] performed direct numerical simulation on turbulent BLF and observed the same phenomena of recirculation assisted flashback. Figure 3.11 shows contours of a streamwise velocity field around a near wall flame front. A backflow region is clearly visible in front of the flame cusp.

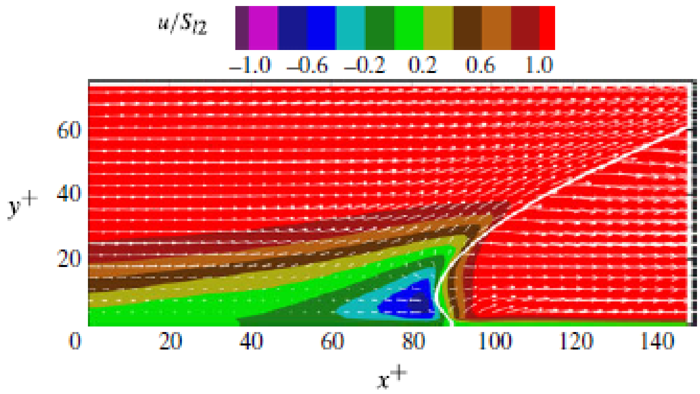


Figure 3.11: Streamwise velocity field (normalized with the laminar flame speed) shown near a flame cusp with a reaction progress variable C = 0.7 (white line). Main reactant flow is from left to right towards the flame front. Results are from the DNS of Gruber et al. [22]

3.2. Boundary layer separation

Eichler's new physical model for BLF discussed in the previous section includes a flame induced flow separation event before the flame propagates upstream in the resulting recirculation region. It is thus necessary to study boundary layer separation to understand flame flashback. In this section the necessary prerequisites for flow separation are introduced along with prediction models for both laminar and turbulent flow separation. In the next section, a TU Munich model to predict the flame flashback for a flame confined in a channel is introduced. The model is based on predicting the onset of flame induced flow separation.

The governing equations for flow in the boundary layer were presented in section 2.1.1. At the wall in a viscous boundary layer the dynamic head $\frac{1}{2}\rho u^2$ is zero due to the no-slip condition so the pressure gradient is only balanced by the shear stress gradient. The streamwise momentum equation (Eq. (2.12)) reduces to:

$$\mu \frac{\partial^2 u}{\partial y^2}\bigg|_{y=0} = \frac{\partial p}{\partial x} \tag{3.2}$$

For a favourable (negative) pressure gradient (FPG) where $(\partial p/\partial x) < 0$ the shear stress gradient at the wall is also negative and $(\partial u/\partial y)$ gradually decreases with increasing y until $(\partial u/\partial y) \approx 0$ at the edge of the boundary layer $y = \delta$. If $(\partial p/\partial x) > 0$ this adverse pressure gradient works to slow down the flow and the shear stress gradient at the wall will be positive. In this case, the velocity gradient $(\partial u/\partial y)$ will initially increase with y before decreasing again towards $(\partial u/\partial y \approx 0)$ at $y = \delta$. Therefore the velocity profile will have an inflection point between $y = 0$ and $y = \delta$. A zero pressure gradient at the wall implies that the velocity profile has an inflection point at the wall.

The adverse pressure gradient case is visualized in Fig. 3.12 where the pressure increase $(\partial p/\partial x) > 0$ eventually leads to separation of the boundary layer.

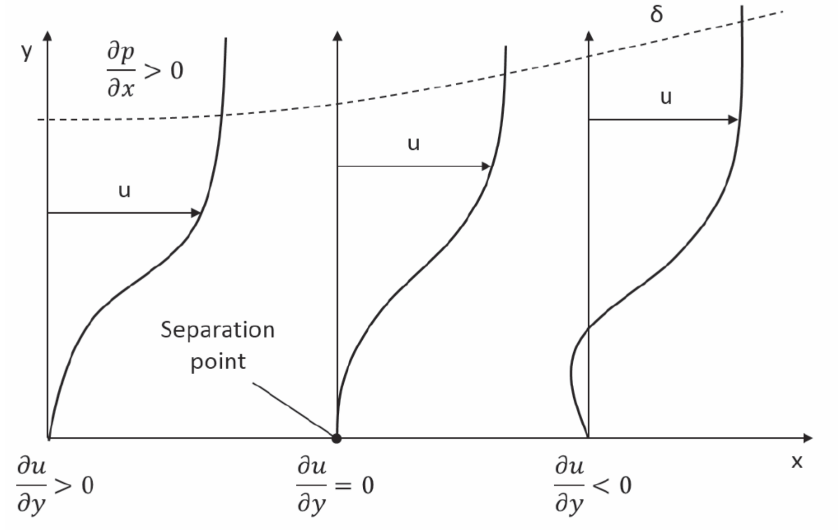


Figure 3.12: Boundary layer separation. Source: Baumgartner [6].

An adverse pressure gradient is a necessary prerequisite for separation. The boundary layer separates when the velocity gradient and shear stress at the wall is zero [32]:

$$\tau_w = \mu \frac{\partial u}{\partial y}\bigg|_{y=0} = 0. \tag{3.3}$$

3.2.1. Stratford's criterion for laminar boundary layer separation

In his doctoral thesis, B.S. Stratford [54] derived a simple formula to predict boundary layer separation for laminar boundary layers. Stratford considered a boundary layer on a flat plate with constant pressure from $x = 0$ to $x = x_m$, illustrated in Fig. 3.13.

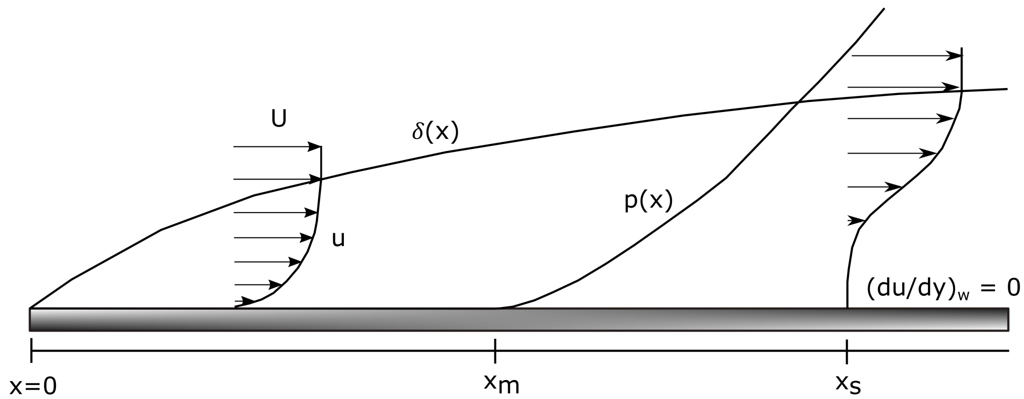


Figure 3.13: Separation of laminar flow over a flat plate due to a sudden increase in pressure.

The subscript m stands for point of minimum pressure. At $x = x_m$ there is a sudden rise in pressure of some arbitrary (but smooth) form. This eventually causes separation of the flow at x_s. The adverse pressure gradient at $x > x_m$ will give rise to an inflection point in the velocity profile as stated before. Stratford divides the boundary layer at the inflection point into an inner sub-layer and an outer layer. In the outer region he states that the pressure should be balanced mostly by inertia while in the inner region the pressure is balanced by viscous shear. Therefore the dynamic head profile keeps its shape in the outer layer but its shape in the inner layer changes with changing pressure. Stratford's division of the boundary layer, and its shape change from zero pressure gradient to adverse pressure gradient flow, is illustrated in Fig. 3.14.

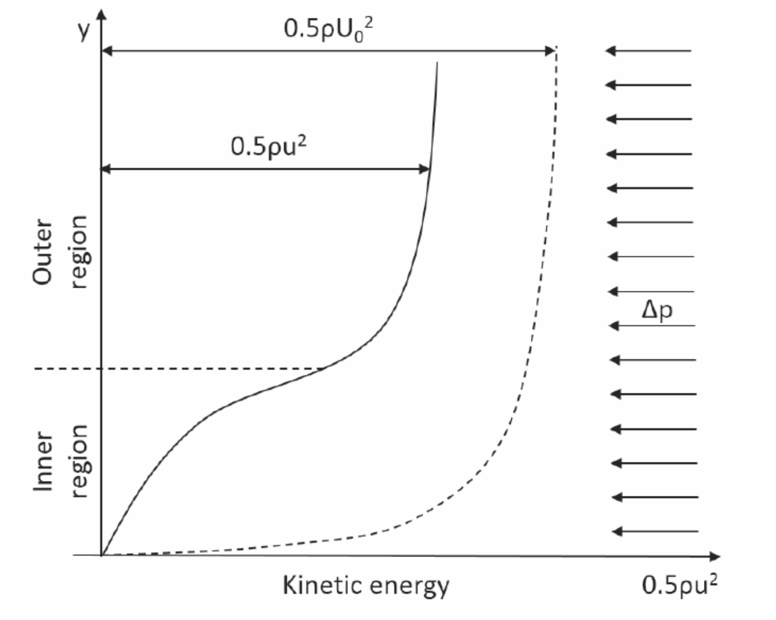


Figure 3.14: The shape change of the dynamic head profile in the boundary layer. The dotted line is for zero pressure gradient flow, the unbroken line is adverse pressure gradient flow. Source: Baumgartner [6].

Figure 3.15 on the next page illustrates how the velocity profile at separation is determined in Stratford's thesis. Stratford obtains an expression for the outer velocity profile by superposition of an inviscid solution for adverse pressure gradient flow and a viscous Blasius boundary layer profile for a zero pressure gradient flow. The total head along a streamline is constant for inviscid flow. For a short interval between x_m and $x > x_m$, this means:

$$\left(\frac{1}{2}\rho u^2\right)_{(x,\psi)} + p = \left(\frac{1}{2}\rho u^2\right)_{(x_m,\psi)} + p_m \tag{3.4}$$

where $\psi = \int_0^y u\text{dy}$. However, since the outer flow is not inviscid, Stratford replaces the velocity profile at x with a visous Blasius flat-plate solution u_b [9] valid for zero pressure gradients:

$$\left(\frac{1}{2}\rho u^2\right)_{(x,\psi)} + p = \left(\frac{1}{2}\rho u_b^2\right)_{(x_m,\psi)} + p_m \tag{3.5}$$

He justifies this by stating that the shape of the outer velocity profile will not be greatly affected by a small sharp rise in pressure.

The inner profile can be derived from Eq. (3.2). To determine the whole velocity profile including the sublayer, the two layers are joined with continuity conditions for u, $\frac{\partial u}{\partial y}$, $\frac{\partial^2 u}{\partial y^2}$ and total flow between the wall and the joining streamline. Stratford ends up with the following criterion for laminar boundary layer separation [32]:

$$C_p\left(x\frac{dC_p}{dx}\right)^2 = 0.0104 \tag{3.6}$$

where

$$C_p(x) = \frac{p(x) - p_m}{\frac{1}{2}\rho U_m^2} = 1 - \left(\frac{U}{U_m}\right)^2 \tag{3.7}$$

is the pressure coefficient which quantifies the portion of dynamic head converted to pressure.

In case of a favourable, negative pressure gradient flow upstream of the adverse pressure gradient flow, the x value needs to be changed for an effective origin $\overline{x}$ in Eq. (3.6) such that the thickness of the boundary layer is correctly accounted for. The effective origin $\overline{x}$ can be defined as:

$$\overline{x} = x - (x_m - \overline{x}_m) \tag{3.8}$$

Here x is the real distance, x_m represents the start of the adverse pressure gradient (at the location of minimum static pressure, hence the subscript m) and $\overline{x}_m$ is an equivalent distance along a zero pressure gradient flow where the boundary layer would have the same momentum thickness θ as in the favourable pressure gradient flow at x_m:

$$\overline{x}_m = \int_0^{x_m} \left(\frac{U}{U_m}\right)^5 dx \tag{3.9}$$

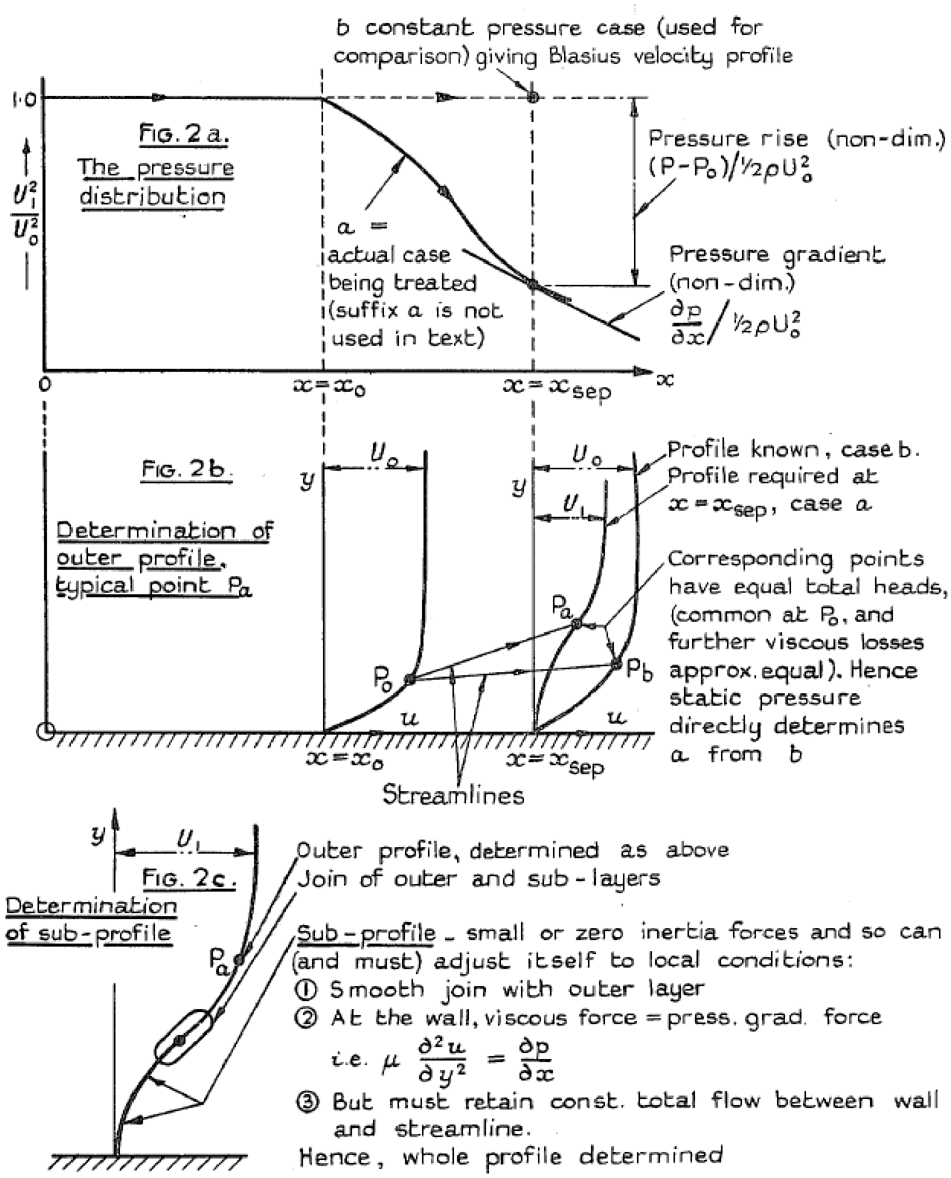


Figure 3.15: Illustration from Stratford's original paper [54] of how the laminar boundary layer velocity profile at separation is determined.

3.2.2. Stratford's criterion for turbulent boundary layer separation

Similarly to the laminar case described above, Stratford [55] also derived a criterion for turbulent boundary layer separation:

$$(2C_p)^{\frac{1}{4}(n-2)}\left(x\frac{dC_p}{dx}\right)^{\frac{1}{2}} = 1.06\beta\left(10^{-6}\mathrm{Re}_x\right)^{\frac{1}{10}} \tag{3.10}$$

resulting from the join of inner and outer velocity profiles at separation. Re_x is the local Reynolds number Ux/ν. The equality applies for $C_p \leqslant \frac{n-2}{n+1}$. The parameter n stems from the use of a power-law approximation for the constant pressure term of the velocity profile in the outer layer. The value of n is a weak function of the Reynolds number according to Stratford [55]. Stratford recommends:

$$n = \log_{10}\mathrm{Re}_s \tag{3.11}$$

such that $6 \leqslant n \leqslant 8$ for $10^6 \leqslant \mathrm{Re}_s \leqslant 10^8$. The parameter β is determined by experiments and varies with $\left(\partial^2 p/\partial x^2\right)_s$, i.e. the curvature of the pressure distribution at the location of separation. Stratford [55] suggested the following values:

$$\begin{aligned}\beta &= 0.66 \text{ for } \left.\frac{\partial^2 p}{\partial x^2}\right|_s < 0\\ \beta &= 0.73 \text{ for } \left.\frac{\partial^2 p}{\partial x^2}\right|_s \geqslant 0\end{aligned} \tag{3.12}$$

According to Kuethe [32], the effective downstream distance $\overline{x}$ from Eq. (3.8) can again be used for x in case of a favorable pressure gradient upstream of x_m, with

$$\overline{x}_m = \int_0^{x_m}\left(\frac{U}{U_m}\right)^3 dx \tag{3.13}$$

or if, in addition, the boundary layer is initially laminar and turbulent transition takes place at $x = x_{tr}$:

$$\overline{x}_m = 38.2\left(\frac{\nu}{U_{tr}x_{tr}}\right)^{\frac{3}{8}}\left[\int_0^1\left(\frac{U}{U_{tr}}\right)^5 d\left(\frac{x}{x_{tr}}\right)\right]^{\frac{5}{8}} x_{tr} + \int_{x_{tr}}^{x_m}\left(\frac{U}{U_m}\right)^3 dx \tag{3.14}$$

Stratford's turbulent separation criterion is simple and was shown to be accurate compared to other available methods by Cebeci et al. [12]. It is conservative, since the predicted pressure rise at separation is 0-10% too low as noted by Stratford himself.

In section 3.3 a BLF model is discussed where the criterion is applied to predict flow separation in front of a turbulent flame stabilized at a wall inside of a duct. It is important to emphasize that both of Stratford's separation criterions are derived for flow over a flat plate with a growing boundary layer. Therefore, in section 4.1 the turbulent boundary layer separation criterion is derived in full and based on the derivation a generalized turbulent separation criterion is presented in section 4.2. The generalized criterion is better suited for application to duct flows.

3.3. TU Munich model to predict confined flame flashback limits

Based on Eichler's [14] new insight into the mechanism of confined wall flashback, Hoferichter [24, 25] developed a model to predict BLF limits for flames confined in a horizontal burner duct. The model is semi-analytical in the sense that it is based on physical intuition but includes a fitting parameter C in the turbulent flame speed closure S_t. Eichler showed that BLF in confined ducts is triggered by a separation of the boundary layer upstream of the tip of the flame front. Hoferichter therefore based the model on Stratford's [55] turbulent separation criterion with $\beta = 0.73$ for $\left(\partial^2 p/\partial x^2\right)_s \geqslant 0$ (positive curvature of the pressure distribution immediately prior to separation) and n = 6 for channel flow:

$$C_p \left(x \frac{dC_p}{dx} \right)^{\frac{1}{2}} = 0.39 \left(10^{-6} \mathrm{Re}_x\right)^{\frac{1}{10}} \tag{3.15}$$

Hoferichter assumed that the flow is fully developed so she removed the dependency of the streamwise coordinate by setting the coefficient 1/10 to zero:

$$C_p \left(x \frac{dC_p}{dx} \right)^{\frac{1}{2}} = 0.39 \tag{3.16}$$

For the pressure distribution in front of the flame, she used the following quadratic expression based on suggestions from Eichler and Baumgartner [6, 14]:

$$p(x) - p(x_m) = \frac{\Delta p}{x_f^2} x^2 \tag{3.17}$$

where x_f is the position of the flame tip. Substitution in Eq. 3.7 yields:

$$C_p(x) = \frac{2\Delta p x^2}{\rho_u U^2 {x_f}^2} \tag{3.18}$$

$$\frac{dC_p(x)}{dx} = \frac{4\Delta p x}{\rho_u U^2 {x_f}^2} \tag{3.19}$$

which inserted into Stratford's criterion (evaluated at x_f) for turbulent boundary layer separation gives:

$$\sqrt{2} \left(\frac{2\Delta p}{\rho_u {U_{FB}}^2} \right)^{\frac{3}{2}} = 0.39 \tag{3.20}$$

Note that the channel centerline velocity U has been replaced by centerline velocity at flashback U_{FB} since the separation is the onset of flashback.

To solve for the centerline velocity at flashback U_{FB} only Δp needs to be determined. It can be found using the standard Rankine-Huginoit conditions for mass and momentum conservation over the flame front [63]:

$$\rho_u u_u = \rho_b u_b \tag{3.21}$$

$$\rho_u {u_u}^2 + p_u = \rho_b {u_b}^2 + p_b \tag{3.22}$$

such that $\Delta p = p_u - p_b = \rho_u {u_u}^2 \left(\frac{\rho_u}{\rho_b} - 1\right)$. Since the flame front is stationary at the onset of flashback, u_u should equal the turbulent burning velocity S_t:

$$\Delta p = p_u - p_b = \rho_u {S_t}^2 \left(\frac{\rho_u}{\rho_b} - 1 \right) \tag{3.23}$$

Hoferichter used *Cantera 2.2* [21] to compute the mixture properties.

An expression for the turbulent burning velocity S_t is needed. Hoferichter used the Damköhler correlation (Eq. (2.30)) but replaced the unstretched laminar burning velocity $S_{l,0}$ with the stretched laminar burning velocity $S_{l,s}$. C should depend on the length scale ratio between the turbulence and the flame [44] but is left as a fitting constant. Hoferichter assumed that the flashback is started at the wall distance of maximum turbulent burning velocity unless the quenching distance of the mixture exceeds it [25]. This location corresponds to the location of maximum streamwise turbulent fluctuations u'. In the two subsections that follow, Hoferichter's approach to modeling the velocity fluctuations u' and the laminar burning velocity S_l is discussed.

3.3.1. Modeling of the turbulent velocity fluctuations

Hoferichter used the following fit for the turbulent velocity fluctuations u' normalized with the shear stress velocity u_τ:

$$\frac{u'}{u_\tau} = a_0 + a_1 \ln(y^+) + a_2 \ln(y^+)^2 + a_3 \ln(y^+)^3 + a_4 \ln(y^+)^4 + a_5 \ln(y^+)^5 \tag{3.24}$$

The coefficients are given in appendix A.1.1. This fit (see Fig. 3.16) is based on experiments for turbulent channel flow and is reasonably accurate for values of $y^+ < 50$.

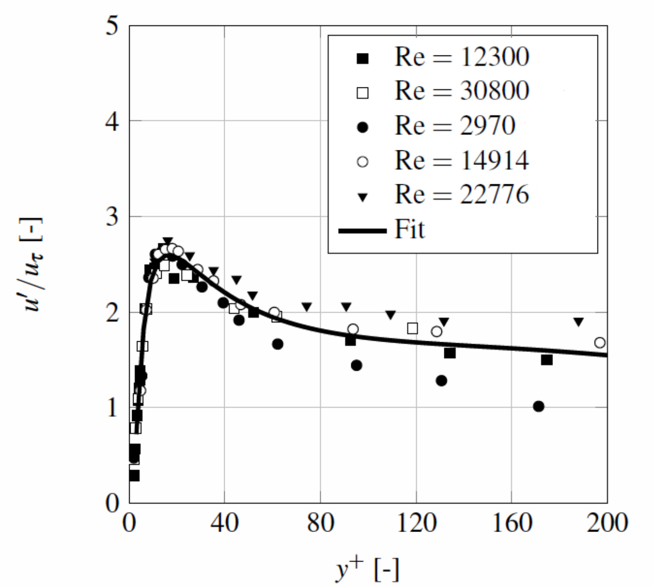


Figure 3.16: Results from experiments for turbulent velocity fluctuations near the walls in channel flows. The fit is given in Eq. (3.24). From Hoferichter [25].

It has a maxima at $y^+ \approx 16$ which indicates that the turbulent burning velocity will also have a maxima at $y^+ \approx 16$.

Finally, Hoferichter related the channel centerline velocity at flashback U_{FB} to the shear stress velocity u_τ via

$$U_{FB} \approx \overline{U}_{FB} + 2.4 u_\tau \tag{3.25}$$

from Pope [47] and

$$\frac{\overline{U}_{FB}}{u_\tau} = \frac{1}{K} \ln\left(\frac{h u_\tau}{\nu}\right) + B - \frac{1}{K} \text{ with } K = 0.41,\ B = 5.0 \tag{3.26}$$

as suggested by White [65] for turbulent channel flow. $\overline{U}_{FB}$ here is the bulk velocity at flashback and h is the height of the burner channel. For turbulent pipe flow Hoferichter assumed $U_{FB} \approx \overline{U}_{FB} + 2.4 u_\tau$ still applies and used the following expression from Schlichting and Gersten [52]:

$$u_\tau{}^2 = \frac{\tau_w}{\rho} = 0.03955 \overline{U}_{FB}^{7/4} \nu^{1/4} h^{-1/4} \tag{3.27}$$

Since u_τ in Eq. (3.25) depends on the output parameter U_{FB} the model needs to be solved iteratively, with e.g. Newton's method or a fixed-point iteration.

Hoferichter's paper [25] carries an important disclaimer:

> "It is assumed that the turbulence parameters upstream of the flame are not highly affected by the presence of the flame if separation is not yet present."

This assumption is necessary to justify the use of cold flow experiments to model the turbulence parameters affecting the flashback prediction. In his MSc thesis, Tober (TU Delft, 2019 [59]) discussed the validity of this assumption. He concluded that the presence of the flame can influence the upstream velocity fluctuations. He did however not include it in his modifications of Hoferichter's model since it only had a minor effect on the flashback limit results. Tober's modifications will be discussed in section 3.3.4.

3.3.2. Modeling of the stretched laminar burning velocity

Hoferichter used Eq. (2.23) to include the effect of flame stretch on the laminar burning velocity. Her treatment of the flame stretch rate κ is explained in section A.1.2 in the appendix.

Hoferichter used Cantera [21] to obtain one dimensional free flame simulation results for the unstretched laminar flame speed $S_{l,0}$ at elevated temperatures using a reaction mechanism by Ó Conaire [40]. She used experimental data for mixtures at room temperature. She represented the results as third order polynomials of the form

$$S_{l,0}(T_u) = b_7 T_u^3 + b_8 T_u^2 + b_9 T_u + b_{10} \tag{3.28}$$

and tabulated the coefficients for different pressures, equivalence ratios and fuels (hydrogen, methane). The free flame simulation results might underestimate $S_{l,0}$ at low burning velocities and for preheated conditions [24].

For the Markstein length, Hoferichter used a derived equation from Bechtold and Matalon [7]:

$$L_M = \delta_F \left(\beta - (\sigma - 1)\,\frac{\gamma_1}{\sigma}\right) \tag{3.29}$$

where δ_F is the laminar flame thickness, $\sigma = (\rho_u/\rho_b)$ is the expansion ratio over the flame front and

$$\beta = \gamma_1 + \frac{1}{2}\mathrm{Ze}\,(\mathrm{Le} - 1)\,\gamma_2 \tag{3.30}$$

which depends on the the Zeldovich number Ze, the Lewis number Le and two parameters γ_1 and γ_2. Hoferichter used

$$\gamma_1 = \sigma,\ \gamma_2 = 1 \tag{3.31}$$

as suggested by Bechtold and Matalon [7]. To calculate the effective Lewis number of the fuel-oxidizer mixture a weighted average of the deficient (D) and excess (E) species is used, also suggested by Bechtold and Matalon [7]:

$$\mathrm{Le} = 1 + \frac{\mathrm{Le}_E - 1 + a\,(\mathrm{Le}_D - 1)}{1 + a} \tag{3.32}$$

with the blending factor

$$a = 1 + \mathrm{Ze}\left(\frac{1}{\phi} - 1\right) \tag{3.33}$$

for fuel-lean or stoichiometric mixtures. The Zeldovich number is defined as

$$\mathrm{Ze} = \frac{E_a\,(T_{\mathrm{ad}} - T_u)}{R T_{\mathrm{ad}}^2} \tag{3.34}$$

with E_a as the global activation energy and R as the universal gas constant $R = 8.314$J/mol/K. The global activation energy is set to a mean value of reported values in literature: $E_a = 30$ kcal/mol = 125604 kJ/mol.

The laminar flame thickness δ_f is estimated using the following expression from Turns [61]:

$$\delta_f = \frac{2\lambda_u}{\rho_u c_{p,u} S_{l,0}} \tag{3.35}$$

valid for Lewis numbers Le = 1.

The calculated Markstein lengths are displayed in Fig. 3.17.

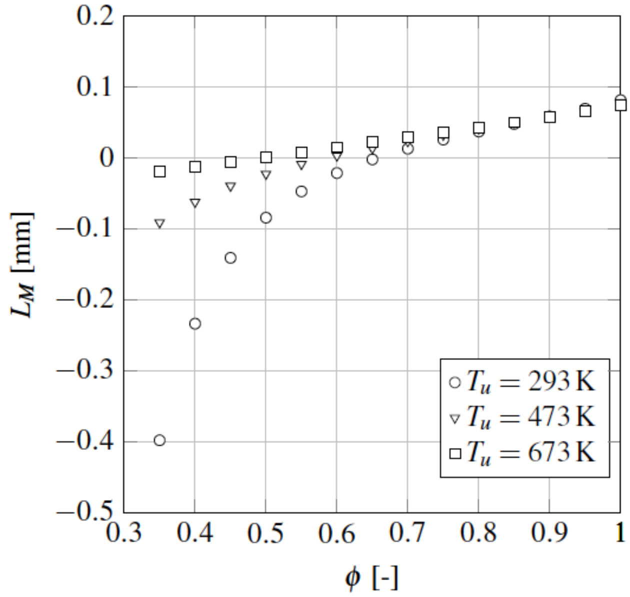


Figure 3.17: Calculated Markstein lengths using Eq. (3.29). Source: Hoferichter [25].

3.3.3. Predicted flashback limits

Hoferichter [24] compared results from her confined flashback model to experimental results from Eichler and Baumgartner [14, 15] for channel and tube burners using lean H_2-air mixtures. Figure 3.18 shows good agreement between Hoferichter's confined flashback predictions and experimental data for the tube burner at atmospheric pressure and temperature.

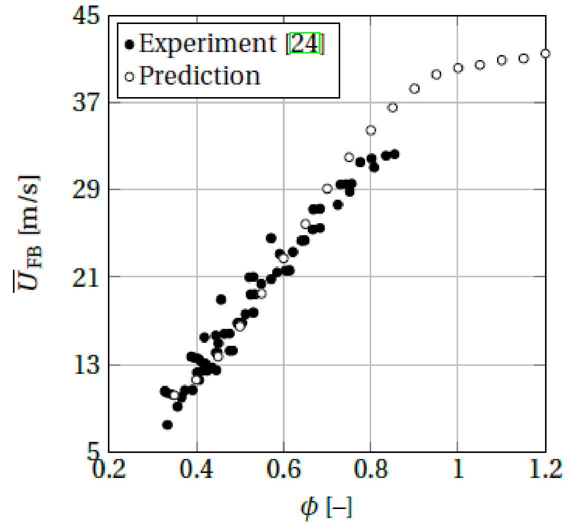


Figure 3.18: Hoferichter's flashback predictions in a $d_h = 40mm$ tube burner compared to experimental data

Figure 3.19 shows results for Eichler's channel burner at different preheating temperatures.

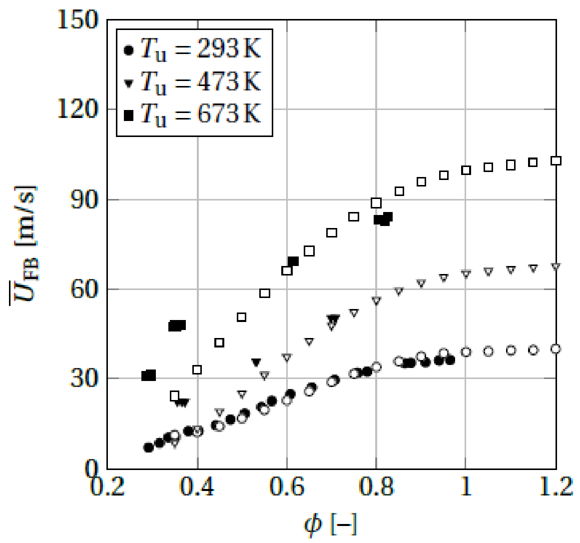


Figure 3.19: Hoferichter's flashback predictions (empty symbols) for a $d_h = 31.5mm$ channel geometry compared to experimental results at different preheating temperatures.

The results show good agreement with the experimental data at room temperature. At elevated temperatures, the model underpredicts below an equivalence ratio of $\phi = 0.6$. Hoferichter mentions several possible reasons for the underprediction at very lean conditions:

- There is high uncertainty in the unstretched laminar burning velocity $S_{l,0}$ at elevated temperatures due to lack of experimental data.
- The calculated Markstein length contains high uncertainty since there is no experimental data for preheated mixtures.

- Lack of accuracy in the Damköhler correlation for turbulent burning velocity S_t

However, Tober [59] corrected the underprediction at low equivalence ratios by accounting for increased turbulent burning velocity due to flame instabilities which form a stable cellular flame structure for hydrogen-air mixtures. This modification and other improvement studies by Tober are discussed in section 3.3.4.

It's worth noting that Hoferichter also plotted the predicted wall distances y_{FB} of flashback initiation, see Fig. 3.20.

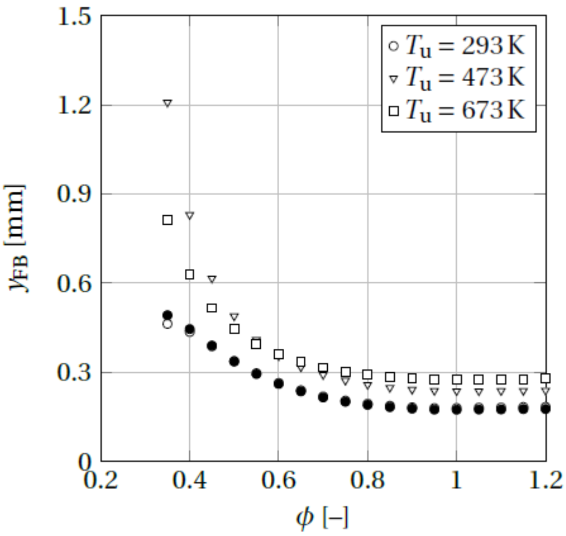


Figure 3.20: Wall distances of flashback initiation according to Hoferichter's [24] confined flashback model. Filled symbols are for the $d_h = 40mm$ tube burner. Empty symbols are for the $d_h = 31.5mm$ channel burner.

The wall distances decrease with equivalence ratio until stoichiometry. Hoferichter concludes that the values of y_{FB} for ambient temperatures are reasonable since they are smaller than Eichler's observed height of the backflow region (y = 0.53mm at $\phi = 0.543$ ($y^+ = 36$) and y = 0.96mm at $\phi = 0.345$ ($y^+ = 35$)).

3.3.4. TU Delft modifications to the flashback model

Tober [59] investigated Hoferichter's confined BLF model in his final thesis. He listed and discussed Hoferichter's assumptions and specifically addressed three phenomena:

- **Turbulence-flame interaction**: Hoferichter assumed that the upstream turbulence was not influenced by the flame.
- **Flame stretch due to anisotropic turbulence**: Hoferichter assumed isotropic turbulence when in reality the channel turbulence is anisotropic.
- **Flame instabilities**: Tober addressed the possibility of flame instabilities leading to cellular flame structures with increased flame speeds.

Based on his investigation Tober recommend two modifications to the model. The first modification was to include the effect of the anisotropy of the turbulence on the flame stretch rate, since Hoferichter assumed isotropic turbulence. Section A.1.3 in the appendix explains how the expression for flame stretch rate changes with anisotropic turbulence. The other modification is based on a paper from Kadowaki [29] on the flame velocity of cellular flames at low Lewis numbers. Tober explained that for lean hydrogen-air flames, a negative Markstein length and a Lewis number less than unity will both contribute to an unstable flame front, the latter due to a thermo-diffusive instability. The Lewis number is the ratio between thermal (α) and mass (D) diffusivities:

$$\mathrm{Le} = \frac{\alpha}{D}$$

Unstable lean hydrogen-air mixtures will however form a stable cellular flame structure. Kadowaki explained that when the Lewis number is unity the turbulent flame speed is proportional to the area of the flame surface.

When the Lewis number decreases, the flame speed increases beyond the area increase. To include this effect, Tober derived the following correlation based on Kadowaki's data:

$$S_{t,\mathrm{corrected}} = \left(0.6052\left(\frac{1}{\mathrm{Le}}^{2}\right) - 1.1314\left(\frac{1}{\mathrm{Le}}\right) + 1.5224\right) S_t \tag{3.36}$$

He called this modification the Lewis number correction. The expression can be used in the range of $0.5 \leq \mathrm{Le} \leq 1$. Above $\mathrm{Le} = 1$ the lean hydrogen-air flames do not show a cellular flame structure. Below $\mathrm{Le} = 0.5$ the cellular flame self-stabilizes and the trend levels off, so the value at $\mathrm{Le} = 0.5$ is used for $\mathrm{Le} \leq 0.5$. A more detailed discussion on the mechanism and effect of flame instabilities is given in section A.1.4 in the appendix.

The results of the modifications are displayed in Figure 3.21 on the next page and compared to both experiments and the default model. The main difference in prediction accuracy is obtained for very lean preheated mixtures. However, the room temperature results are overpredicting at very lean equivalence ratios.

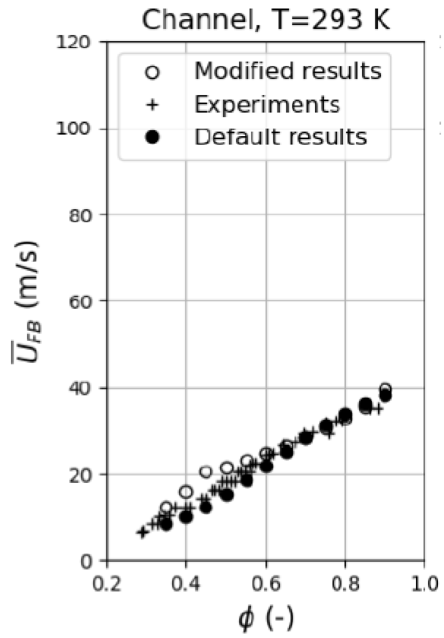
Regarding turbulence-flame interaction, Tober included the effect of a flame on the upstream turbulence by using experimental data from Jainski [28]. Jainski showed that the turbulence fluctuations increased in the presence of a flame resulting in higher flashback propensity. The tuning constant needed in the model decreased accordingly from $C = 2.3$ to $C = 1.9$. Tober did however not recommend using this modification, perhaps due to the difficulty of obtaining the correct flame affected turbulence fluctuations for other cases.

Tober also tried using the following turbulent flame speed correlation from Lin et al. [35] instead of the simple Damköhler closure given in Eq. (2.30):

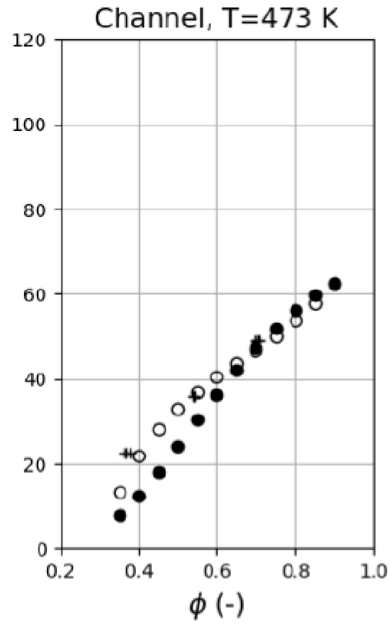
$$\frac{S_t}{S_{l,0}} = 10.5 \times \mathrm{Le}^{-0.82}\left(\frac{u'}{S_{l,0}}\right)^{0.45}\left(\frac{\Lambda}{\delta_f}\right)^{-0.41}\left(\frac{P}{P_{\mathrm{ref}}}\right)^{0.75}\left(\frac{T}{T_{\mathrm{ref}}}\right)^{-1.33} \tag{3.37}$$

In short, it did not improve the model.

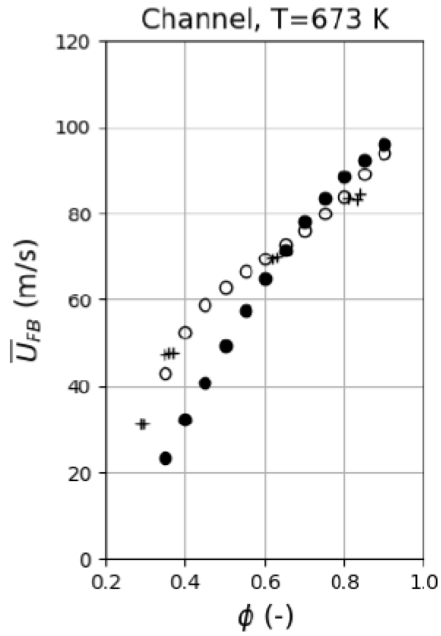
In section 4, specifically subsection 4.3.1, the validity of the prediction model for very lean mixtures is discussed. The calculated Markstein length rises to unphysical values for very lean mixtures causing the stretched flame speed to be wildly overestimated. The result is that including flame stretch effects on the laminar flame speed ruins the prediction accuracy at the leanest equivalence ratios while improving it only slightly at higher equivalence ratios.



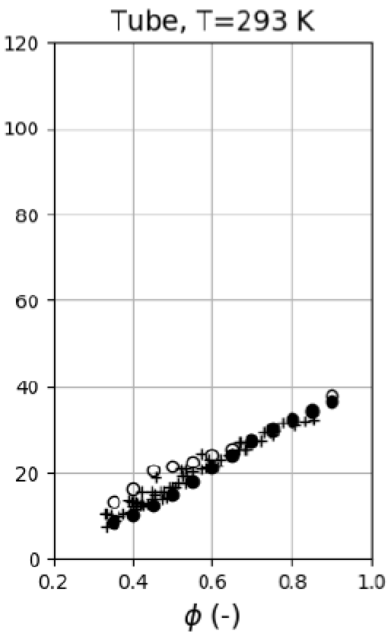
(a) Channel, room temperature mixture. The modified results overpredict in the leaner half.



(b) Mixture preheated to 473K. The modified results predict better at lower equivalence ratios.



(c) Mixture preheated to 673K. The modified results predict better, especially for leaner mixtures.



(d) Tube, room temperature mixture. The modified results seem to overpredict in the leaner half.

Figure 3.21: Results of the flashback model including Tober's (TU Delft) modifications compared to the default model and experiments (TU Munich [14, 25]). The modifications improve the prediction accuracy for preheated mixtures but cause overprediction for very lean room temperature mixtures. Source: Tober's MSc thesis [59].

4

A generalized turbulent boundary layer separation criterion

The BLF model is built for confined flames in fully developed channel and tube flow. However, Stratford's turbulent boundary layer separation criterion was designed for flow over an airfoil. In this chapter, the criterion will be derived in detail to investigate how it should optimally be applied in the BLF model. A case will be made for the claim that Hoferichter's application of the criterion results in an inaccurate representation of the mean velocity profile at separation and thus inaccurate predictions of the flame backpressure magnitude at separation. Then a generalized criterion is presented and validated. Finally, the validity of the BLF model at low equivalence ratios is discussed in the context of the Markstein length and flame stretch effects.

4.1. Full derivation of Stratford's criterion for turbulent boundary layer separation

Stratford's turbulent separation criterion was already introduced in section 3.2.2. Equation (3.10) relates the pressure recovery factor C_p and its derivative at the maxima of the pressure profile to the shape of a mean velocity profile with zero wall shear, i.e. the shape of the mean velocity profile at separation. With a known pressure profile the coefficient of pressure C_p and its derivative $\mathrm{d}C_p/\mathrm{dx}$ can be calculated to determine if the boundary layer should separate or not.

It is interesting to take a closer look at Stratford's criterion and its assumptions. Stratford assumes that the outer layer of a turbulent boundary layer will keep its shape for a short distance downstream of a sudden adverse pressure gradient due to the dominating inertia and negligible shear. This idea is illustrated in Fig. 4.1.

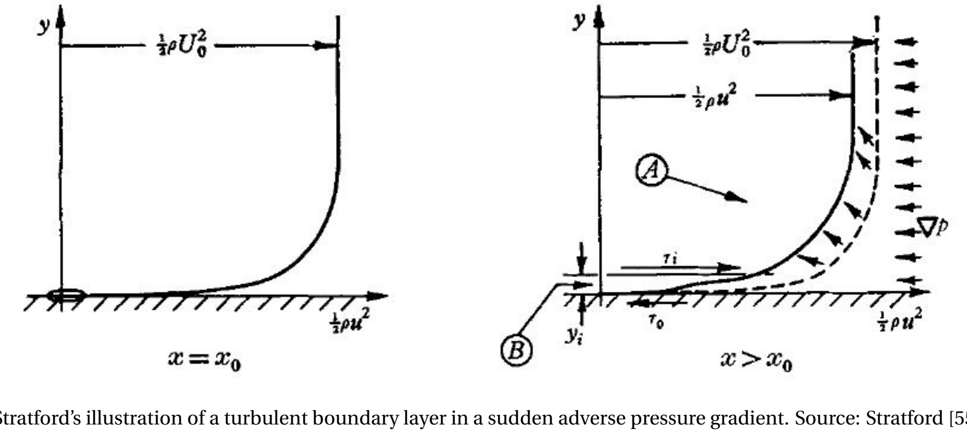


Figure 4.1: Stratford's illustration of a turbulent boundary layer in a sudden adverse pressure gradient. Source: Stratford [55].

The separation condition is derived by equating the total pressure on a streamline

$$\psi = \int_0^y u \, \mathrm{dy}$$

in the outer layer ($\psi \geqslant \psi_i$) for an adverse pressure gradient flow u and an imaginary zero pressure gradient flow u' (not to be confused with the fluctuating part of the turbulent velocity). The dynamic head of the adverse pressure gradient flow is equal to the dynamic head of the zero pressure gradient flow minus the rise in static pressure:

$$\frac{1}{2}\rho u^2_{(x,\psi)} = \frac{1}{2}\rho u'^2_{(x,\psi)} - (p - p_0)\ ,\ \psi \geqslant \psi_i \tag{4.1}$$

Stratford uses the 1/n-th power law for the outer layer of the zero pressure gradient turbulent boundary layer:

$$\frac{u'}{U_0} = \left(\frac{y'}{\delta'}\right)^{\frac{1}{n}} \tag{4.2}$$

where U_0 is the far-field velocity at the point of minimum pressure and δ' is the thickness of the boundary layer. From equations 4.1 and 4.2 Stratford derives:

$$C_p = \left(\frac{y'}{\delta'}\right)^{\frac{2}{n}} \left(1 - \frac{u^2}{u'^2}\right) \tag{4.3}$$

with

$$C_p = \frac{p - p_0}{\frac{1}{2}\rho U_0^2} \leqslant \left(1 - u^2/u'^2\right)$$

since $y' \leqslant \delta'$. The inner profile at separation is derived from mixing length theory, starting with the shear stress for positive $\partial u/\partial y$:

$$\tau = \rho\,(Ky)^2\,(\partial u/\partial y)^2 \tag{4.4}$$

where $K = 0.41$ is the von Kármán constant and Ky is the mixing length in the wall boundary layer. Integrating Eq. (3.2) gives:

$$\tau = y\frac{\partial p}{\partial x}$$

at separation when the shear stress at the wall τ_0 is zero. This equation is assumed to be valid in the viscous layer close to the wall. Equating these two expressions gives:

$$\frac{1}{2}\rho u^2 = \frac{2}{(0.41\beta)^2}\frac{\partial p}{\partial x} y\ ,\ \left(\tau_0 = 0, y < y_i\right) \tag{4.5}$$

using the no-slip condition at the wall. The von Kármán constant has been multiplied by β, an empirical factor added to describe the effect of the adverse pressure gradient on the mixing length.

Stratford finds the following two equalities:

$$\left(\frac{y'}{\delta'}\right)^{\frac{2}{n}} = \left(\frac{3(0.41\beta)^4}{(n+1)\left(n\delta'\frac{dC_p}{dx}\right)^2}\right)^{\frac{1}{n-2}} \tag{4.6}$$

$$\frac{u^2}{u'^2} = \frac{3}{n+1} \tag{4.7}$$

by joining the inner and outer mean velocity profiles from Eq. (4.2) and Eq. (4.5) using the following two expressions: $\psi\,(\partial u/\partial y)^3$ and $u^2/\left(\psi \partial u/\partial y\right)$.

By substituting Eq. (4.6) and Eq. (4.7) into Eq. (4.3) Stratford arrives at the following equation valid for the mean velocity profile at separation:

$$C_p = \left(\frac{3(0.41\beta)^4}{(n+1)\left(n\delta'\frac{dC_p}{dx}\right)^2}\right)^{\frac{1}{n-2}} \left(1 - \frac{3}{n+1}\right)$$

Rearranging gives:

$$C_p^{\frac{1}{4}(n-2)}\left(\delta'\frac{dC_p}{dx}\right)^{\frac{1}{2}}=\left(\frac{3(0.41\beta)^4}{(n+1)n^2}\right)^{\frac{1}{4}}\left(1-\frac{3}{n+1}\right)^{\frac{1}{4}(n-2)} \tag{4.8}$$

which is Stratford's criterion without any expression for the boundary layer thickness δ'.

To include an expression for the thickness of the boundary layer, Stratford uses a correlation for flow over a flat plate citing Goldstein [20] and Schlichting [50]:

$$\delta'=\frac{(n+1)(n+2)}{n}\theta' \tag{4.9}$$

with

$$\theta'=0.036x\mathrm{Re}_x^{-\frac{1}{5}} \tag{4.10}$$

for the momentum thickness as a function of the distance x from where the boundary layer starts to grow and the local Reynold's number Re_x.

Inserting this expression for δ' into Eq. (4.8) leads to

$$C_p^{\frac{1}{4}(n-2)}\left(x\frac{dC_p}{dx}\right)^{\frac{1}{2}}=\left(\frac{3(0.41\beta)^4}{0.036^2}\right)^{\frac{1}{4}}\mathrm{Re}_x^{\frac{1}{10}}\left(\frac{(n-2)^{\frac{1}{4}(n-2)}}{(n+1)^{\frac{1}{4}(n+2)}(n+2)^{\frac{1}{2}}}\right) \tag{4.11}$$

Stratford then uses

$$\frac{(n-2)^{\frac{1}{4}(n-2)}}{(n+1)^{\frac{1}{4}(n+2)}(n+2)^{\frac{1}{2}}}=\frac{1}{10.7\times(2.00)^{\frac{1}{4}(n-2)}}$$

which he states is accurate within 1% for $6\leqslant n\leqslant 8$ to arrive at his final separation criterion:

$$(2C_p)^{\frac{1}{4}(n-2)}\left(x\frac{dC_p}{dx}\right)^{\frac{1}{2}}=1.06\beta(10^{-6}\mathrm{Re}_x)^{\frac{1}{10}} \tag{4.12}$$

Note that $C_p\leqslant(n-2)/(n+1)$ always applies. This limitation results from $y'\leqslant\delta'$ in Eq. (4.2).

4.2. A generalized separation criterion

Hoferichter used Stratford's turbulent boundary layer separation criterion (Eq. (4.12)) with $\beta = 0.73$ as recommended by Stratford for $\left(\partial^2 p/\partial x^2\right)_s \geqslant 0$ (positive curvature of the pressure distribution immediately prior to separation) and n = 6 for fully developed channel flow. She also removes the local Reynolds number dependency by setting the exponent $1/10$ to zero:

$$C_p \left(x \frac{dC_p}{dx}\right)^{\frac{1}{2}} = 0.39 \tag{4.13}$$

Hoferichter then evaluates the criterion at $x = x_f = 0.01$ m since Eichler's DNS of laminar flame flashback in channels found the extent of the recirculation region to be approximately 10mm [14].

Interestingly, by using Eq. (3.16) and $x = x_f = 0.01$ m it is implicitly assumed that the value of the boundary layer thickness is $\delta' = 2.12 \times 10^{-4}$ m. The criterion assumes the turbulent boundary layer is growing on a flat plate and that the turbulent boundary layer thickness is captured by equations 4.9 and 4.10 combined:

$$\delta' = \frac{(n+1)(n+2)}{n} \times 0.036x\mathrm{Re}_x^{-\frac{1}{5}}$$

rewritten with exponent $a = -1/5$:

$$\delta' = \frac{(n+1)(n+2)}{n} \times 0.0023x\left(10^{-6}\mathrm{Re}_x\right)^a$$

Inserting $n = 6$, $x = x_f = 0.01$ m and setting the exponent a to zero gives:

$$\delta' = 2.12 \times 10^{-4}\ \mathrm{m}$$

The 1/n-th law (Eq. (4.2)) should have a matching pair of far-field velocity U_0 and boundary layer thickness δ'. Since Hoferichter uses the centerline velocity for U_0, the boundary layer thickness should be the channel halfwidth or the pipe radius. In fact, the 1/n-th law was originally introduced by J. Nikuradse for turbulent boundary layers in fully developed pipe flow where the radius of the pipe was used as the boundary layer thickness [51]. Figure 4.2 on the next page illustrates this point. It can be seen that using $\delta' = 2.12 \times 10^{-4}$ m results in an overestimation of the mean velocity in the outer layer (red). By using the correct value the outer layer is well represented above $y^+ \approx 30-50$ (blue). It is therefore recommended to instead use Eq. (4.8):

$$C_p^{\frac{1}{4}(n-2)} \left(\delta' \frac{dC_p}{dx}\right)^{\frac{1}{2}} = \left(\frac{3(0.41\beta)^4}{(n+1)n^2}\right)^{\frac{1}{4}} \left(1 - \frac{3}{n+1}\right)^{\frac{1}{4}(n-2)}$$

along with a matching pair of centerline velocity and channel halfwidth (or pipe radius) for U_0 and δ' respectively, along with an appropriate value for the fitting parameter n. Using $n = 6$ and $\beta = 0.73$ as before gives:

$$C_p \left(\frac{H}{2} \frac{dC_p}{dx}\right)^{\frac{1}{2}} = 0.0565 \tag{4.14}$$

$$C_p \left(\frac{D}{2} \frac{dC_p}{dx}\right)^{\frac{1}{2}} = 0.0565 \tag{4.15}$$

for a channel with height H and a pipe with diameter D, respectively. The coefficient of pressure is:

$$C_p = \frac{p - p_0}{\frac{1}{2}\rho U_{0,\mathrm{centerline}}^2} \tag{4.16}$$

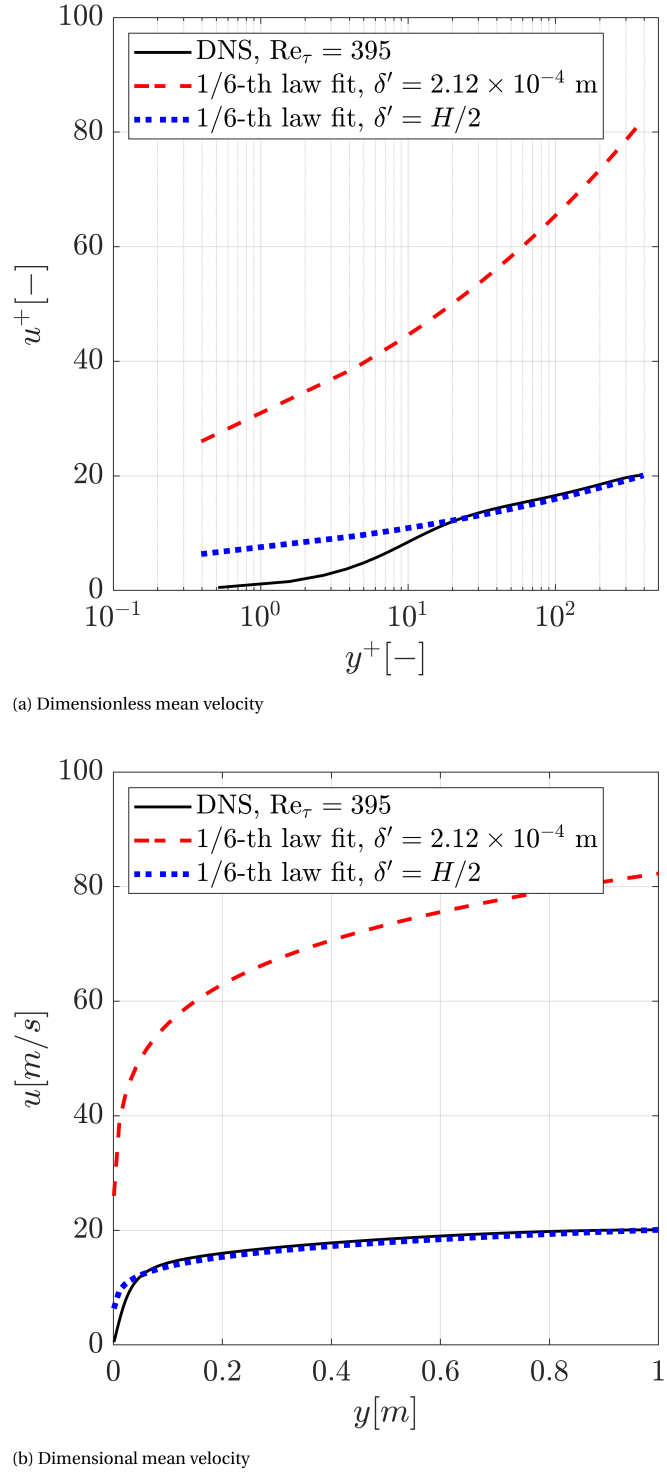


Figure 4.2: The 1/6-th-power law applied to a turbulent boundary layer in a channel. The red dashed line is how it is applied in Eq. (4.13) by Hoferichter in the BLF model. The blue dotted line is how it can be applied using Eq. (4.8) and anchored correctly, i.e. using a matching pair of boundary layer thickness δ' and "far-field" velocity U_0. The DNS data is from Pecnik et al. [42].

4.3. Model duplication using the generalized separation criterion

The BLF model was implemented in code using Python 3.7. Tober's code for his improved BLF model (section 3.3.4) was used as a template and the generalized separation criterion (Eq. (4.8)) was implemented:

$$C_p^{\frac{1}{4}(n-2)}\left(\delta'\frac{dC_p}{dx}\right)^{\frac{1}{2}} = \left(\frac{3(0.41\beta)^4}{(n+1)n^2}\right)^{\frac{1}{4}}\left(1-\frac{3}{n+1}\right)^{\frac{1}{4}(n-2)}$$

This resulted in a third iteration of the BLF model. The code is given in appendix A.2.2. The results from all three iterations are displayed and compared to experiments in Fig. 4.3.

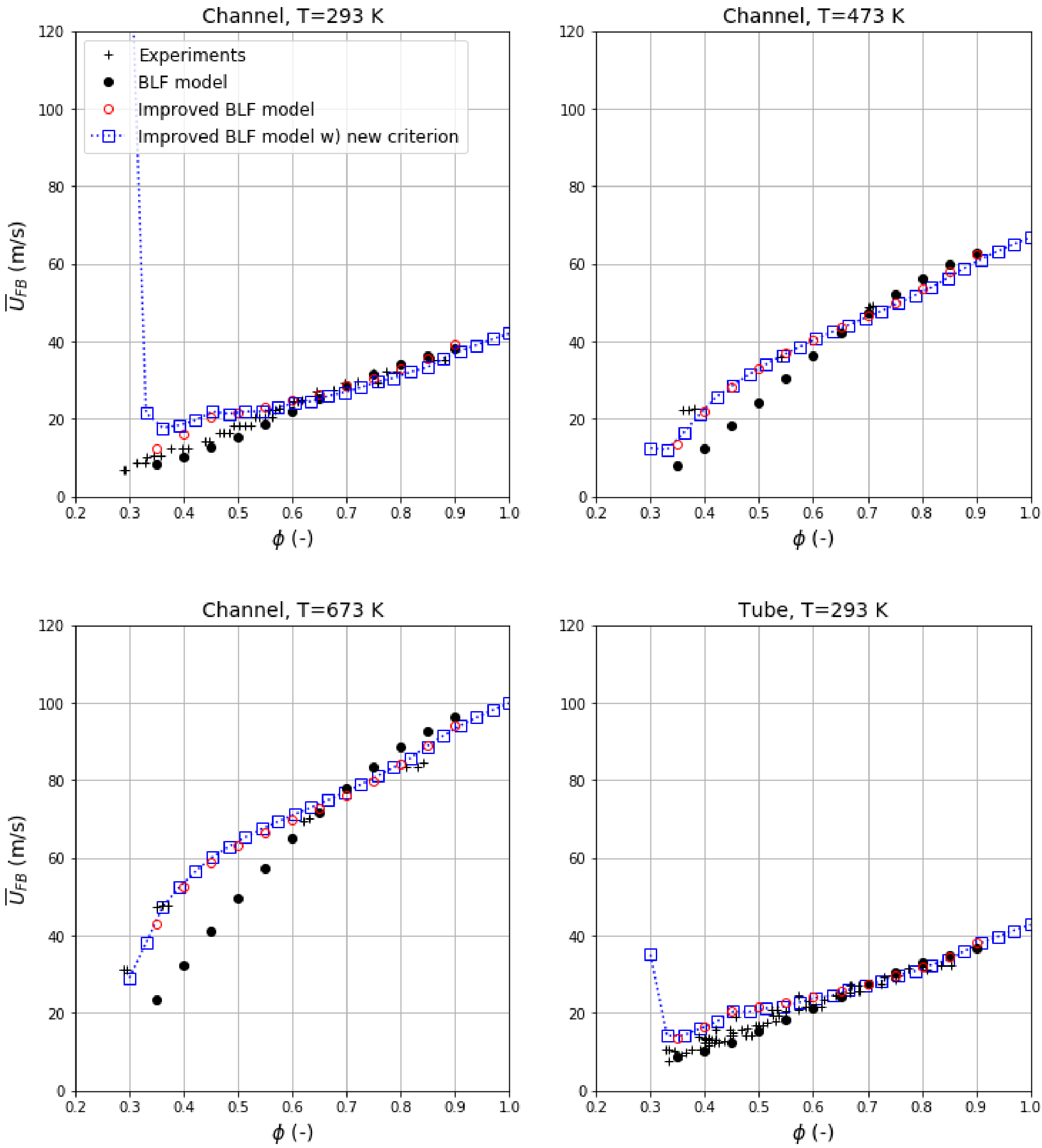


Figure 4.3: Results from the BLF model with the generalized criterion (new equality) compared to previous results. Values for the fitting parameters are given in Table 4.1.

Apart from using the generalized criterion, some minor changes were also made compared to Tober's original code:

1. A correct expression for the hydraulic diameter in the channel is implemented:

$$D_{h,\text{channel}} = \frac{4 \times \text{area}}{\text{perimeter}} = \frac{4wh}{2(w+h)}$$

   It was previously overestimated with an incorrect expression: $2\left(wh/\pi\right)^{\frac{1}{2}}$

2. The expansion ratio in Eq. (3.23) is expressed as $\sigma = \rho_u/\rho_b$ instead of T_{ad}/T_u as this was an unnecessary substitution.

Both changes affect the results but the effects are expected to be minor.

Table 4.1 shows the values used for C in the Damköhler flame speed closure (Eq. (2.30)):

$$S_t = S_{l,s}\left(1 + C\left(\frac{u'}{S_{l,s}}\right)^{0.5}\right)$$

Table 4.1: Best fit values for C (in Eq. (2.30)). The outer boundary layer is fitted with the 1/nth-power law (Eq. (4.2)).

	Channel		**Pipe**	
	n	**C**	**n**	**C**
BLF model	6	2.3	6	2.3
Improved BLF model	6	2.0	6	2.0
Improved BLF model w) generalized criterion (n constant)	6	0.87	6	0.7
Improved BLF model w) generalized criterion (C constant)	7	1.05	8	1.05

It also shows which n was used in the 1/n-th power law fit (Eq. (4.2)):

$$\frac{u'}{U_0} = \left(\frac{y'}{\delta'}\right)^{\frac{1}{n}}$$

The results are now given down to $\phi = 0.30$ to show how the model deviates at low equivalence ratios for room temperature. Instead of following the experimental results, the flashback limits shoot up due to overpredicted flame speeds. This is due to the calculated Markstein length decreasing rapidly at low equivalence ratios for room temperatures (see Fig. 4.4 on the next page).

Otherwise the results using the generalized criterion are quite similar to results for Tober's improved BLF model. However, the values for C have decreased considerably, from 2.0 to 0.87 for the channel and 0.70 for the pipe, meaning the computed pressure difference Δ_p over the flame front at flashback has decreased.

The n in the 1/nth-power law can be varied in the generalized criterion. The results can be fitted identically well using the same value for C for both geometries but changing n. An example is $n = 7$ for the channel and $n = 8$ for the pipe with $C = 1.05$.

At T = 293 K and $\phi = 0.35$ in the channel, the model (blue) is overpredicting even more than previously (red). This is due to the correct expression for the hydraulic diameter D_h implemented which gives a ca. 50% lower D_h, resulting in a smaller value for the turbulence length scale in Eq. (A.6) and thus a larger flame stretch rate κ. As the equivalence ratio is lowered the stretched laminar flame speed $S_{l,s}$ increases due to the rapidly falling Markstein length. Since the flame stretch rate κ has increased this effect is now larger. The same results with the incorrect expression for the hydraulic diameter are shown to follow the previous results (red) in Fig. 4.5 for comparison.

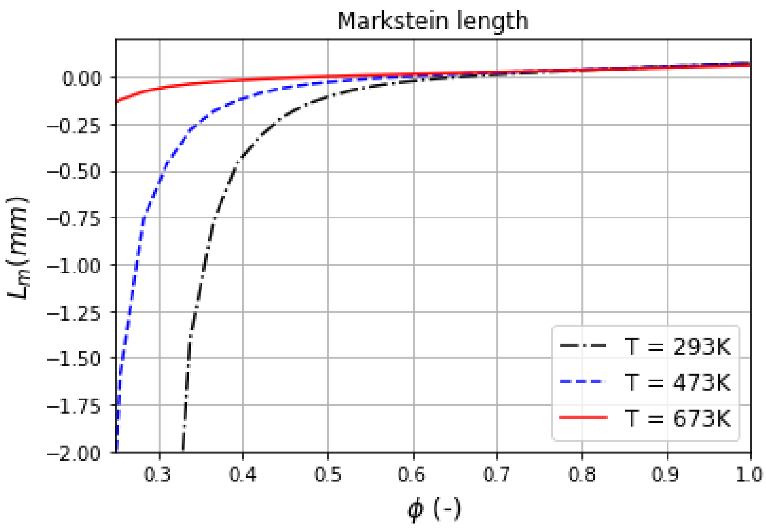


Figure 4.4: Markstein length calculated in the flashback model for different inlet temperatures.

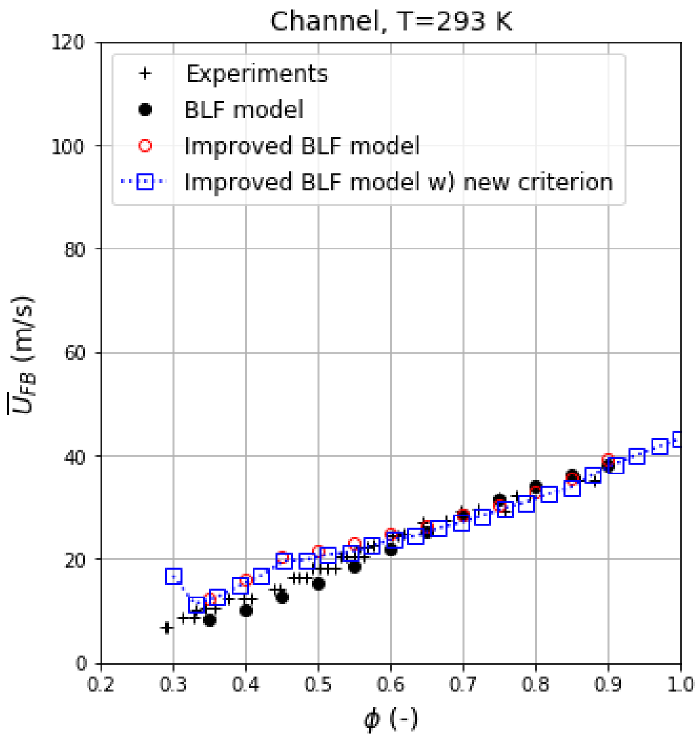


Figure 4.5: Results from the BLF model with the generalized criterion using the incorrect hydraulic diameter from the previous iterations of the model.

4.3.1. The Markstein length and model validity at low equivalence ratios

For low equivalence ratios, Tober's improvements did improve the results at elevated temperatures but they resulted in slight overprediction at room temperature. Tober explains in his thesis that the two modifications he makes, the anisotropic flame stretch and the Lewis number correction cause underprediction and overprediction respectively at low equivalence ratios. The result is a balance between the two effects which does not follow the experimental results as well at room temperatures as it does for elevated temperatures. Due to this discrepancy it can be noted that a different method is used to compute the unstretched laminar burning velocity $S_{l,0}$ for room temperatures on one hand and elevated temperatures on the other hand. Since Hoferichter had experimental results available for room temperatures only, she used Cantera [21] at elevated temperatures to calculate flame speeds using a reaction mechanism by Ó Conaire [40]. She tabulated all flame speed results as coefficients for the polynomial in Eq. (3.28). Figure 4.6 on the next page shows a comparison between the laminar flame speed used in the BLF models (from Hoferichter's polynomial) and flame speeds acquired by using Cantera. At elevated temperatures the two methods agree which is expected since Hoferichter derived the coefficients from the same Cantera simulations. At room temperature the Cantera results are underpredicting at low equivalence ratios and slightly overpredicting at high equivalence ratios. Hoferichter did mention that due to this observation, it is likely that the flame speed is underpredicting at low equivalence ratios for elevated temperatures.

Therefore, due to the discrepancy observed in flashback limits, it is interesting to see what happens with the flashback limits if the Cantera results are also used for room temperature mixtures. The results of this modification can be seen in Fig. 4.7. The tuning constants are unchanged and are given in Table 4.1. Compared to Fig. 4.3, starting at $\phi = 0.6$ (where the Markstein length is negative and starting to increase the stretched laminar flame speed) the results follow the experiments better until ca. $\phi = 0.4$ where the predictions rise rapidly. The lower values for the unstretched laminar flame speed $S_{l,0}$ from Fig. 4.6a will indeed cause two competing effects:

1. A decreased value for unstretched laminar flame speed $S_{l,0}$ should decrease the turbulent flame speed through Eq. (2.23):

   $$S_{l,s} = S_{l,0} - L_M \kappa$$

   and

   $$S_t = S_{l,s}\left(1 + C\left(\frac{u'}{S_{l,s}}\right)^{0.5}\right)$$

2. However, decreasing $S_{l,0}$ will also increase the absolute value of the calculated Markstein length L_M:

   $$L_M = \delta_F\left(\beta - (\sigma - 1)\frac{\gamma_1}{\sigma}\right)$$

   through the calculated flame thickness δ_F (Eq. (3.35)):

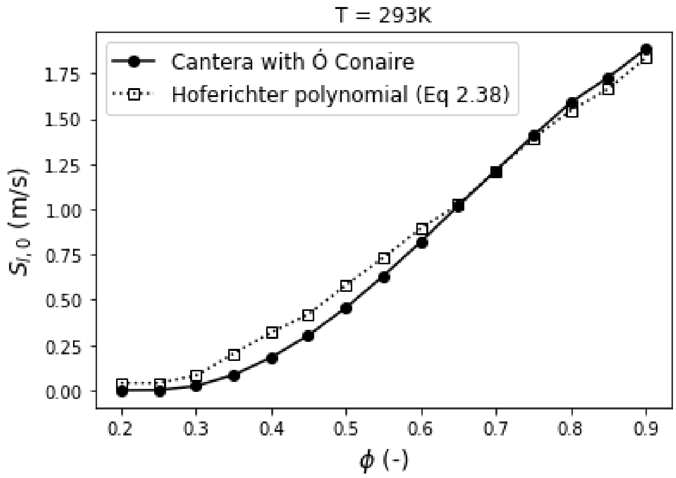
   $$\delta_F = \frac{2\lambda_u}{\rho_u c_{p,u} S_{l,0}}$$

   which will increase the stretched flame speed and therefore turbulent flame speed.

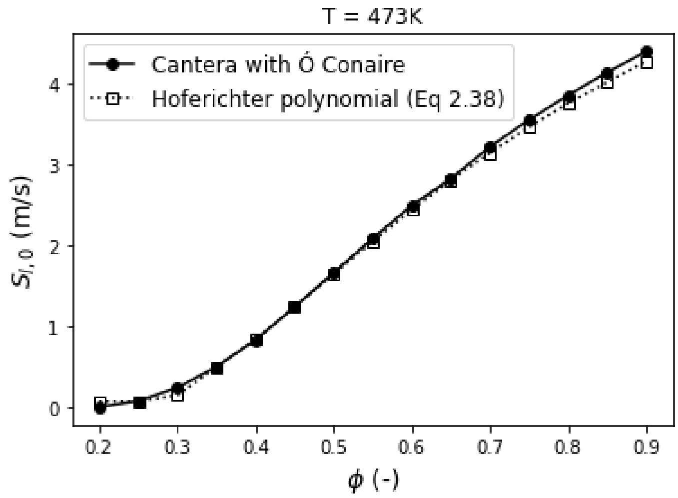
Equation (3.35) is from Turns [61] and assumes a Lewis numer of Le = 1. It is used widely in literature [24]. Figure 4.8 shows that as the laminar flame speed tends towards zero at low equivalence ratios, the calculated flame thickness increases to unphysical values on the order of 1 mm resulting in a sharp drop in calculated Markstein lengths. It also shows the calculated Lewis number as a function of the hydrogen-air mixture. The Lewis number decreases from 1 to approximately 0.4 at an equivalence ratio of 0.3. Using Cantera to calculate the laminar flame speed for all temperatures will improve the flashback predictions between $\phi = 0.40$ and $\phi = 0.60$, but due to the competing effects of changing the laminar flame speed the model is now only predicting adequately down to $\phi = 0.40$ instead of $\phi = 0.35$. The unphysical flame thickness and Markstein length predictions and the resulting excessive flame stretch below $\phi \approx 0.40$ causes very high stretched flame speeds.

By setting the Markstein length to zero, $L_M = 0$, the effect of using the unstretched laminar flame speed instead of the stretched flame speed can be investigated:

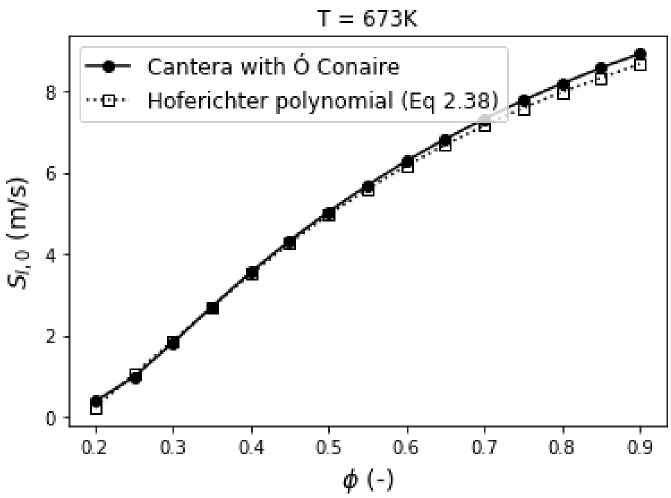
$$S_{l,s} = S_{l,0} - L_M \kappa = S_{l,0}$$



(a)



(b)



(c)

Figure 4.6: Comparison of unstretched laminar flame speeds calculated by Cantera and from the polynomial of Hoferichter.

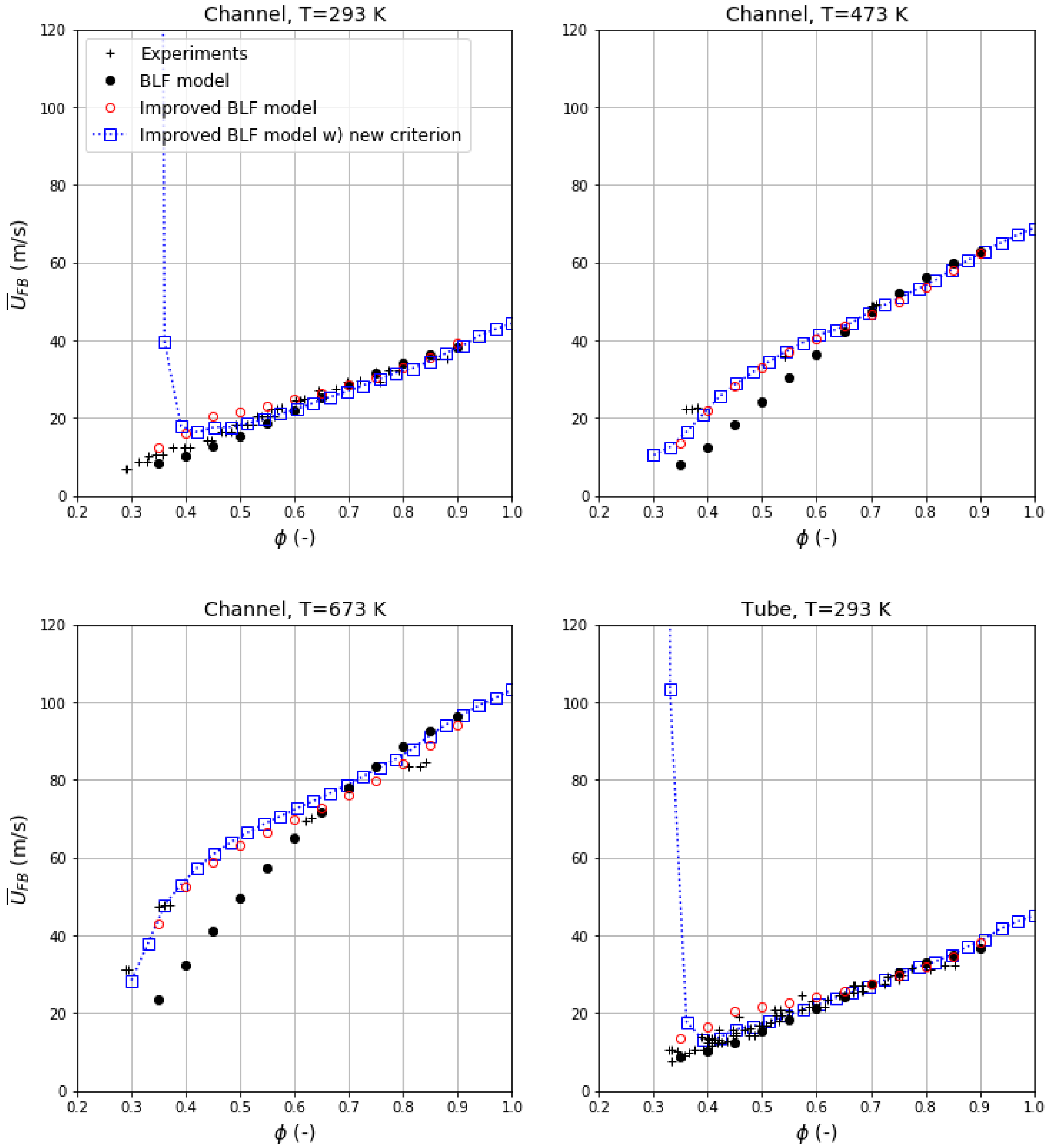
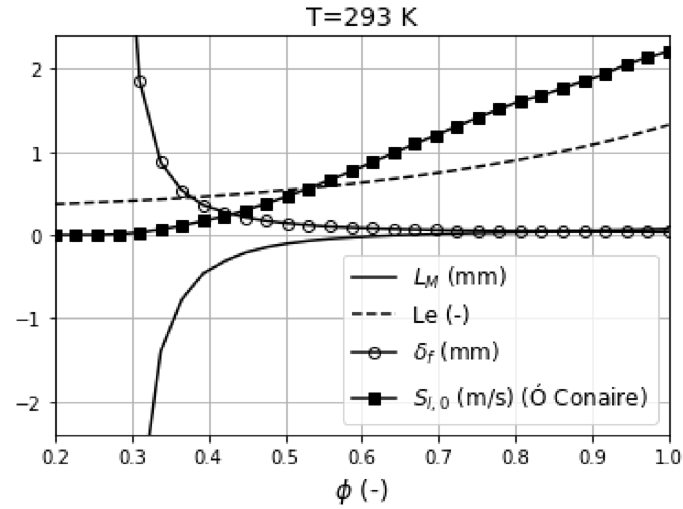
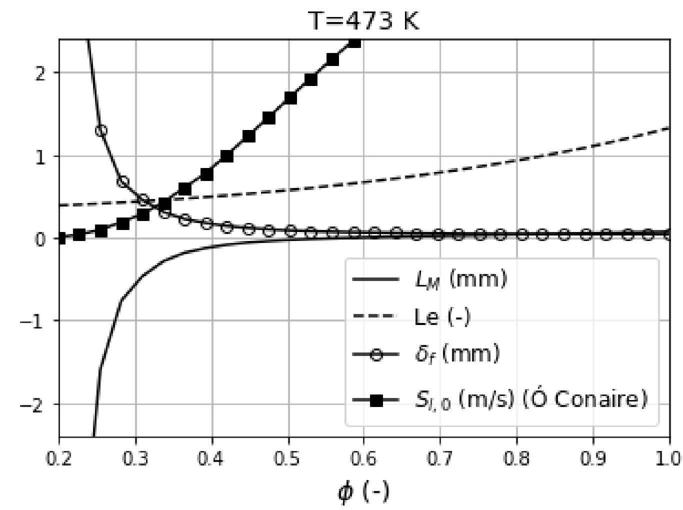


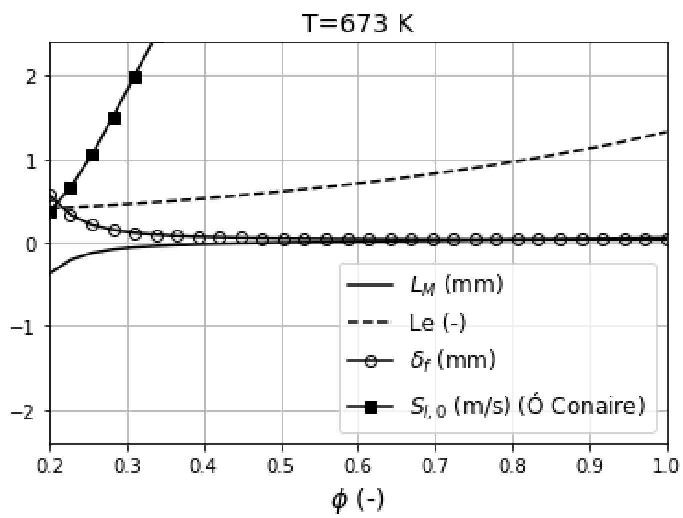
Figure 4.7: Flashback limits using Cantera for flame speed calculations



(a)



(b)



(c)

Figure 4.8: Markstein length L_m, Lewis number Le, flame thickness δ_f and unstretched laminar flame speed $S_{l,0}$ at different preheating temperatures.

The turbulent flame speed using the Damköhler closure is then:

$$S_t = S_{l,s}\left(1 + C\left(\frac{u'}{S_{l,s}}\right)^{0.5}\right) = S_{l,0}\left(1 + C\left(\frac{u'}{S_{l,0}}\right)^{0.5}\right)$$

The results of this modification are displayed in Figure 4.9 on the next page. The tuning constants are again unchanged, given in Table 4.1. The effects are most noticeble at equivalence ratios below $\phi = 0.6$. Compared to a non-zero Markstein length (cf. Fig. 4.3) the flashback limits at room temperature are now predicted accurately down to lower equivalence ratios. There is even slight underprediction at the very lowest equivalence ratios. Removing the calculated Markstein length and resulting stretched laminar flame speed in the model does not have a very noticeable effect on flashback limits at higher equivalence ratios and/or higher preheat temperatures indicating that for moderate Markstein lengths, the calculated flame stretch does not have a large effect on the laminar flame speed. Note that the Lewis number correction is still implemented so the turbulent flame speed is increased for low Lewis number mixtures due to the onset of a cellular flame structure, as discussed in section 3.3.4.

In conclusion, the calculated flame thickness reaches unphysical values at low equivalence ratios causing the Markstein length to also reach unphysical values. The lower the equivalence ratio, the more unphysical the Markstein length becomes. This affects the accuracy of the results and causes overpredicted flashback limits at low equivalence ratios, especially noticable for low preheat temperatures. Using Cantera to calculate the laminar flame speed for the room temperature mixtures does not improve the predictions. However, removing the flame stretch effect on the laminar flame speed by setting the Markstein length to zero will extend the validity of the model to lower equivalence ratios without changing the prediction accuracy much at higher equivalence ratios, indicating that the flame stretch effect captured by the Markstein length does not have a large effect on the laminar flame speed. However, the flashback limits at the very lowest equivalence ratios for room temperature mixtures are slightly underpredicted.

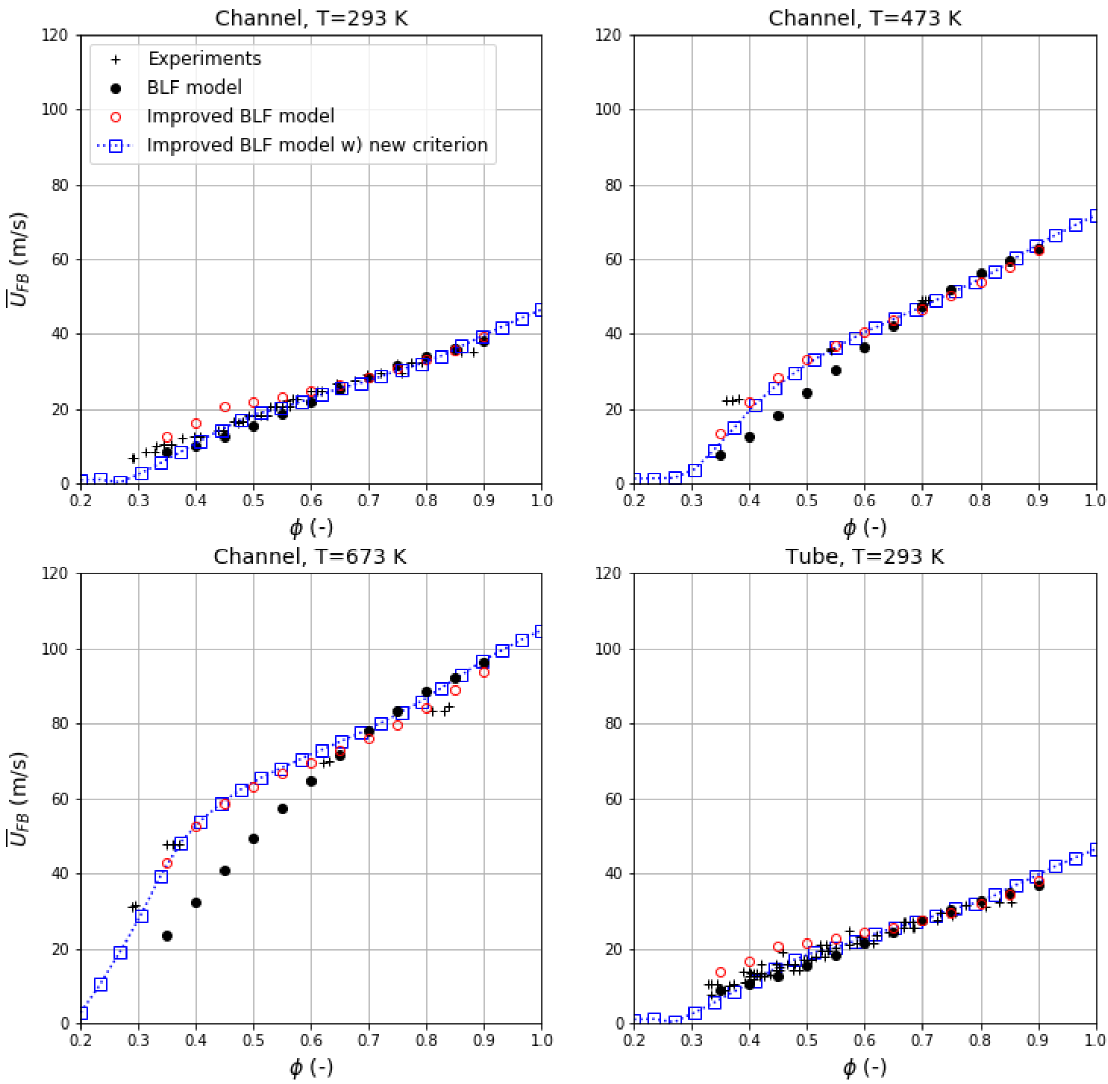


Figure 4.9: Flashback limits using the unstretched laminar flame speed in the Damköhler flame speed closure. Equivalent to setting the Markstein length to zero.

5

Prediction of flashback limits using flow simulation

The BLF model discussed in previous chapters requires information about the flow in the burner. Empirical expressions were used in all previous iterations of the model to obtain both the flow velocity and turbulence parameters in the boundary layer (cf. section 3.3.1). These expressions are only available in literature for a finite number of standard flows, like channel and tube flow. To be able to apply the flashback model in new burner designs, flow measurements could be carried out. Another option is to solve the governing equations of fluid flow numerically using computational fluid dynamics (CFD) software. The feasibility of coupling the BLF model to CFD will be investigated in this chapter. This will be referred to as the BLF+CFD model. In section 5.1 the model has been implemented in a Python 3.7 code coupled to ANSYS Fluent 19.0 and validated for the standard channel geometry. It is then applied to diffuser flows with underlying adverse pressure gradients in section 5.2.

5.1. Model duplication using CFD

The BLF model, with both Tober's improvements from section 3.3.4 and the generalized separation criterion introduced in section 4.2, has been implemented in code using CFD simulation results. First the CFD results are validated against experimental data in section 5.1.1. Then the implementation of the BLF+CFD model is discussed in section 5.1.2. Finally in section 5.1.3, the results are compared to experiments.

5.1.1. CFD simulation and validation

A two dimensional cut along the center of the channel was made to approximate the flow where it is not influenced by side walls. This is illustrated in Fig. 5.1.

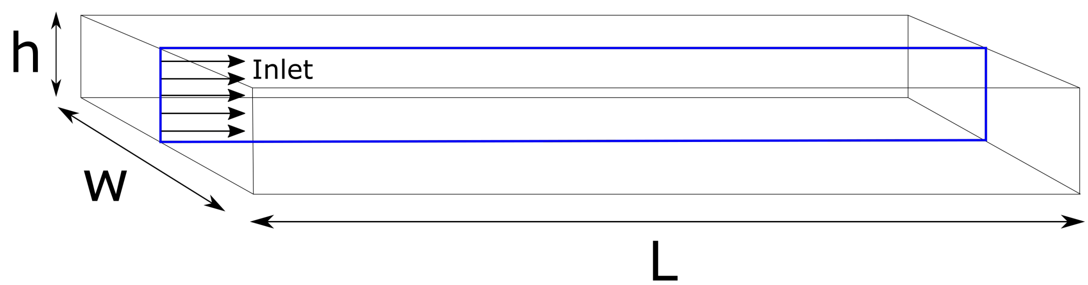
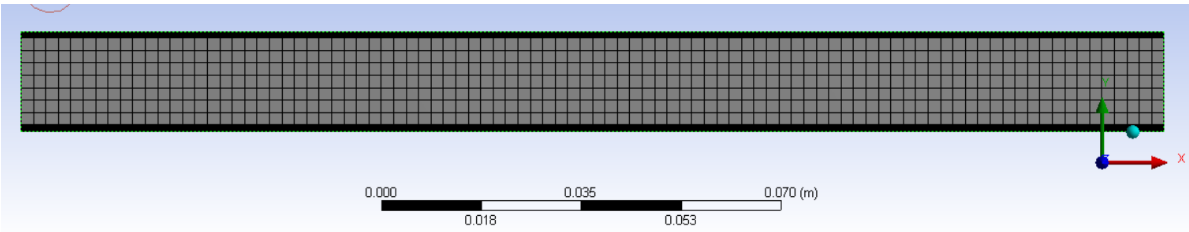
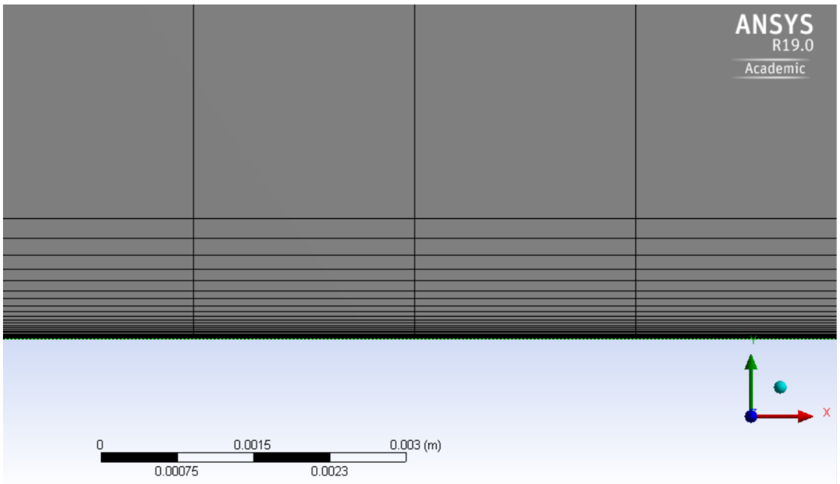


Figure 5.1: Illustration of the channel as it was modeled in ANSYS Fluent (blue). Not to scale.



(a) Channel geometry and mesh.



(b) Wall inflation layers.

Figure 5.2: Channel geometry and mesh and the wall inflation layer.

The 2-D channel geometry and mesh is displayed in Fig 5.2. The channel was made rather short (Length = 10 × Diameter) and the mesh very coarse to minimize the computation costs. The important near wall flow was however fully resolved using 30 inflation layers. This resulted in a total number of a mere 6300 nodes. The mesh and solver settings are given in Table 5.1.

Table 5.1: Fluent mesh and solver settings for the channel.

Nodes	6324
Boundary layer	Fully resolved using inflation layers
Solver type	Pressure based, steady state, 2-D
Solution method	SIMPLE
Viscous model	RSM
Fluid	Average H_2-air mixture (atmospheric air for validation)
Inlet	Uniform velocity, 5% turbulent intensity
Outlet	Pressure outlet

The Reynolds Stress Model (RSM) was used to close the Reynolds Averaged Navier-Stokes (RANS) equations. The RSM gives anisotropic turbulence which is necessary for Tober's modifications to the BLF model to include anisotropic flame stretch. A uniform velocity profile is applied at the inlet with a zero pressure outlet. A no-slip condition is applied at the walls. The hydraulic diameter is specified at both the inlet and the outlet and the turbulence intensity is set to 5%.

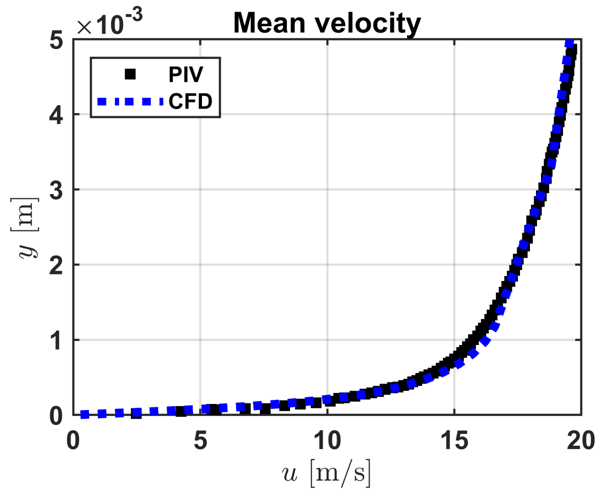


Figure 5.3: Channel mean velocity. CFD results compared to PIV experiments by Eichler [14]. Air mass flow $\dot{m} = 60$ g/s

Figure 5.3 shows mean velocity results in the boundary layer at the channel outlet. The CFD results are compared to PIV experiments from Eichler [14], carried out close to the flame stabilization zone. These simulations and experiments were done using pure air flow at room temperature and atmospheric pressure.

Figure 5.4 shows the dimensionless mean and fluctuating velocities.

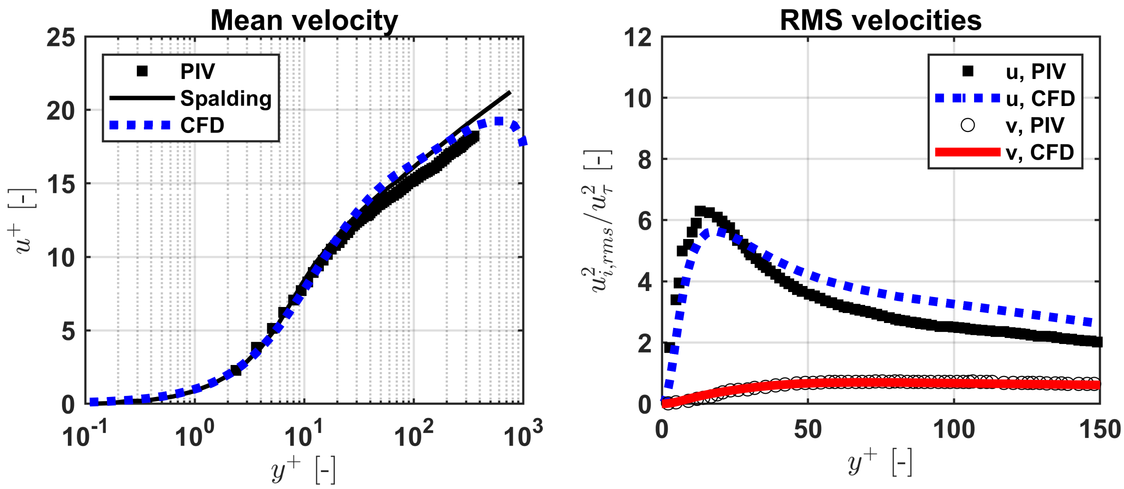


Figure 5.4: Turbulent boundary layer in the channel from CFD compared to PIV experiments by Eichler [14]. Air mass flow $\dot{m} = 60$ g/s

Here the velocities are normalized with the friction velocity u_τ and plotted against the dimensionless wall distance y^+. The mean velocity results agree well with the canonical Spalding profile [53] and the experimental data. The fluctuating streamwise velocity is underpredicted very slightly at the maxima near $y+ \approx 15$ and overpredicted further away from the wall.

5.1.2. Implementation in code

In chapter 4 the Python code from Tober [59], which solves the improved BLF model, was modified to implement the generalized Stratford criterion. A new code has been written which uses CFD results for flow information, shown in appendix A.2.3 titled 'BLF+CFD model: Channel'. The code calls a function (shown in appendix A.2.1) to interpolate the Fluent solution data from the mesh nodes to the desired profile.

In order to obtain a map of flashback limits, all investigated combinations of inlet bulk velocity and equivalence ratios are checked for large scale flow separation due to the flame backpressure effect. Separate CFD simulation runs are therefore required at each bulk velocity. The density and the viscosity of the fuel-air mixture in the premixer needs to be specified before the simulation is started. As a simplification, the density and viscosity of the hydrogen-air mixture is based on an average equivalence ratio of $\phi = 0.6$. This simplification is necessary to have explicit flow information from CFD results, since the equivalence ratio at flashback for the given inlet velocity is the predicted variable. The system of equations that make up the BLF model is now linear and easily solved. To see if this simplification is justifiable, the percentage variation in density and viscosity is plotted in Fig. 5.5 as a function of equivalence ratio.

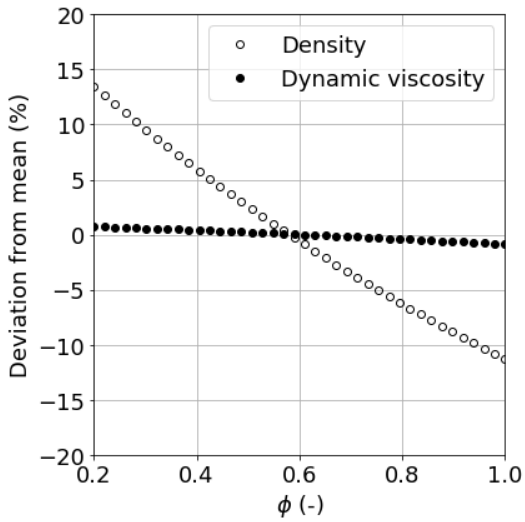


Figure 5.5: Density and viscosity variations of unburned hydrogen-air mixtures. Valid for all three preheating temperatures studied. Calculated using Cantera [21].

It's evident that the dynamic viscosity is almost constant. Meanwhile the density, and therefore the Reynolds number, varies up to 10% between $0.3 \leq \phi \leq 0.9$ where the experimental flashback results lie.

5.1.3. Predicted flashback limits

The predicted flashback limits are given in Fig. 5.6. The stretched laminar flame speed with a non-zero Markstein length was used in turbulent flame speed closure so flashback limits are not obtained below equivalence ratios around 0.35-0.45. This is due to the non-physical values of flame speed obtained at low equivalence ratios (see section 4.3.1). The results (blue) correspond well to the results found in section 4.3 (red) where empirical flow correlations were used. There is some limited deviation for preheating temperatures of 673 K. Also the tuning constant C in Damköhler's turbulent flame speed closure

$$S_t = S_{l,s}\left(1 + C\left(\frac{u'}{S_{l,s}}\right)^{0.5}\right)$$

has decreased slightly. Values for the fitting parameters C and n used in the turbulence flame speed closure and the outer mean velocity profile fit (Eq. (2.30) and Eq. (4.2) respectively) are given in Table 5.2.

For n = 6, C goes from 0.87 to 0.80. For n = 7 it goes from 1.05 to 0.95.

Table 5.2: Best fit values for C (Eq. (2.30)) for two different values of n.

	Channel	
	n	**C**
BLF+CFD model	6	0.80
BLF+CFD model	7	0.95

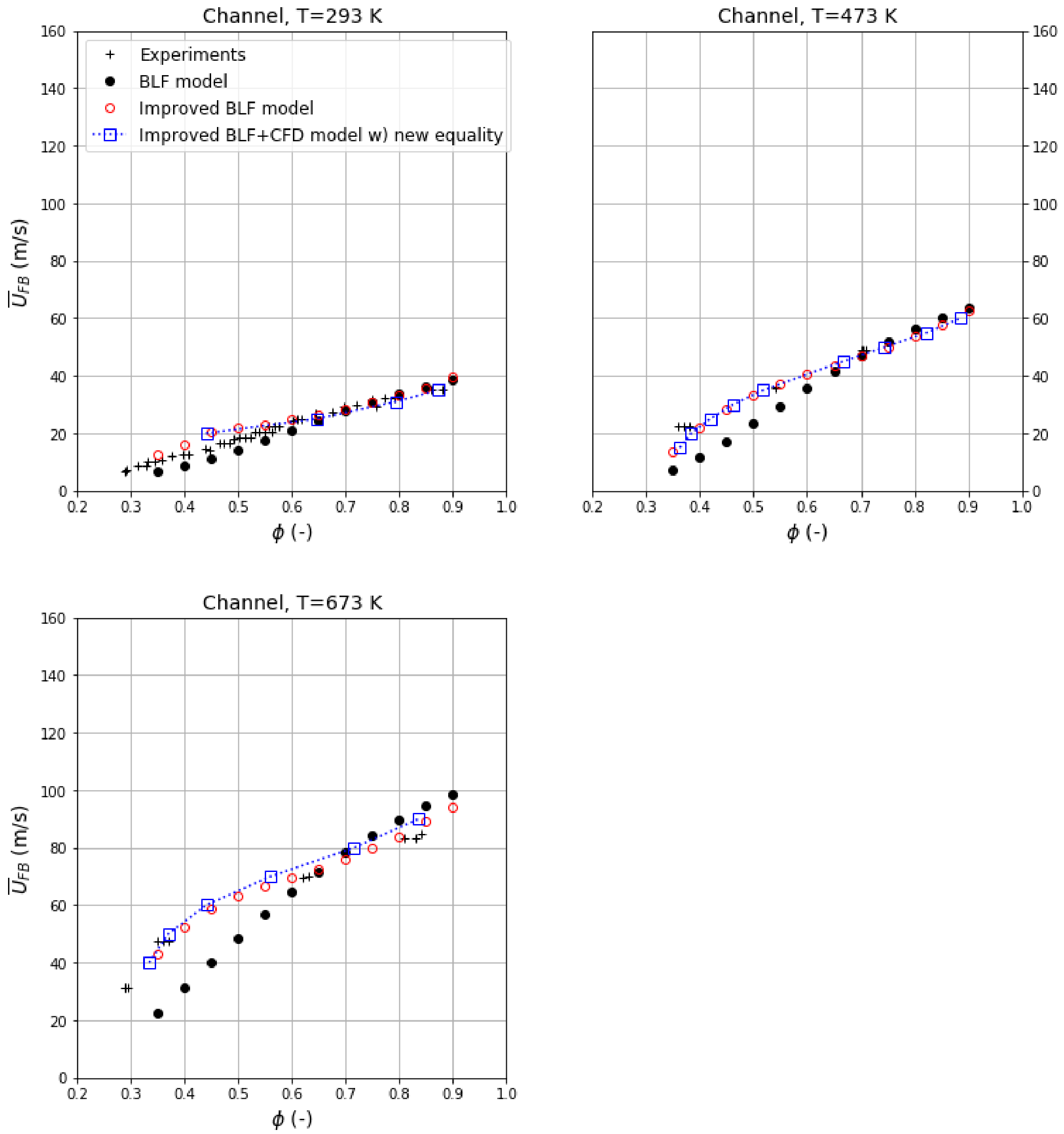


Figure 5.6: Results from the BLF model with the generalized criterion and using CFD channel flow results (blue). The stretched laminar flame speed is used in the Damköhler flame speed closure. Values for the fitting parameters used are given in Table 5.2.

Figure 5.7 shows the same results but using the unstretched laminar flame speed in the turbulent flame speed closure, similar to Fig. 4.9. The effect of using the unstretched laminar flame speed is effectively the same here as in the non-CFD coupled code, namely that the model can predicts flashback at all equivalence ratios with some underprediction at the low end at room temperature.

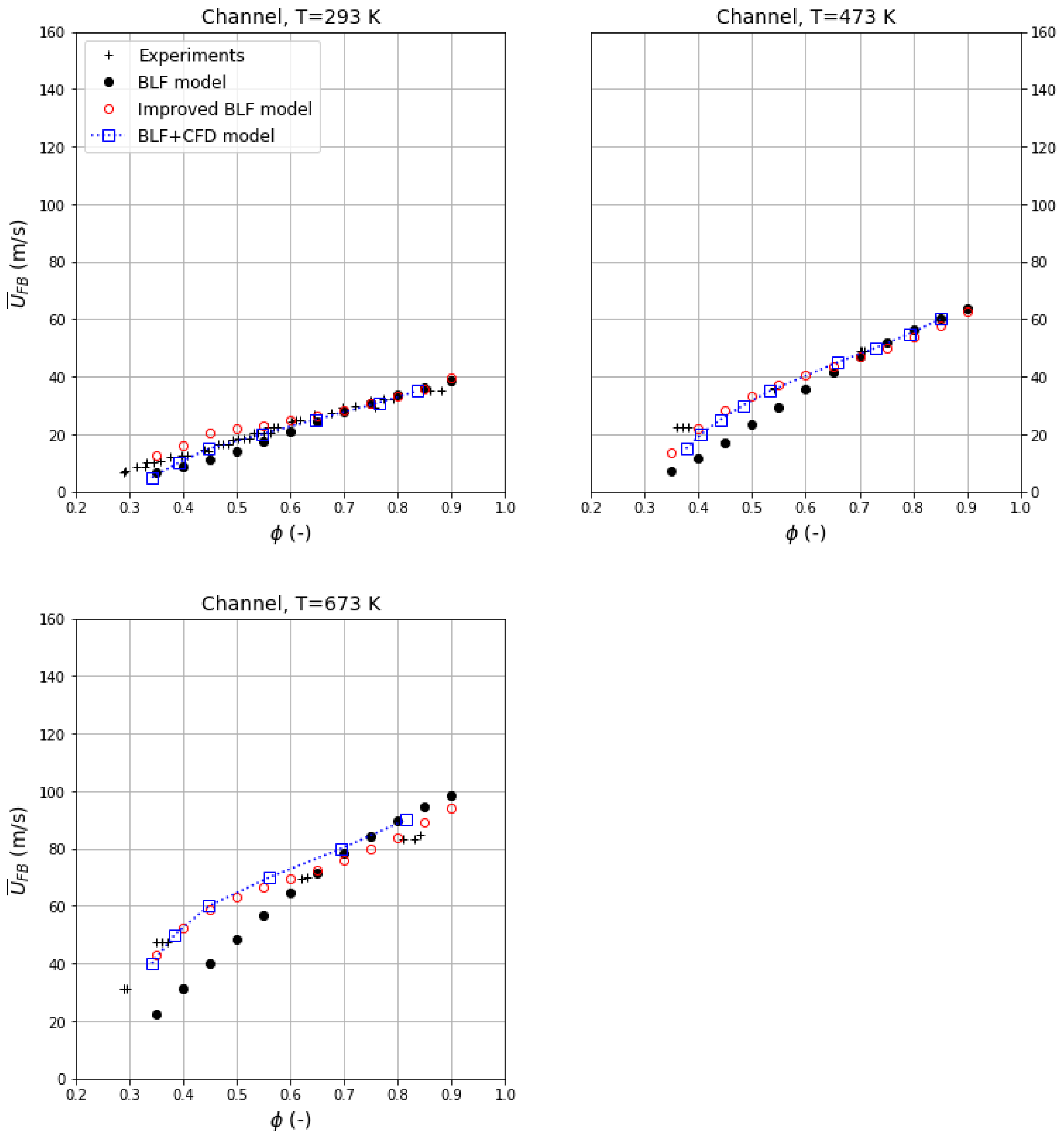


Figure 5.7: Results from the BLF model with the generalized separation criterion and using CFD channel flow results (blue). Here the unstretched laminar flame speed is used in the Damköhler flame speed closure. Values for the fitting parameters used are given in Table 5.2.

5.2. Application to adverse pressure gradient flow

As discussed in section 3.1, Eichler carried out confined turbulent BLF experiments in a channel, a tube and planar asymmetric diffusers with opening angles of 2° and 4°. A sketch of Eichler's measurement section with a diffuser ramp inserted is displayed here again in Fig. 5.8.

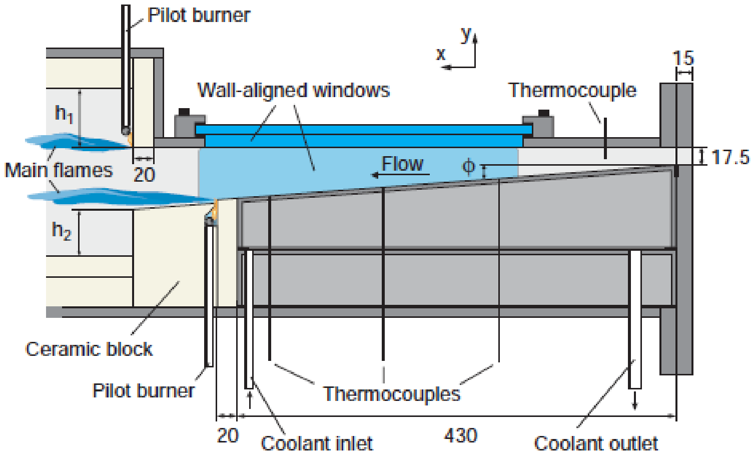


Figure 5.8: A sketch of the experimental measurement section. The lower wall section is interchangeable. 0°, 2° and 4° ramps were used during experiments to vary the pressure gradient. The dimensioning is done in millimeters. Source: Eichler's PhD thesis [14].

When low Mach number, incompressible fluid flows in a diffuser, it is retarded due to mass conservation given that it fills the expanding cross-sectional area. Subsequently the kinetic energy is converted to static pressure according to Bernoulli's principle. The resulting pressure increase is called the pressure recovery. Diffuser flow is therefore subject to an adverse pressure gradient which is a necessary prerequisite for boundary layer separation as discussed in 3.2. Flow separation in diffusers is usually an undesired effect since it hinders the effective expansion of the flow and decreases pressure recovery. Increasing the opening angle in the diffuser will increase the adverse pressure gradient and the tendency for the flow to separate.

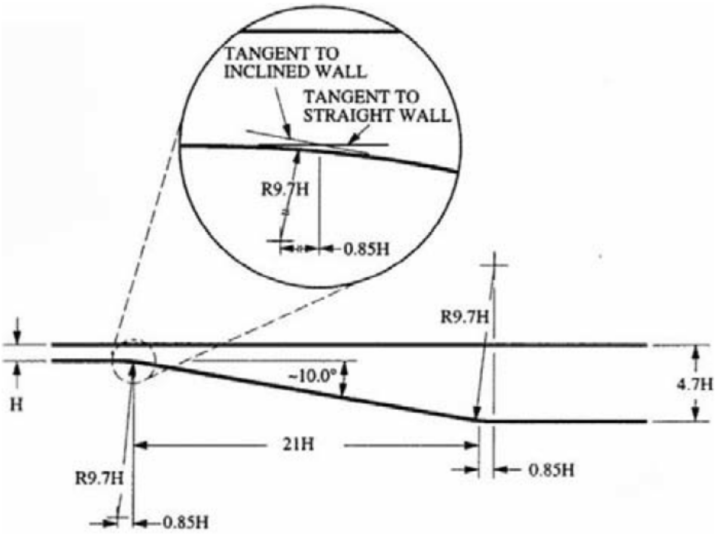
The next section deals with the choice of a suitable turbulence model for flow in diffusers. In section 5.2.2 the CFD results for the 2° and 4° diffusers, using the chosen turbulence model, are compared to experiments. Then the implementation of the BLF+CFD model for the diffuser cases is presented in section 5.2.3. A method to fit the outer boundary layer and thereby tailoring the separation criterion to the flow is presented in section 5.2.4. Finally, the predicted flashback limits are given in section 5.2.5.

5.2.1. Choice of turbulence model

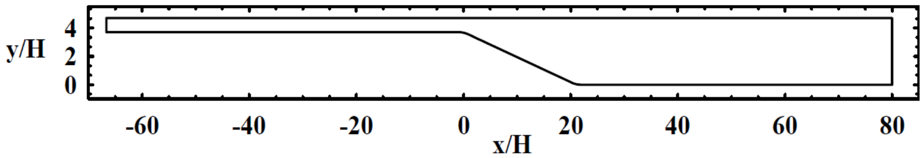
A turbulence model needs to be selected that is suitable for diffuser flow. The RSM model was chosen due to the anisotropy of the turbulence necessary for Tober's flame stretch modification. Other turbulence models also predict anisotropic turbulence, e.g. the V2F model. Isotropic turbulence models can also be considered if the unstretched laminar flame speed is used.

El-Behery and Hamed [16] compared the performance of six turbulence closures in a 10° planar asymmetric diffuser dimensioned in Fig. 5.9. The flow was assumed to be steady and incompressible. The following six turbulence models were compared in the study: Standard k-ϵ (SKE), Low Reynolds Number k-ϵ (LRNKE), Standard k-ω (SKW), Shear Stress Transport k-ω (SST), Reynolds Stress Model (RSM) and the v^2-f Turbulence Model (V2F). These models are all available in FLUENT 19.0 with the exception of the V2F model.[1] All models but the RSM are based on the Boussinesq approximation and use an eddy viscosity formulation to calculate the Reynolds stresses. The Reynolds Stress Model is a more elaborate model where a transport equation is solved for each Reynolds stress. The reader is referred to El-Behery's paper for more details on the individual models [16].

[1] It is however possible to implement the V2F model in Fluent manually using User Defined Scalars (UDS) and User Defined Functions (UDF).



(a) Schematics



(b) CFD domain with extended inlet and outlet

Figure 5.9: The 10° planar asymmetric diffuser. Source: El-Behery [16].

The results showed that the V2F model performed best followed by the SKW and SST models, when considering pressure recovery, mean velocity, turbulent kinetic energy and correct prediction of separation and reattachment. The RSM model did not correctly predict the flow separation and overpredicted the kinetic energy in the core flow while requiring the largest computational time. It did however predict the streamwise Reynolds Stress $\overline{u'u'}$ better than other models. The SKE and LRNKE performed poorly overall.

To verify El-Behery's results and to establish correct meshing and settings in Fluent, the 10° diffuser was modeled and meshed in ANSYS DesignModeler and Fluent 19.0. Figure 5.10 shows the detailed mesh.

The inlet channel is long to allow the mean velocity profile to develop. Table 5.3 lists the mesh and solver settings.

Table 5.3: Mesh and solver settings for the 10° diffuser for turbulence model comparison.

Nodes	245,676
Boundary layer	Fully resolved using edge sizing with bias
Solver type	Pressure based, steady state, 2-D
Solution method	SIMPLE
Viscous model	RSM, SKW, SST, SKE
Fluid	Atmospheric air
Inlet	Uniform velocity, 5% turbulent intensity
Outlet	Pressure outlet

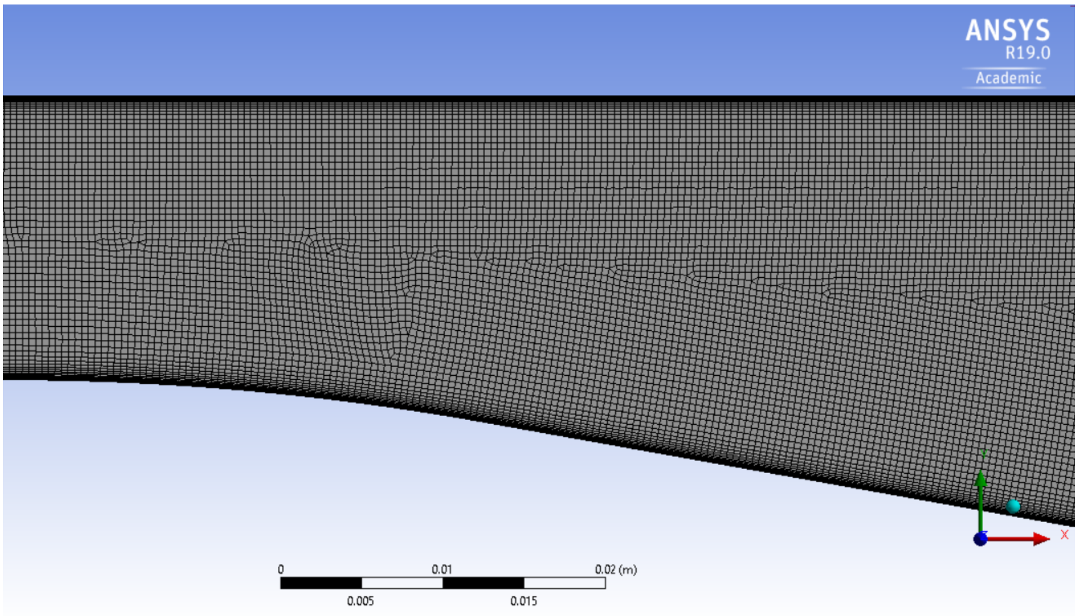
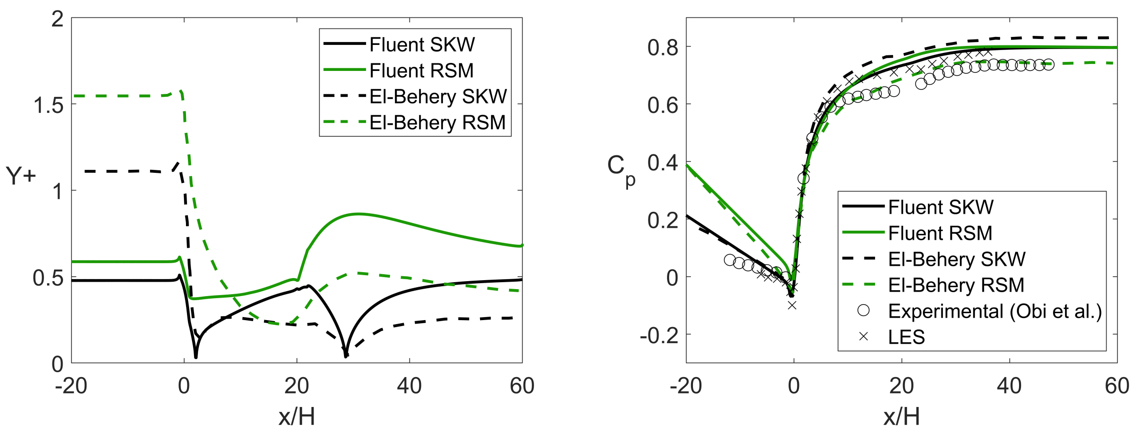


Figure 5.10: The 10° diffuser mesh.

Figure 5.11 shows the resulting dimensionless wall distance y^+ and the coefficient of pressure C_p at the lower wall of the diffuser using the SKW model and the RSM.



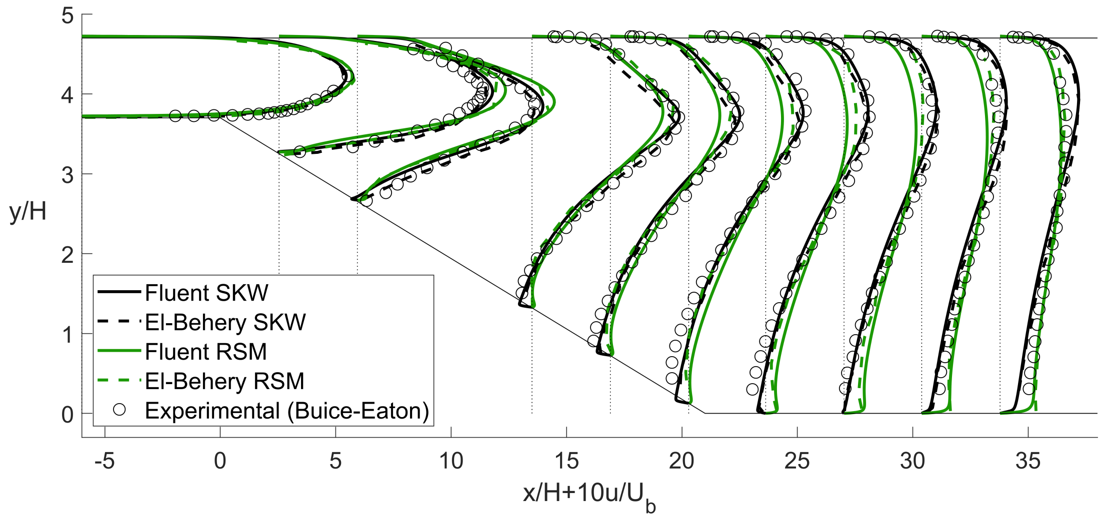
(a) Dimensionless wall distance y^+

(b) Coefficient of pressure C_p

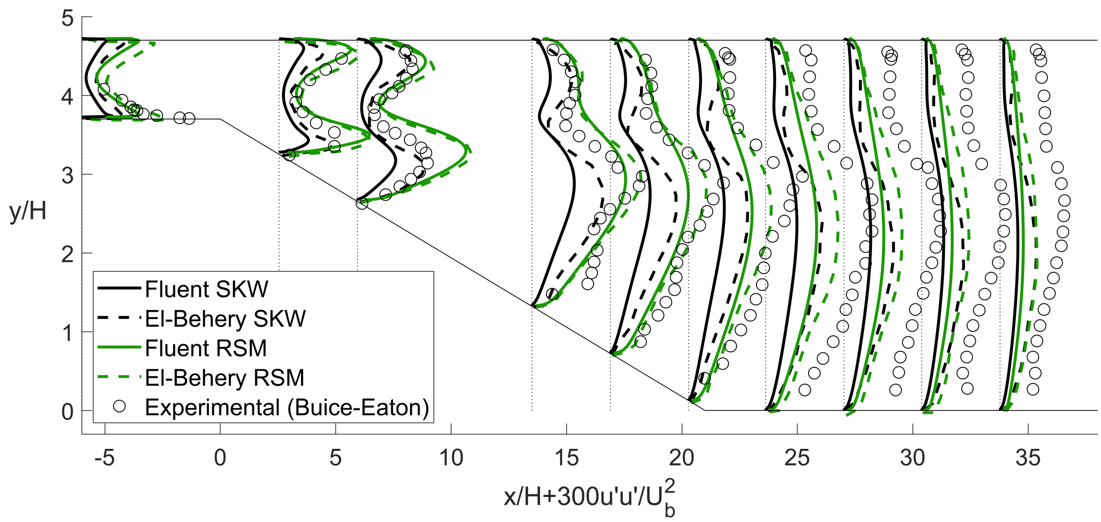
Figure 5.11: Fluent flow parameters at the lower diffuser wall compared to El-Behery's CFD results, experiments [41] and LES [31].

The results are compared to El-Behery's CFD results, experimental results from Obi et al [41] and LES results [31]. The y^+ is always below 1 which indicates that the boundary layer has been fully resolved. The coefficient of pressure results show good agreement with the data.

Figure 5.12 shows the mean velocity and streamwise Reynolds stress plotted at selected positions throughout the diffuser.



(a) Mean velocity

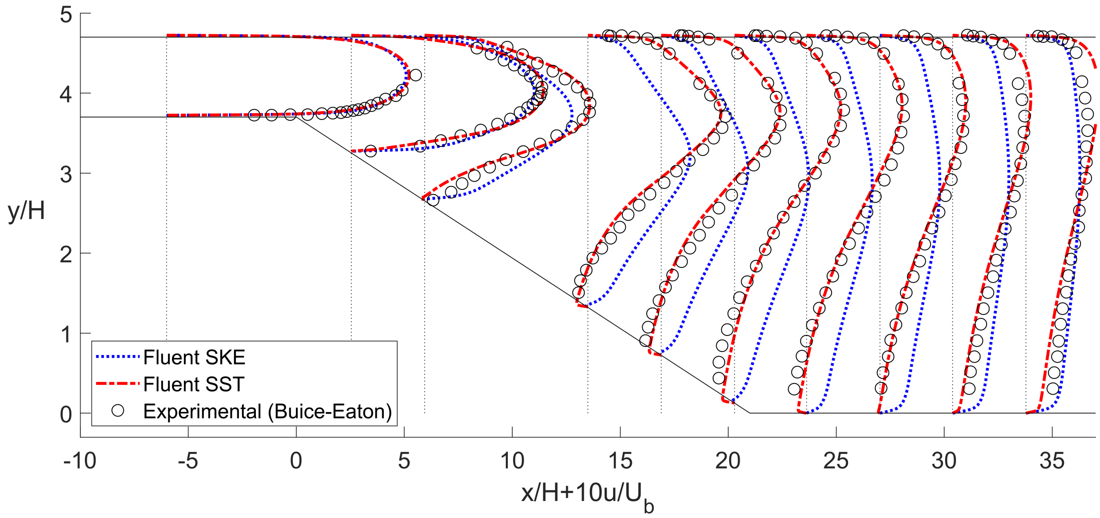


(b) Streamwise Reynolds Stress

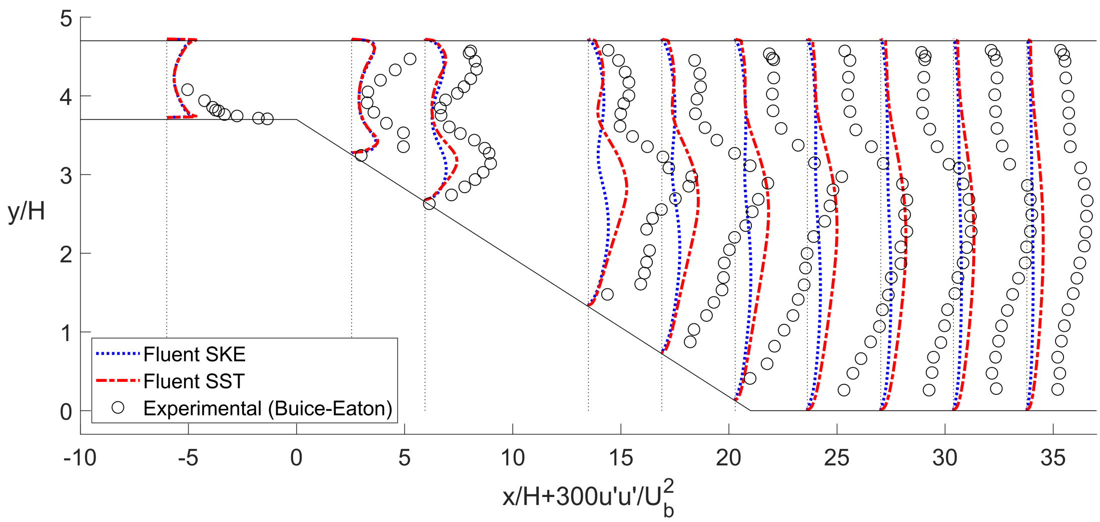
Figure 5.12: Flow results in the 10° diffuser compared to El-Behery's CFD results for the SKW and RSM turbulence models. Experimental results sourced from El-Behery's paper [16].

There is quite good agreement between the Fluent simulations and El-Behery's CFD results. The mean velocity profile is captured very well with the SKW model. As El-Behery noted, it delivers correct predictions for separation at the lower wall. The RSM results do not show separation but the mean velocity profile is still captured quite well. The RSM results predict the streamwise Reynolds stress better, especially around the midpoint of the diffuser. The velocity fluctuations are important to predict turbulent flame speeds.

Figure 5.13 shows a comparison of the SST and the SKE models against experimental data. The SST model performs similarly to the SKW model in Fig. 5.12 while the SKE performs poorly.



(a) Mean velocity



(b) Streamwise Reynolds Stress

Figure 5.13: Fluent results using the SST and SKE turbulence models. Experimental results sourced from El-Behery's paper [16].

The Reynolds Stress Model was selected for the following diffuser flashback predictions. It's chosen since it is available in Fluent, it predicts the $\overline{u'u'}$ Reynolds stress well which will be used to derive the velocity fluctuations u' for the turbulent flame speed closure, and captures the main flow mean velocity well. The lack of separation prediction is not of great concern since the opening angles of the diffusers will be limited to 4°. The larger computational costs associated are also not a big concern for steady, two dimensional, incompressible and isothermal flow.

5.2.2. CFD simulation and validation for diverging burners

The general dimensions of the 2° and 4° Eichler diffusers as they were modelled in Fluent is displayed in Fig. 5.14.

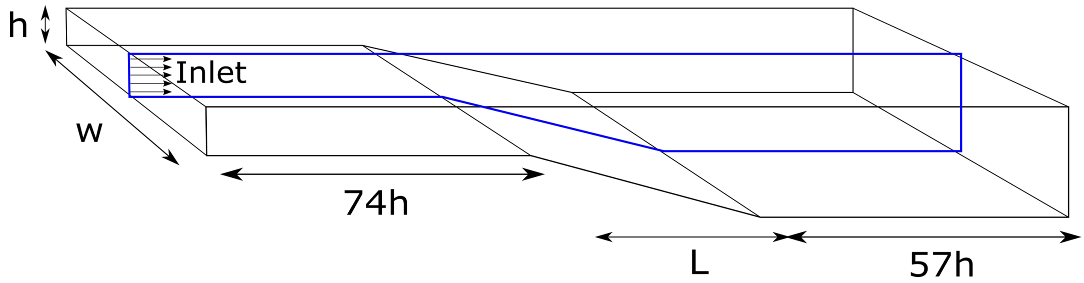


Figure 5.14: The diffusers as they were modeled in Fluent (blue). Not to scale.

A snapshot of the 4° diffuser section from the DesignModeler software is displayed in Fig. 5.15.

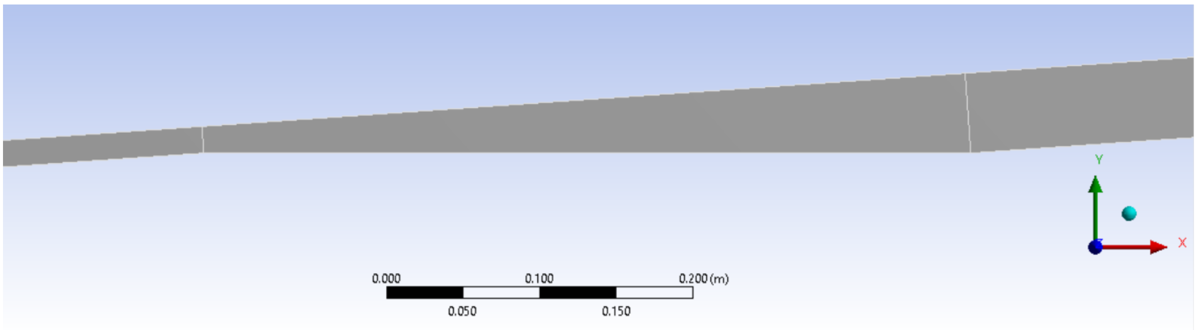


Figure 5.15: A snapshot of the 4° diffuser from DesignModeler.

The diffuser was meshed similarly to the 10° diffuser. Table 5.4 lists the mesh and solver settings.

Table 5.4: Mesh and solver settings for the Eichler diffusers.

Nodes	455,952
Boundary layer	Fully resolved using edge sizing with bias
Solver type	Pressure based, steady state, 2-D
Solution method	SIMPLE
Viscous model	RSM
Fluid	Average H2-air mixture (Atmospheric air for validation)
Inlet	Uniform velocity, 5% turbulent intensity
Outlet	Pressure outlet

Eichler [14] measured the mean and fluctuating velocities in the 2° diffuser at different positions E1, E2 and E3 (see Fig. 5.16) using PIV.

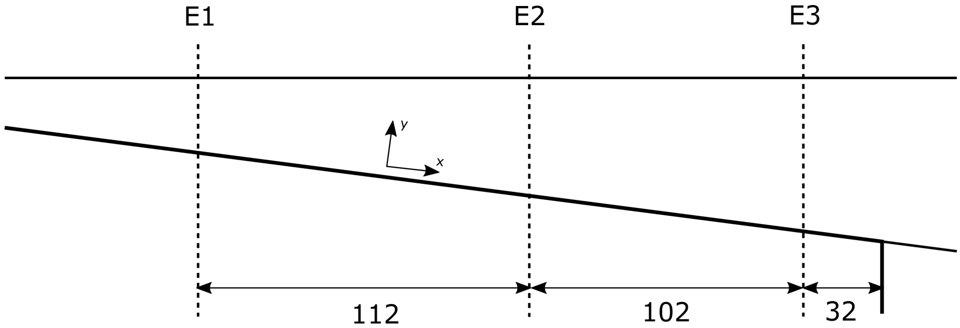


Figure 5.16: The PIV measurement positions and the local coordinate system in Eichler's diffuser experiments [14]. Dimensions are in millimeters.

Figure 5.17 shows CFD results compared to these experiments as well as the canonical profile for channel flow from Spalding [53].

It is important to capture correctly both the outer mean velocity profile and the maxima of the velocity fluctuations. The mean velocity profile will affect the generalized separation criterion as explained later, while the maximum u' is used in the turbulent flame speed closure. The mean velocities are captured quite well in the simulations, especially close to the wall. The deviation from experiments is greatest at the most upstream position E1 where it seems that the experimental profile has not developed fully. The CFD results show an increased velocity in the wake region compared to the Spalding channel profile, which is expected in adverse pressure gradient flow [39]. Eichler did not publish data for the wake region in the 2° diffuser.

The dimensionless fluctuation profile (Reynolds stresses) from PIV increases between the most upstream position E1 and E2 but keeps a similar shape and magnitude between E2 and E3. At E2 and E3 the streamwise fluctuations peak at c.a. 8.5 compared to 6 in the channel (cf. Fig. 5.4). The profiles from CFD are not changing appreciably between measurement positions. The peak streamwise fluctuation velocities are underpredicted in the CFD simulations compared to the PIV data at E2 and E3. At E1 the peaks are similar but further from the wall the streamwise fluctuations are overpredicted. The friction velocity $u_\tau = \sqrt{\tau_w/\rho}$ is decreasing in the diffuser due to the general decrease of mean velocity wall gradient with the streamwise coordinate. Therefore the nominal turbulence fluctuations are decreasing as well if the dimensionless profiles are constant throughout. The dimensional turbulence fluctuation profiles for the 2° diffuser are displayed in Fig 5.18.

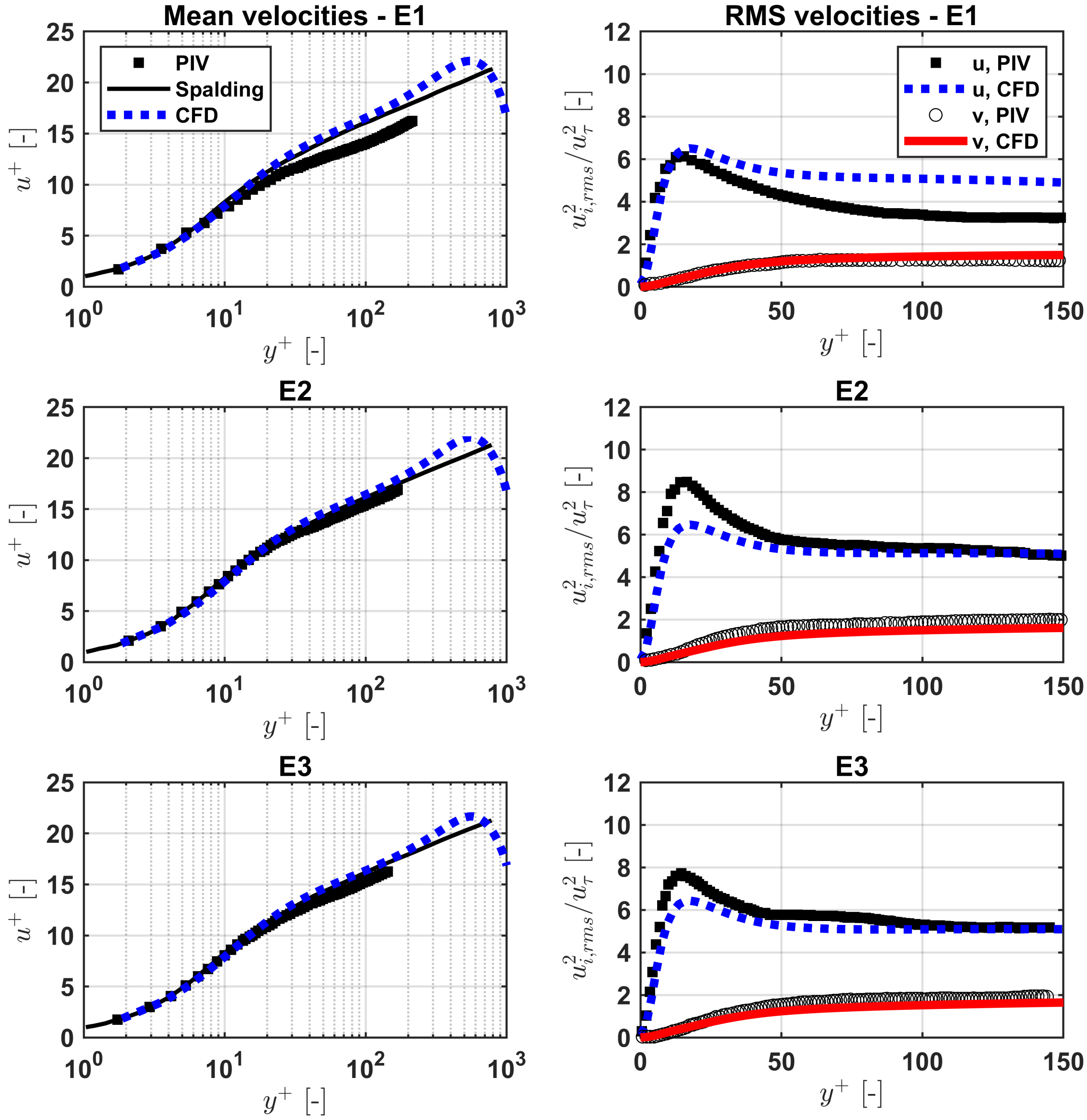


Figure 5.17: The turbulent boundary layer in the 2° diffuser. CFD results using the RSM turbulence model compared to PIV experiments from Eichler [14]. The mass flow at the inlet is $\dot{m}$ = 60 g/s

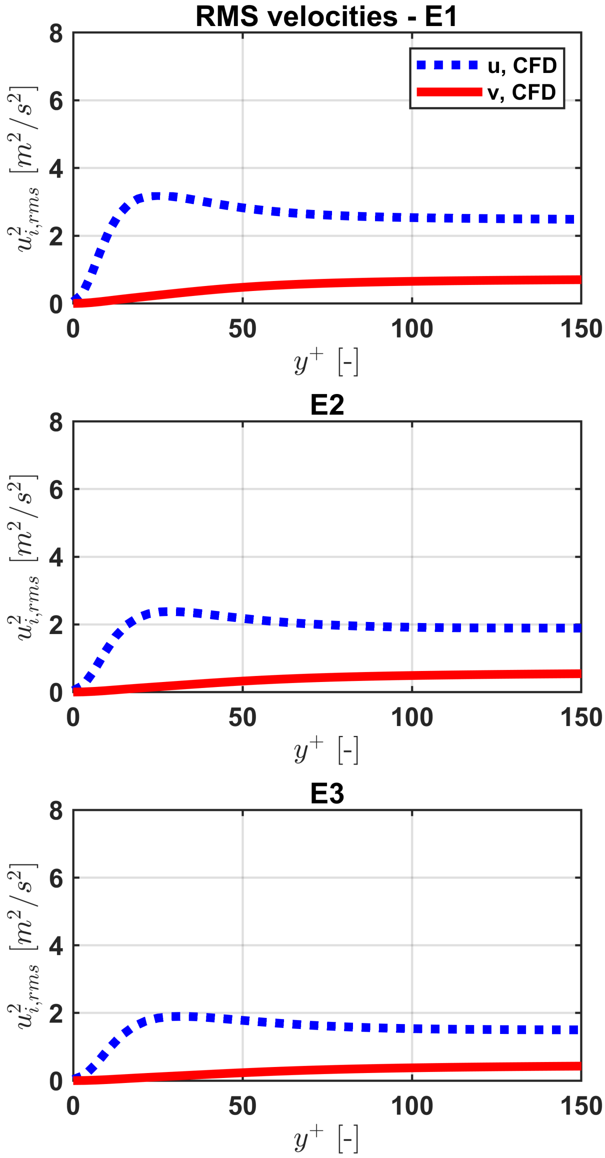


Figure 5.18: Dimensional turbulence fluctuations in the 2° diffuser at $\dot{m}$ = 60 g/s. Results from CFD using RSM for turbulence closure. No experimental data available.

Figure 5.19 shows the CFD results from the 4° diffuser. Here LDA measurements are available showing the structure of the mean velocity in the wake. The dimensionless mean velocity in the wake is underpredicted by CFD throughout the diffuser but the results are better in the boundary layer closer to the wall (below $y+ = 10^2$). The maxima of the streamwise fluctuations increases from 8.5 in the 2° diffuser to 10 in the 4° diffuser according to PIV. Again, CFD consistently underpredicts the fluctuation velocities. Figure 5.20 shows the dimensional RMS velocities decreasing throughout the diffuser. The magnitude of the dimensional RMS velocities are lower compared to the 2° diffuser.

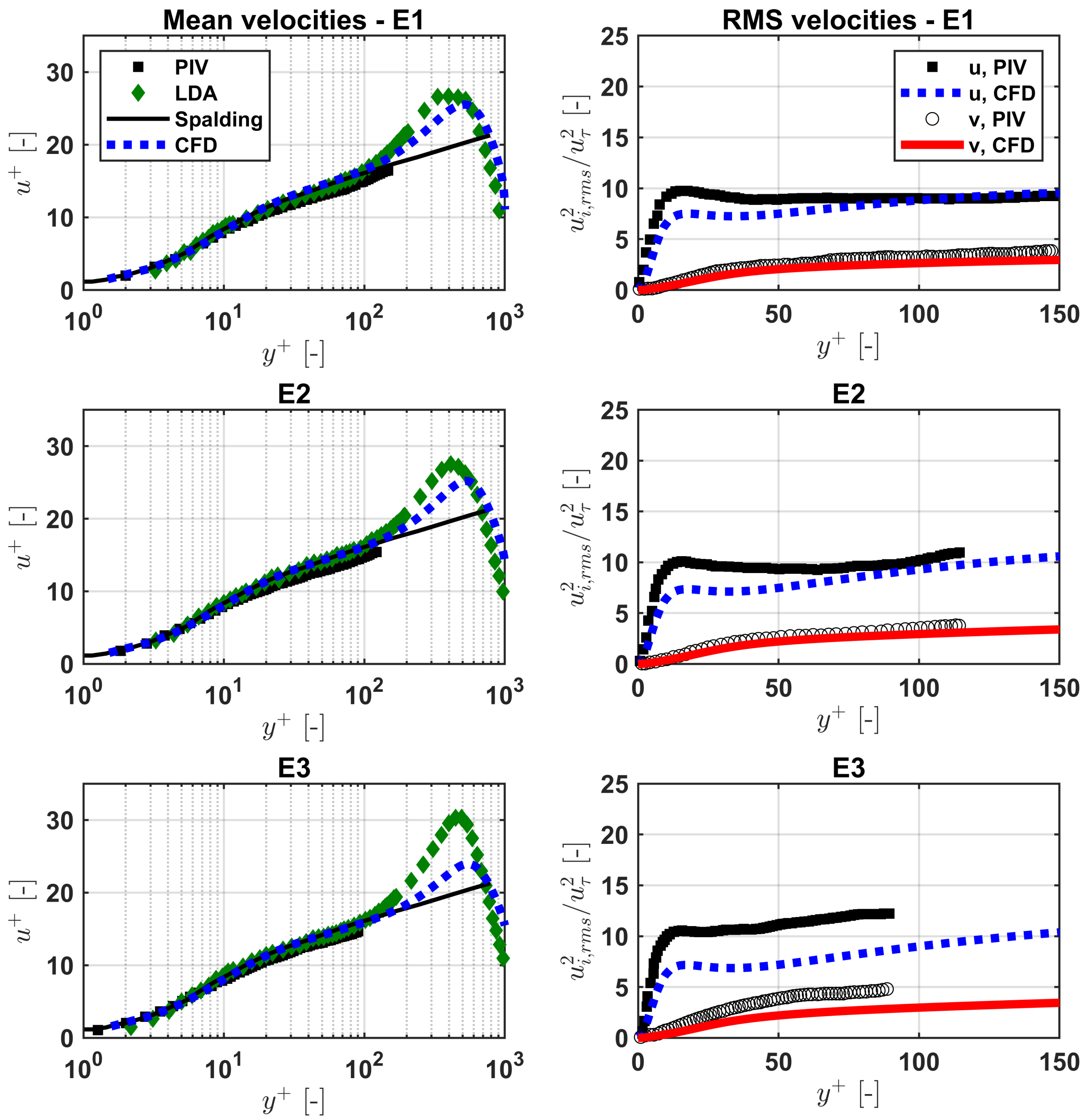


Figure 5.19: The turbulent boundary layer in the 4° diffuser. CFD results using the RSM turbulence model compared to PIV and LDA experiments from Eichler [14]. The mass flow at the inlet is $\dot{m} = 60$ g/s

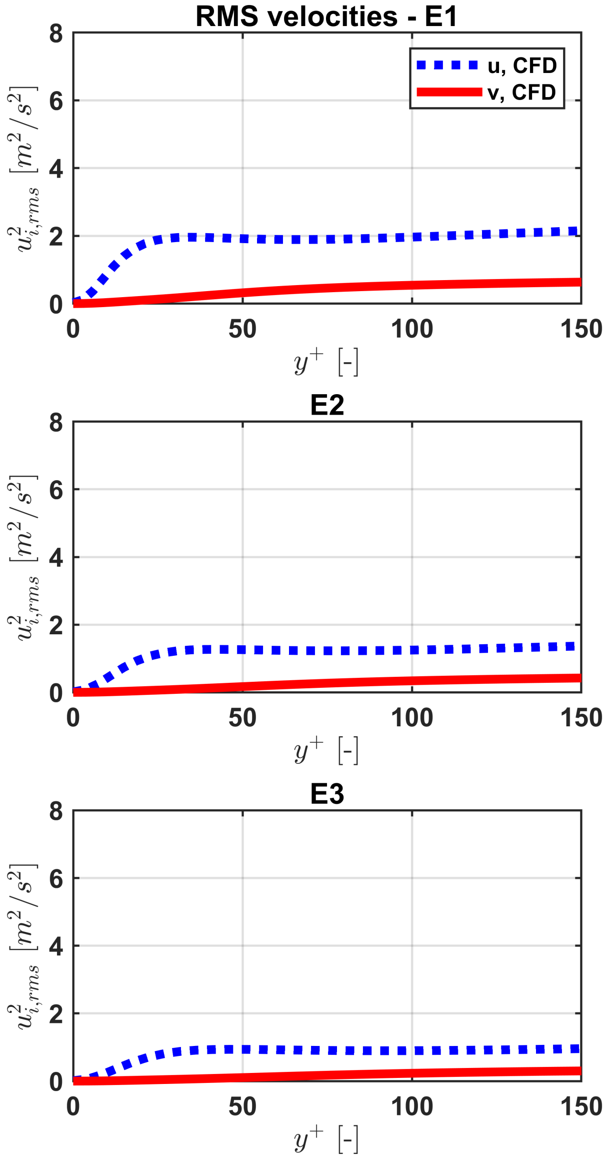


Figure 5.20: Dimensional turbulence fluctuations in the 4° diffuser at $\dot{m}$ = 60 g/s. Results from CFD using RSM for turbulence closure. No experimental data available.

5.2.3. Implementation in code

The BLF model is based on the assumption that flame flashback is initiated by flow separation in front of the stabilized flame. The flow expansion over the flame front causes a local adverse pressure gradient. Both the local adverse pressure gradient and the underlying adverse pressure gradient in the diffusers need to be taken into account since they should both contribute to flow separation.

The following additional assumption has been made in order to apply the BLF+CFD model in the 2° and 4° diffuser geometries:

- The BLF+CFD model with Stratford's separation criterion can be applied locally at any streamwise position in the diffuser, where a turbulent flame is assumed to be stabilized at the lower wall, in a similar way as it has been applied for the channel flow (see section 5.1). This is possible since the the local pressure gradient due to the backpressure effect is expected to be much larger than the underlying pressure gradient, the flow is not expected to separate without the flame being present, and the effect of the underlying adverse pressure gradient is mostly to retard the bulk fluid flow. This effect will be captured by fitting the outer (turbulent) boundary layer using the generalized Stratford criterion from section 4.2.

A Python 3.7 code was written to solve the BLF+CFD model for the diffuser cases. The code is given in section A.2.4 in the appendix. A block diagram is displayed in Figure 5.21. It illustrates how the onset of flashback is predicted given a certain inlet bulk velocity and equivalence ratio. Assumptions are highlighted in orange.

The pressure profile in front of the flame is still assumed to be one-dimensional:

$$p(x) - p(x = x_0) = \frac{\Delta p_{\text{flame}}}{x_f^2} x^2$$

with x_f = 10 mm, based on recommendations from the channel experiments of Eichler and Baumgartner [6, 14]. A term can be added to include the underlying pressure gradient in the cold flow, as recommended by Tober [59]:

$$p(x) - p(x = x_0) = \frac{\Delta p_{\text{flame}}}{x_f^2} x^2 + \left(\frac{\partial p_{\text{flow}}}{\partial x}\right) x \tag{5.1}$$

There are two main differences with this final version of the BLF+CFD code compared to the code written for the channel case in section 5.1:

1. A different method to process the Fluent results is used. This is explained in appendix A.1.5.
2. The outer, fully turbulent layer of the mean streamwise velocity profile at separation is now automatically fitted using the 1/n-th power law.

The fitting of the outer layer is discussed in the next subsection.

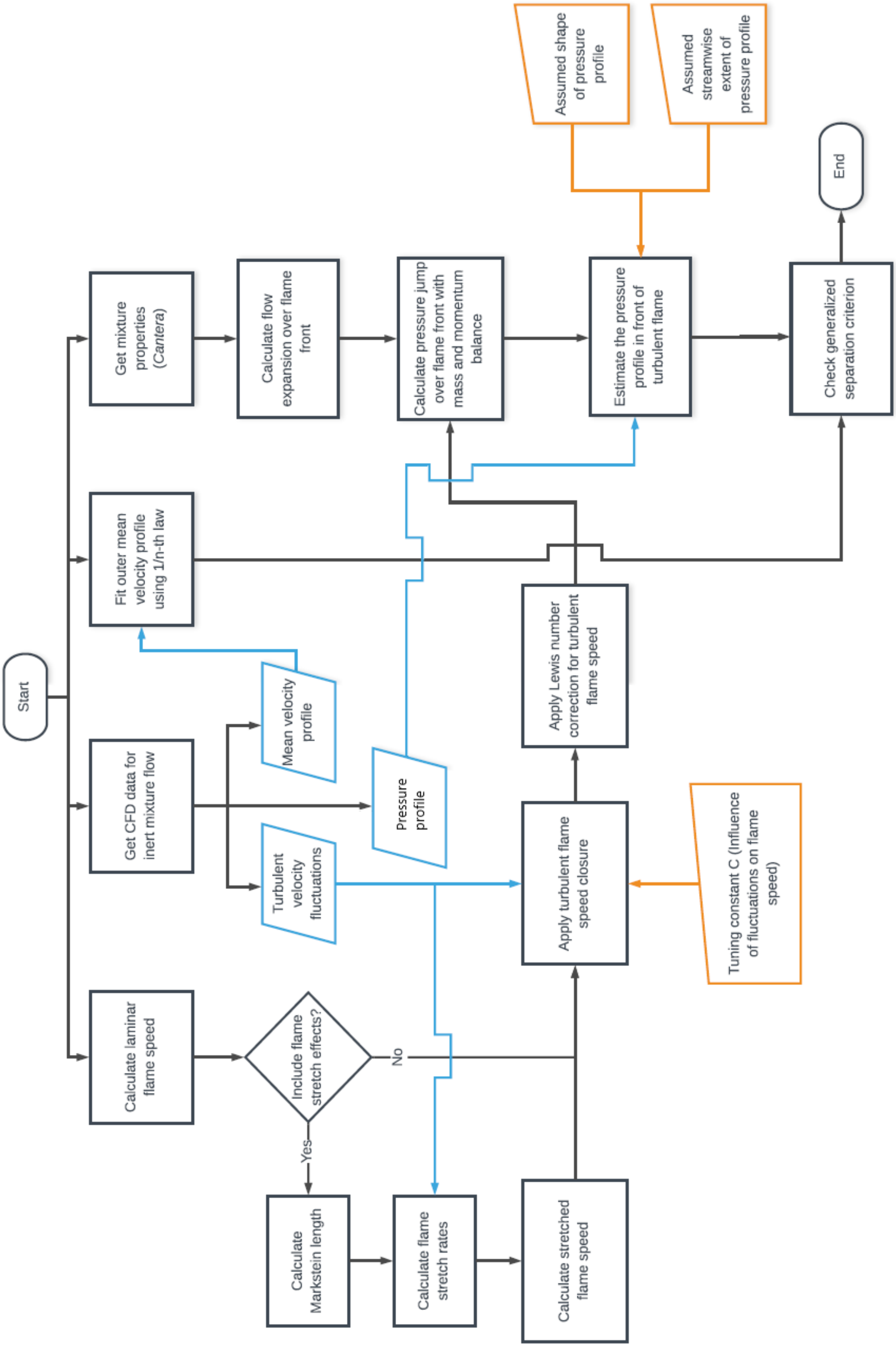


Figure 5.21: A block diagram of the BLF+CFD model representing the process for predicting flashback for a given mixture and flow conditions. Blue indicates information from CFD while orange indicates manual user inputs.

5.2.4. Fitting the outer layer

The mean velocity profile will deviate from a canonical channel or tube profile in the presence of an adverse pressure gradient. Stratford used the 1/n-th power law to fit the outer, fully turbulent layer of the profile (Eq. (4.2)):

$$\frac{u}{U_0} = \left(\frac{y}{\delta}\right)^{\frac{1}{n}}$$

This equation has been used to fit the mean velocity profile in the diffuser. The fitting parameters are the boundary layer thickness δ, the corresponding outer velocity U_0, and the power law constant n. The result is a tailor-made separation criterion (Eq. (4.8)) which models the diffuser velocity profile at separation:

$$C_p^{\frac{1}{4}(n-2)}\left(\delta\frac{dC_p}{dx}\right)^{\frac{1}{2}} = \left(\frac{3(0.41\beta)^4}{(n+1)n^2}\right)^{\frac{1}{4}}\left(1-\frac{3}{n+1}\right)^{\frac{1}{4}(n-2)}$$

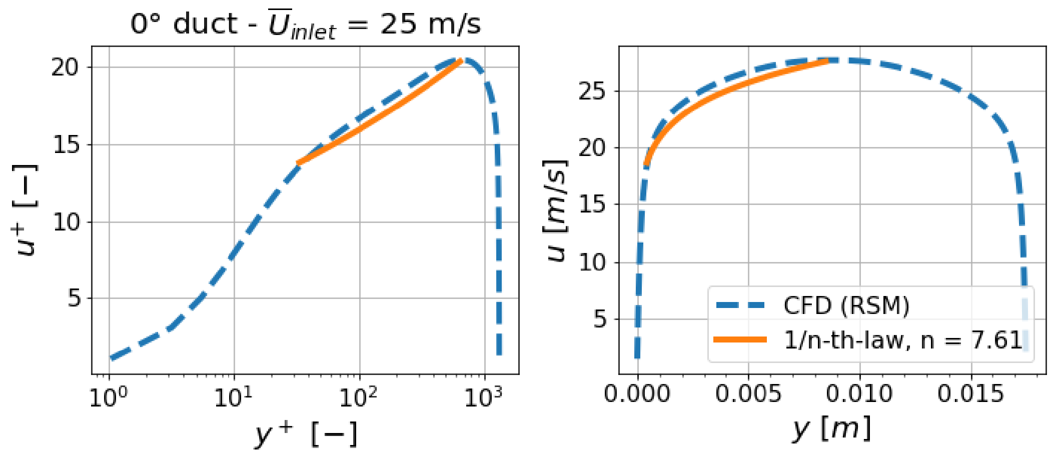
C_p is the coefficient of pressure:

$$C_p = \frac{p-p_0}{\frac{1}{2}\rho U_0^2}$$

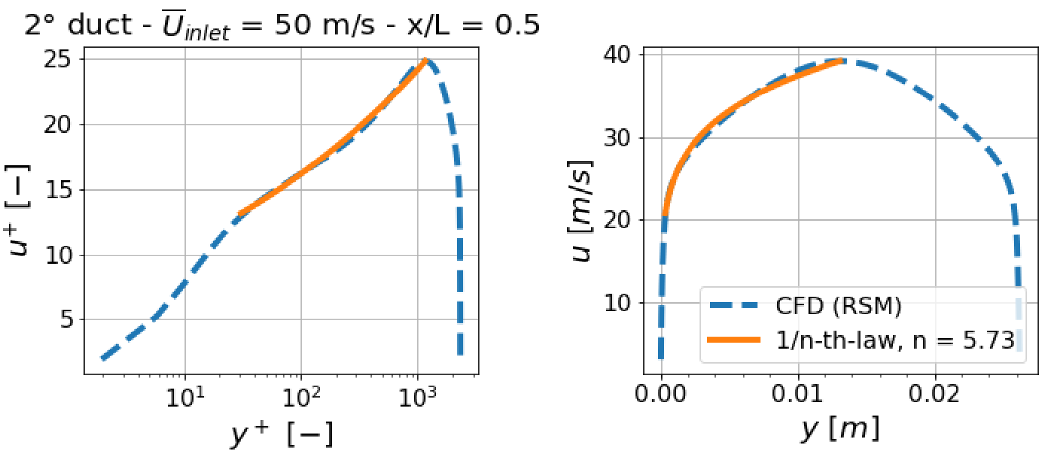
The boundary layer thickness δ and U_0 should be a matching pair, i.e. the value of U_0 should be taken at the y-coordinate y = δ. The boundary layer thickness can take the value of the duct halfwidth $\delta = h/2$ (where the maximum streamwise velocity is expected) or a value nearer to the wall, as discussed below. A least squares method is used to determine the best value of n. The method is applied such that the fitted profile fits as close as possible at $30 < y^+ < 50$. The logarithmic region influenced by turbulence extends down to $y^+ = 30$ [47]. The main idea is to fit the outer layer such that it is as accurate as possible near the transition region.

Figure 5.22 shows examples of fits in ducts with opening angles of 0°, 2° and 4° respectively. The profiles from CFD are captured adequately, especially the 2° profile. In the 4° diffuser the profile has a markedly different shape characterized by a higher velocity in the wake region.

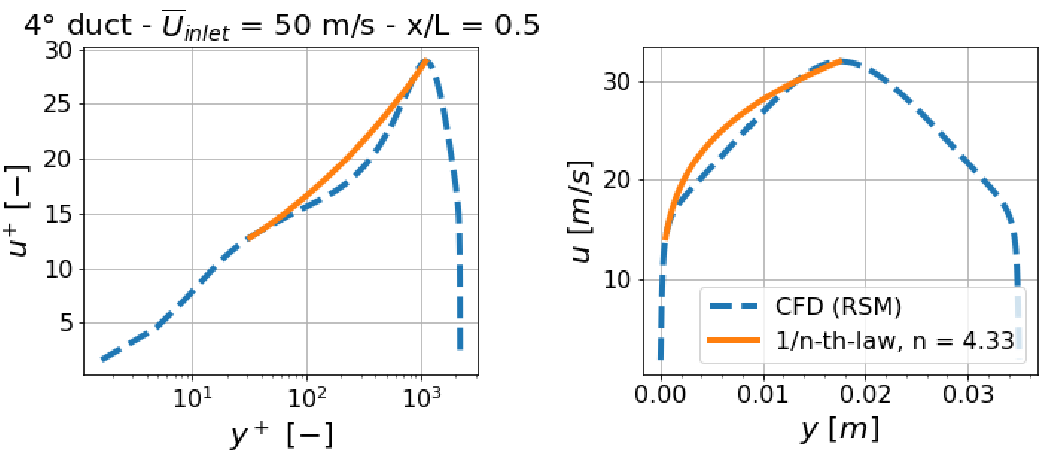
Somewhat better fits are obtained, especially for the 4° diffuser, if the fitted profiles are anchored nearer to the viscous sublayer. This is illustrated in Figure 5.23, where the boundary layer thickness δ in the fit has been reduced from $h/2$ to $h/6$. The high velocity wake region is avoided allowing the fit to better represent the shape of the outer layer closer to the inner layer. This region is the important part to fit since the Stratford criterion is derived by joining the inner layer and the outer layer. As discussed in section 4.1, the inner layer is derived from mixing length theory under the assumption of zero wall stress at separation and depends on the magnitude of the adverse pressure gradient.



(a) 0° channel, 25 m/s inlet velocity.

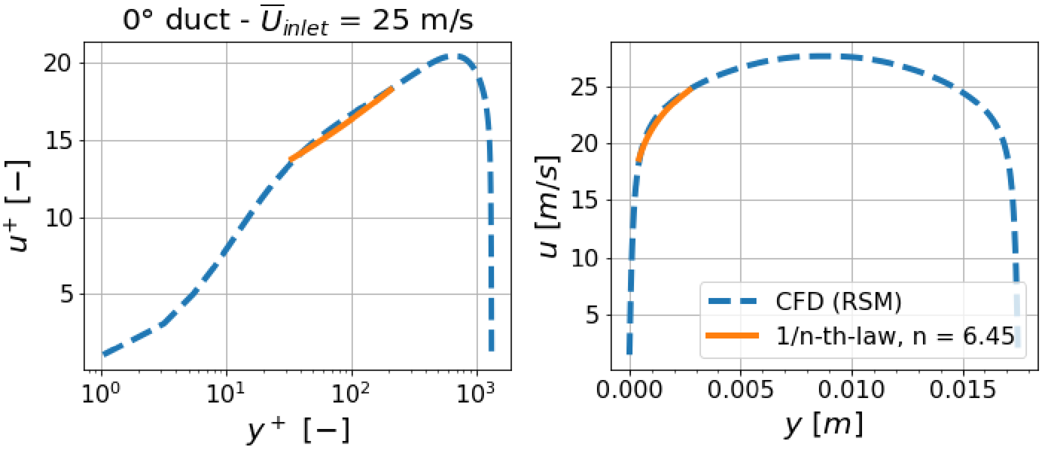


(b) 2° channel, 50 m/s inlet velocity. The CFD profile was exported from the midpoint of the diffuser.

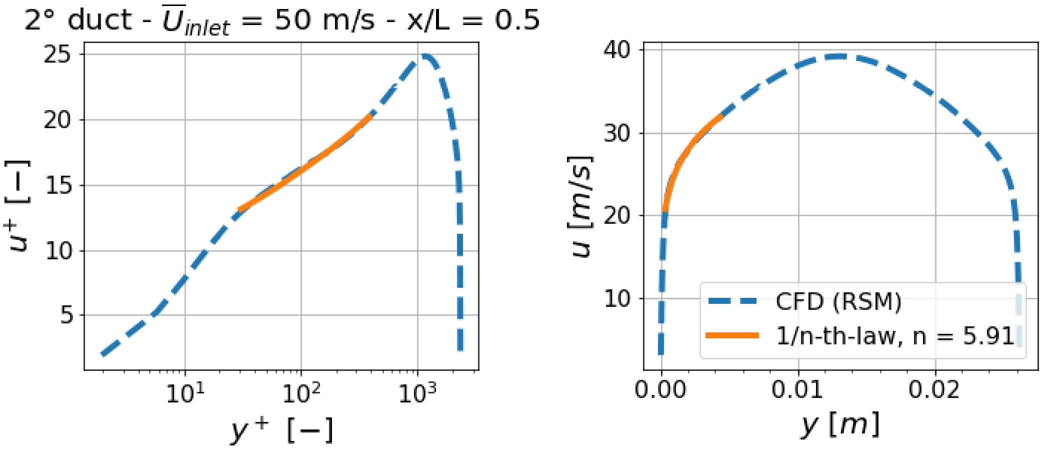


(c) 4° channel, 50 m/s inlet velocity. The CFD profile was exported from the midpoint of the diffuser.

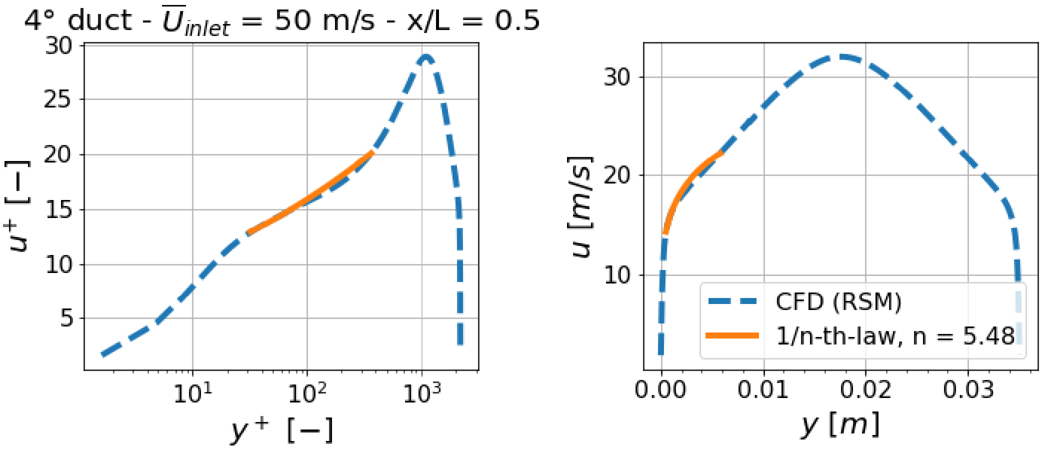
Figure 5.22: Examples of results from the fitting of the outer layer of the mean streamwise velocity profile in the BLF+CFD code. The fitted profile is anchored at the midpoint of the channel at the maxima of the velocity.



(a) 0° channel, 25 m/s inlet velocity.



(b) 2° channel, 50 m/s inlet velocity. The CFD profile was exported from the midpoint of the diffuser.

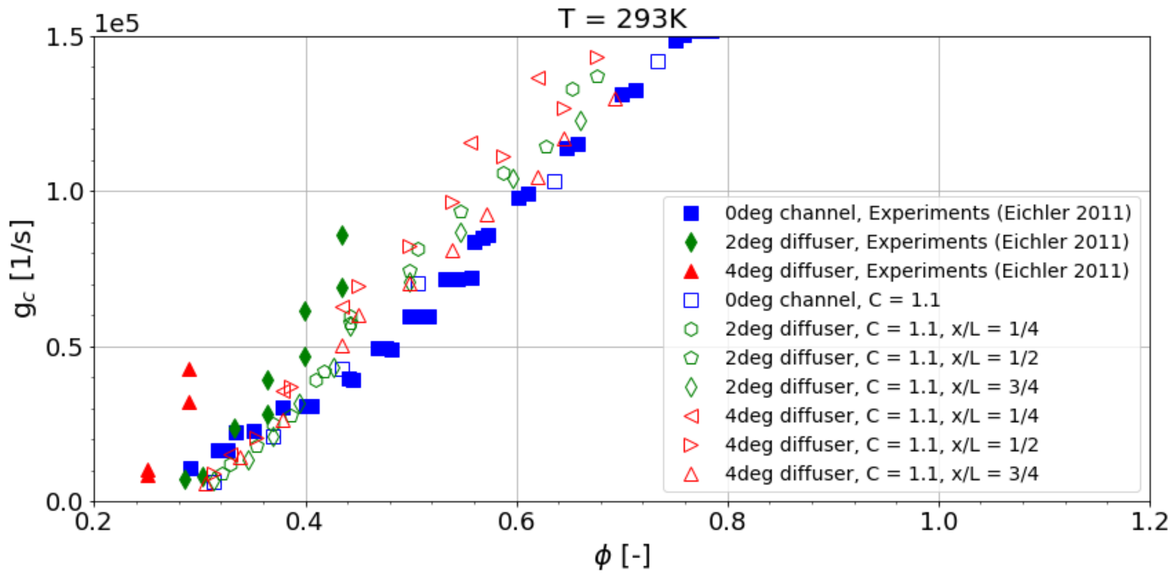


(c) 4° channel, 50 m/s inlet velocity. The CFD profile was exported from the midpoint of the diffuser.

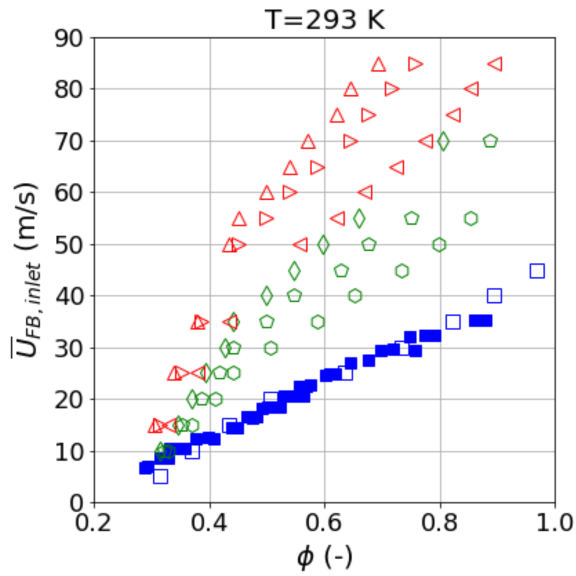
Figure 5.23: Results from the fitting of the outer layer of the mean streamwise velocity profile in the BLF+CFD code. Here the fit is anchored at 1/6th of the height of the channel instead of at the midpoint as in Figure 5.22.

5.2.5. Predicted flashback limits

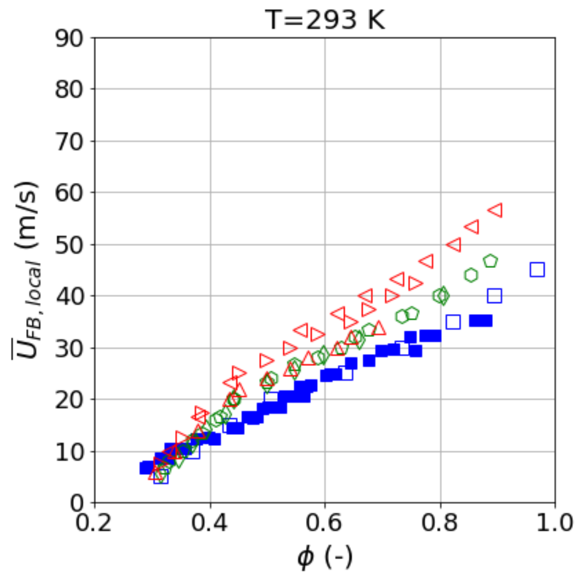
The BLF+CFD model was applied to the 2° and 4° diffuser geometries studied by Eichler (see section 3.1.1 and Fig. 3.5). Eichler measured turbulent wall flashback limits for hydrogen-air mixtures at ambient pressure and room temperature. His results were limited to very lean mixtures due to a limitation in maximum mass flow rates of air and fuel. The results from the model are compared to his experiments in Fig. 5.24.



(a) Critical gradient results.



(b) Inlet bulk velocity at flashback.



(c) Local bulk velocity at flashback.

Figure 5.24: Flashback limits from the BLF+CFD model for the 0° channel and 2° and 4° diffusers. The inlet and local bulk velocities at flashback was not given by Eichler for the diffuser cases.

The following axial positions in the diffusers were investigated:

$$\frac{x}{L} = \frac{1}{4}, \frac{1}{2} \text{ and } \frac{3}{4}$$

where L is the length of the diffuser sections. The C in the Damköhler turbulent flame speed closure (Eq. (2.30)):

$$S_t = S_{l,0}\left(1 + C\left(\frac{u'}{S_{l,0}}\right)^{0.5}\right)$$

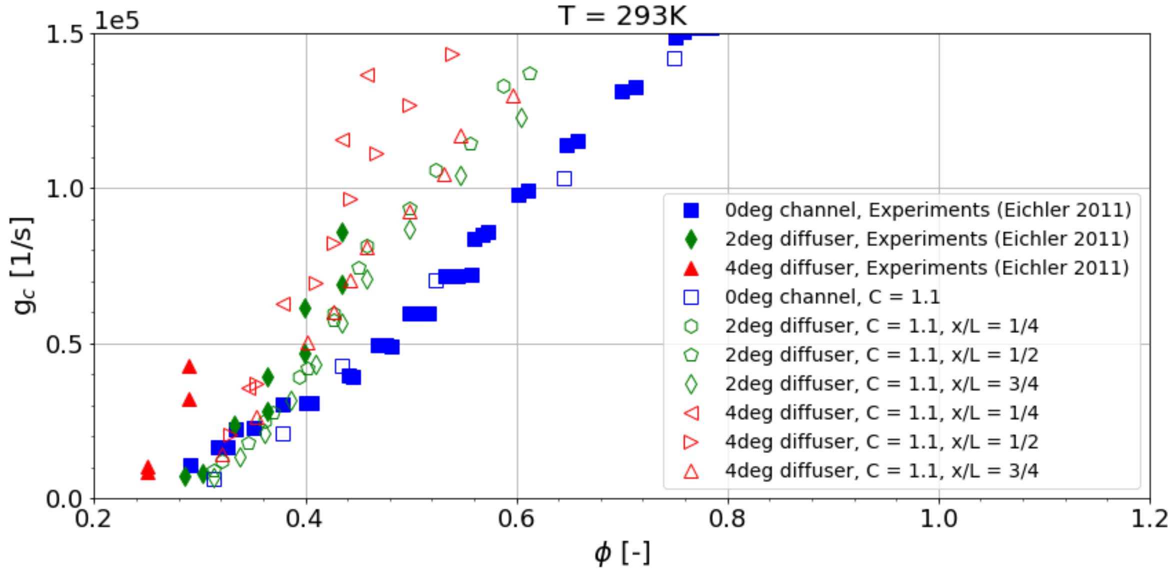
is $C = 1.1$, based on the best fit for the channel. Note that the unstretched laminar flame speed is used since using the stretched laminar flame speed resulted in unphysical flame speeds at low equivalence ratios as shown in section 4.3.1, while the important effects of the Lewis number on the turbulent flame speed are still captured using the Lewis number correction implemented by Tober with Eq. (3.36).

Figure 5.24a shows the critical gradient g_c (i.e. du/dy at the wall at flashback) compared to experiments. The flashback limits for the diffusers lie slightly above the channel results, due to a slightly different shape of the mean velocity profile. However they do not follow the experimental trends.
Figure 5.24b shows the inlet bulk velocity at flashback compared to the channel experiments. The inlet bulk velocities at flashback are higher in the diffusers due to the retardation of the flow.
The local bulk velocity is shown in Fig. 5.24c, calculated using simple mass conservation:

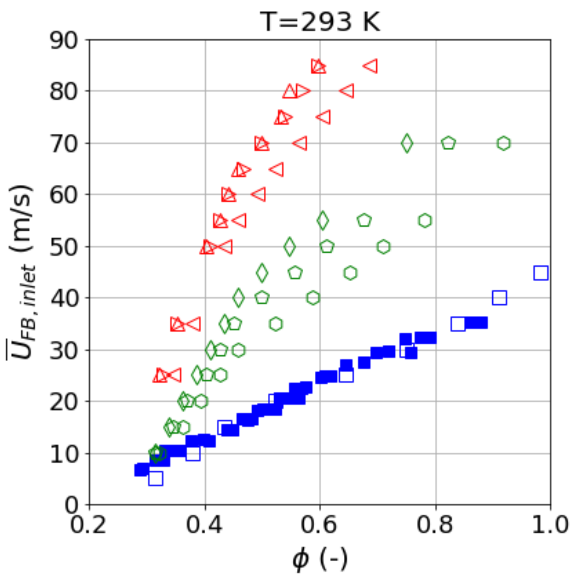
$$\overline{U}_{\text{local}} = \overline{U}_{\text{inlet}} \frac{A_{\text{inlet}}}{A_{\text{local}}} \tag{5.2}$$

The local bulk velocities at flashback are higher in the diffuser. This increase is caused by the higher critical gradients at flashback.
The inlet and local bulk velocities at flashback was not given by Eichler for the diffuser cases and could not be derived from his results.

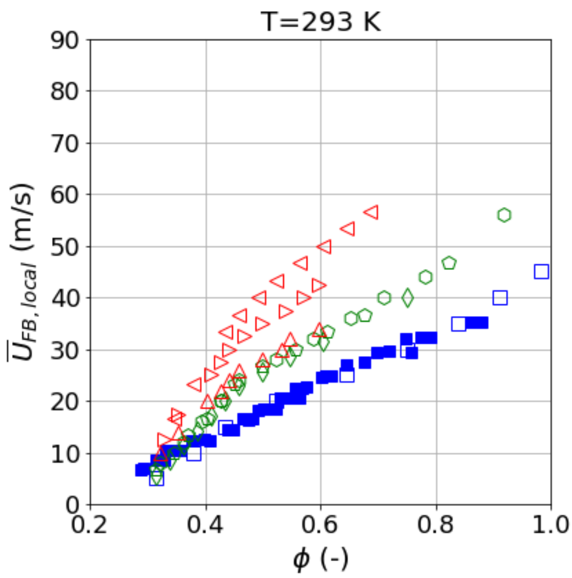
Figure 5.25 shows the results after adding the term $x\left(\partial p_{\text{flow}}/\partial x\right)$ to the backpressure expression to include the full effects of the underlying pressure gradient on the separation tendency as explained in section 5.2.3. Comparing Figs. 5.24 and 5.25 shows that the extra term does have a non-negligible effect on the results, increasing the backpressure effect and subsequently both the coefficient of pressure C_p and its derivative $\partial C_p/\partial x$.



(a) Critical gradient results.



(b) Inlet bulk velocity at flashback.



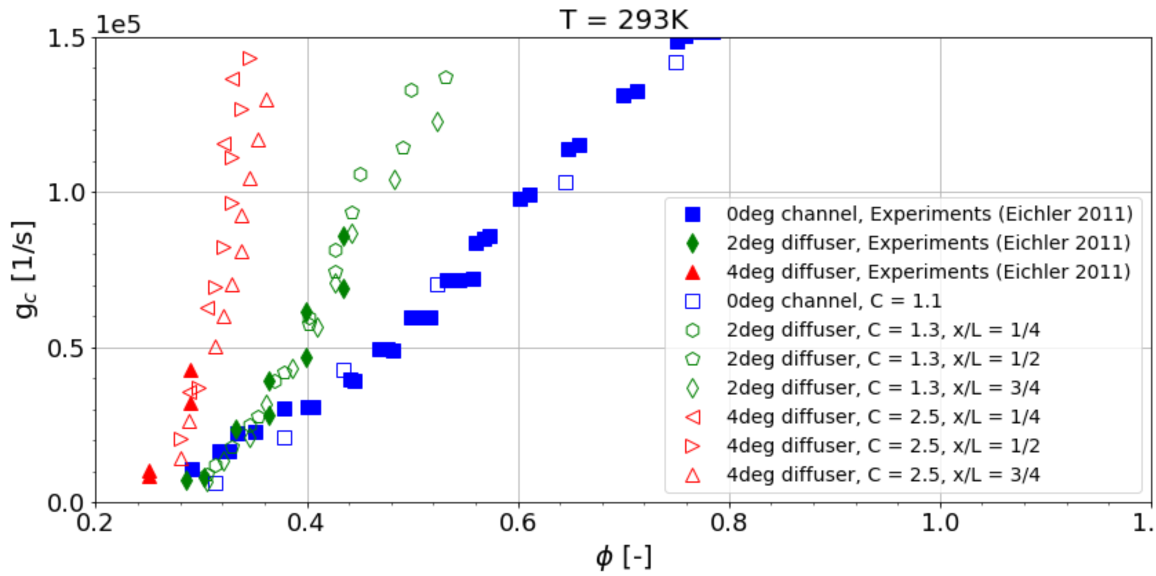
(c) Local bulk velocity at flashback.

Figure 5.25: Flashback limits from the BLF+CFD model for the 0° channel and 2° and 4° diffusers with an added backpressure term for the underlying adverse pressure profile.

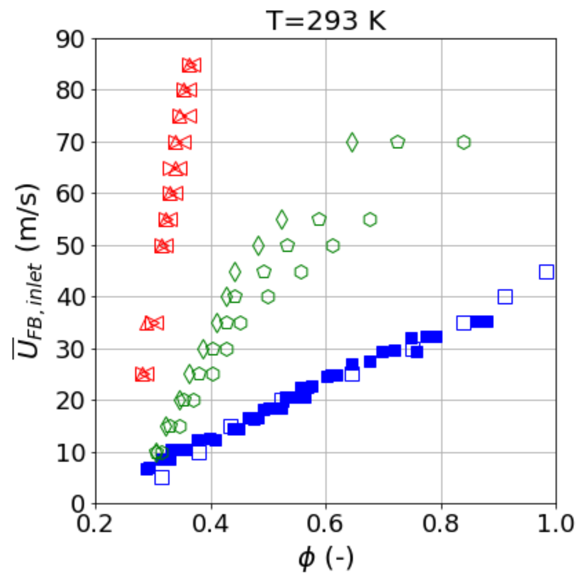
Figure 5.26 shows the same results as Fig. 5.25, but the turbulent flame speed has been tuned to the experimental data:

- 0° channel: $C = 1.1$
- 2° diffuser: $C = 1.3$
- 4° diffuser: $C = 2.5$

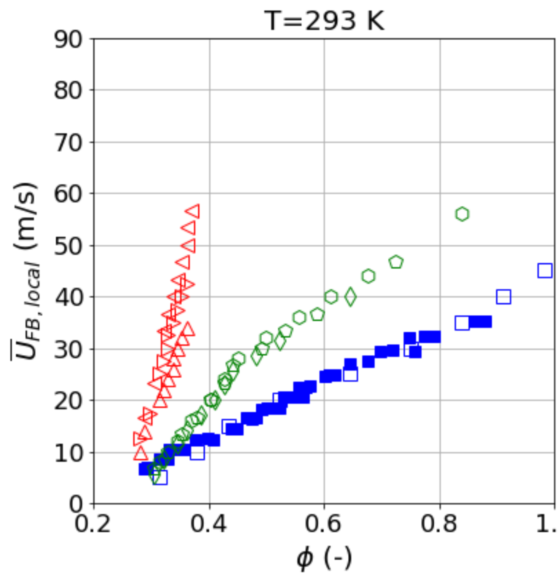
After tuning, the model predicts flashback in the diffusers quite well for the data available and seems to continue with a reasonable trend for where there is no data available. Eichler did not do measurements using preheated mixtures in the diffusers. There is a noticable split between the 4° diffuser results at the most downstream position $x/L = 3/4$ compared to the two upstream positions, due to the difference in the shape of the mean velocity profile.



(a) Critical gradient results.



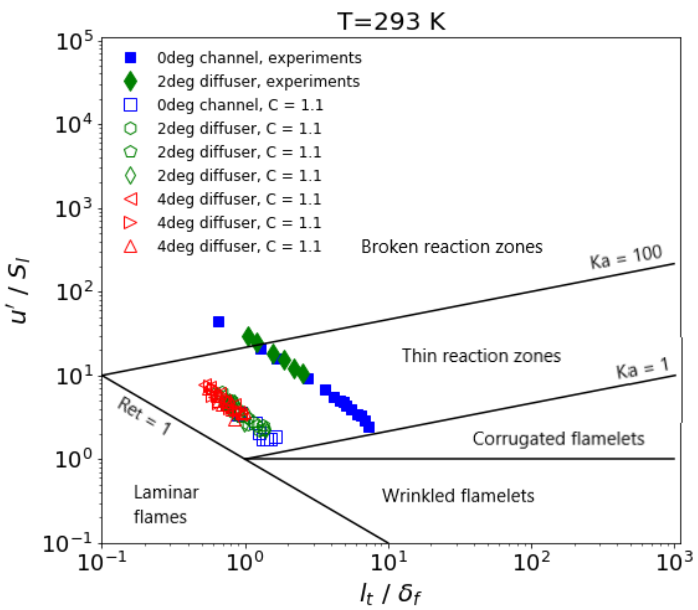
(b) Inlet bulk velocity at flashback.



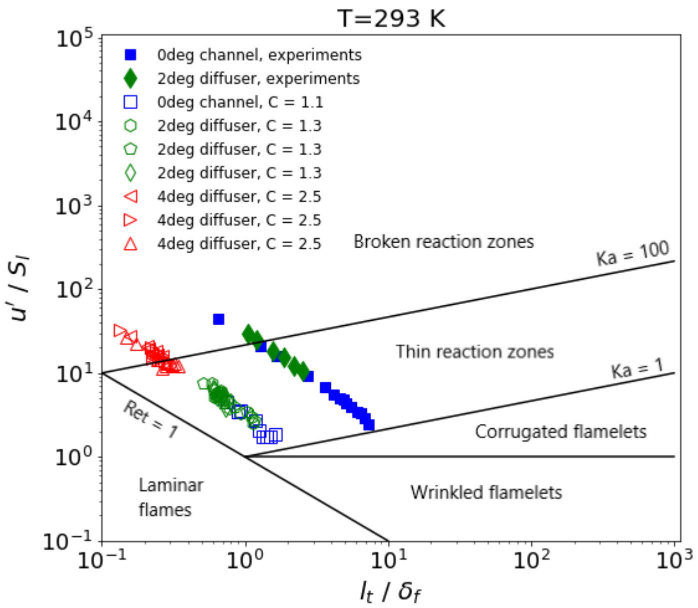
(c) Local bulk velocity at flashback.

Figure 5.26: Flashback limits from the BLF+CFD model for the 0° channel and 2° and 4° diffusers after tuning for the diffuser cases.

Figure 5.27 shows the calculated turbulent combustion regime diagram for the flames, compared to estimations from experiments for the 0° and 2° ducts. Eichler did not show estimations for the 4° diffuser for



(a) Before tuning.



(b) After tuning.

Figure 5.27: Calculated turbulent combustion regimes at flashback for the BLF+CFD model results compared to the estimated turbulent combustion regimes for the experimental data (see Fig. 3.8).

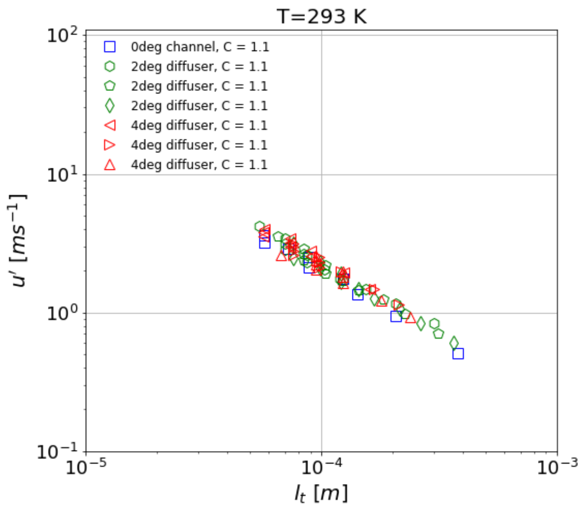
unknown reasons. The results are shown with and without tuning. Increasing C leads to a north-west shift on the diagram, since the laminar flame speed at flashback is lower with a higher C, leading to an increase of u'/S_l and a decrease of l_t/δ_f. The decrease of l_t/δ_f is due to the inverse dependency of the flame thickness on the flame speed in Eq. (3.35):

$$\delta_F = \frac{2\lambda_u}{\rho_u c_{p,u} S_{l,0}}$$

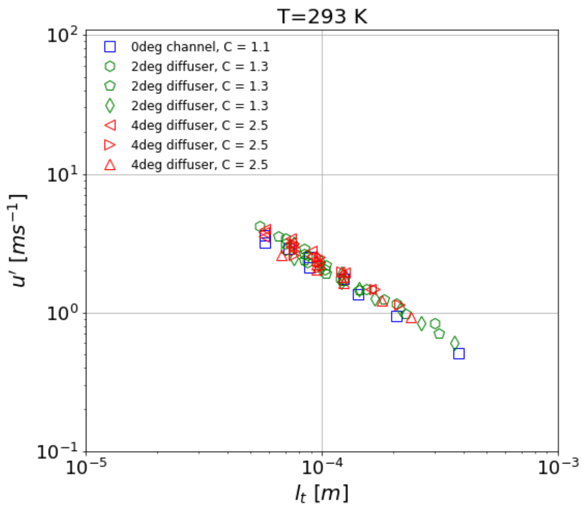
The integral length scale is computed with the following expression, suggested in Fluent's user manual [27]

$$l_t = 0.09^{3/4} \frac{k^{\frac{3}{2}}}{\epsilon} \tag{5.3}$$

at the wall distance of maximum turbulent intensity in the near-wall flow ($y+ \leqslant 40$), where the flashback is assumed to be initiated. The computed length scale and streamwise velocity fluctuations at flashback do not change with C as seen in Fig. 5.28. This shows that the north-west shift in the turbulent combustion



(a) Before tuning.



(b) After tuning.

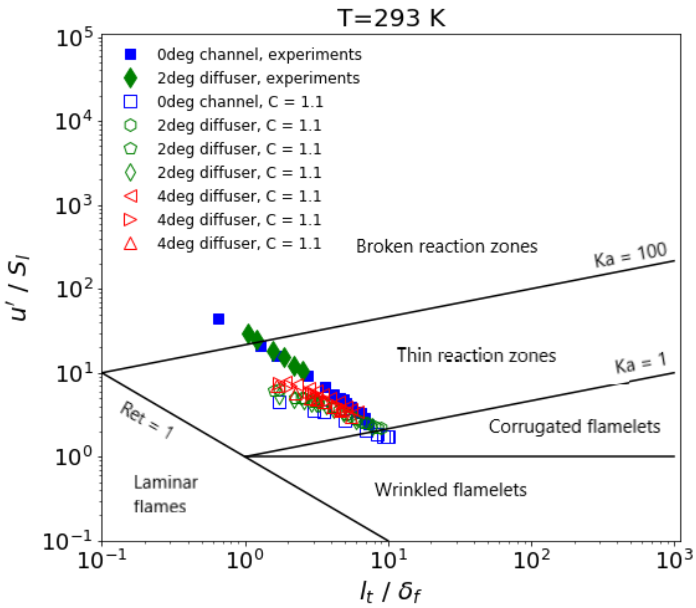
Figure 5.28: Streamwise velocity fluctuations u' versus the calculated integral length scale of turbulence l_t.

regime after tuning (see Fig. 5.27b) is only due to a different $S_{l,0}$ at flashback. Furthermore, the u' vs l_t data at flashback is very similar for all three cases (0°, 2° and 4° ducts) and, in particular, the computed turbulent length scale l_t at the assumed location of flashback is not changed appreciably. In section 2.2.2 it was noted that the C is expected to depend on the ratio of the turbulence length scale to the flame thickness l_t/δ_f. If the adverse pressure gradient changes the nature and scale of the turbulent eddies near the wall, then this does not seem to be captured with CFD using the Reynolds Averaged Navier-Stokes equations. Finding a correlation between the length scale l_t and the tuning needed is therefore not possible with these results.

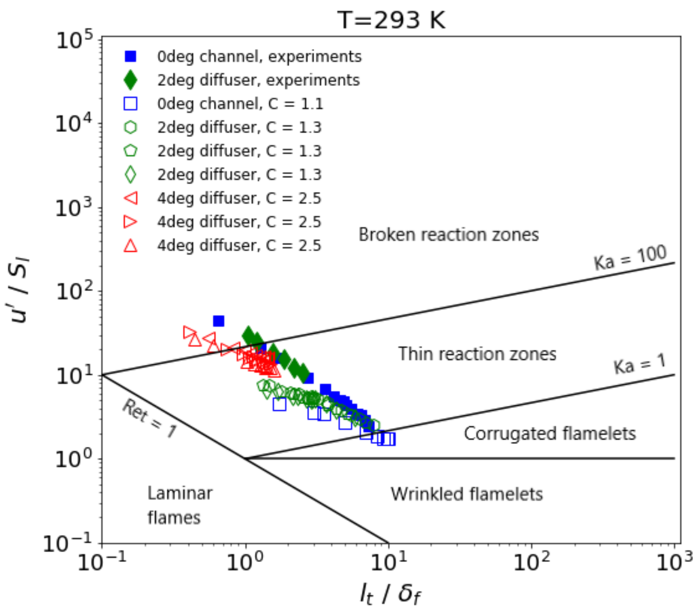
Eichler used

$$L^+ = \frac{\nu}{u_\tau} = 30 \tag{5.4}$$

for the length scale representing the average diameter of the quasi-streamwise vortices in the buffer layer according to Robinson [49]. Figure 5.29 shows how the flames are shifted to the right on the diagram when $L^+ = 30$ is used. The flames are mostly in the thin reaction zone regime, where the flamelet assumption for determining the turbulent flame speed is still considered valid as noted in section 2.2.2.



(a) Before tuning.



(b) After tuning.

Figure 5.29: Calculated turbulent combustion regimes at flashback using the same length scale as in the estimations from experiments.

5.2.6. Discussion

As discussed in section 3.1, Eichler reasoned that the increased flashback tendency (in terms of a higher critical gradient) in adverse pressure gradient flow was most likely caused by an increased frequency of low-speed streaks in the near-wall turbulent flow. He came to this conclusion after showing that both the near-wall mean velocity profile and the backpressure effect of the flame should not be influenced. The diffuser flow has been simulated and the mean velocity profile fitted. The most important parameters in the velocity profile fit are the combination of the outer velocity U_0 and the boundary layer thickness δ. This captures the retarded flow and results in increased flashback tendency for the same inlet bulk velocity. However, it only results in slightly higher wall gradients and local bulk velocities at flashback as shown in Fig. 5.24. The deviation from the channel results could simply be artefacts related to the fact that the velocity profile is not captured perfectly. That would be in agreement with Eichler's reasoning.
Adding a term to the backpressure expression for the underlying adverse pressure gradient does affect the critical gradient markedly, to the extent shown in Fig. 5.25. Now two effects of the underlying pressure gradient are included: the retardation of the mean flow which acts gradually over the whole length of the diffuser, and a local increment to the sharp, sudden pressure rise of the flame. The second effect will increase the flashback propensity in terms of the critical gradient.

However the C from the Damköhler turbulent flame speed closure (which was also left as a fitting constant in the original BLF model) needs to be increased to fully reproduce the experimental critical gradient results (see Fig. 5.26a). It is possible that the adverse pressure gradient changes the turbulent length scale versus flame thickness ratio causing higher flame speeds and increased flashback propensity in the diffusers. The dependency of the turbulent flame speed on this length scale ratio is not trivial. Lin [35] published two correlations of $S_t/S_{l,0}$ for hydrogen-rich fuels, and stated that their applicability depended on the turbulent Damköhler number:

$$\mathrm{Da} = \frac{l_t S_{l,0}}{u'\delta_f} \tag{5.5}$$

For Da > 1 the time scale of the chemistry is smaller than the turbulence and the dependency of l_t/δ_f is reported to be positive:

$$\frac{S_t}{S_{l,0}} = 0.8 \times \mathrm{Le}^{-1.38} \left(\frac{u'}{S_{l,0}}\right)^{0.80} \left(\frac{l_t}{\delta_f}\right)^{0.13} \tag{5.6}$$

and for Da < 1 the dependency is negative:

$$\frac{S_t}{S_{l,0}} = 4.6 \times \mathrm{Le}^{-1.84} \left(\frac{u'}{S_{l,0}}\right)^{0.59} \left(\frac{l_t}{\delta_f}\right)^{-0.28} \tag{5.7}$$

In Eq. (2.31) from Muppala [38], which performed well for a large data set, the dependency is positive.

In section 3.1.1 the effect of the top wall on the flame backpressure effect was discussed. Eichler reasoned that the presence of a top wall should increase the wake velocity, reduce the curvature of the streamlines in front of the flame and reduce the backpressure magnitude. Eichler noted that the downstream boundary layer in the diffuser was more susceptible to flashback than the upstream boundary layer, and linked it to the increased duct height. In the BLF model the backpressure magnitude is only a function of the expansion ratio and the turbulent flame speed so this effect is not captured. In fact, the BLF+CFD results (see Fig. 5.26a) show the opposite, since the downstream critical gradients are rather lower than the upstream ones.

6 Conclusion

The research questions were presented in section 1.2. Concluding remarks related to each question are given below.

- *Why does an adverse pressure gradient increase confined flame flashback propensity?*

This is still an open question. Eichler reasoned that there were no differences in the mean velocity profile or flame backpressure that could explain it. He suggested that the reason was an observed increased frequency of low-speed streaks in the boundary layer which increase the likelihood of upstream flame propagation.
After applying the BLF+CFD model, the impact of the shape of the mean velocity profile seems to be minimal or none. The impact of the underlying adverse pressure gradient is important, and could be part of the reason for higher critical gradients in diffusers by increasing the separation tendency. The largest effect seems to be due to a difference in the time-resolved nature of the near-wall turbulence and its effect on the flame speed.

- *How should Stratford's turbulent boundary layer separation criterion be applied to predict flame backpressure induced flow separation in fully developed channel flow?*

Stratford's separation criterion has been derived in section 4.2, resulting in a generalized criterion which can be applied to general boundary layers. In the original BLF model, the flow speed in the outer layer was overestimated leading to a more flashback resistant boundary layer. The generalized criterion was applied in the BLF model and validated resulting in lower turbulent flame speeds at separation and backpressure levels that are closer to reality as explained in the next section.

After validating the generalized separation criterion, the poor performance of the model at low equivalence ratios was investigated. The flame stretch effect on flame speed, which was included and discussed in detail by Hoferichter and Tober, was shown to be unimportant and mainly introduced large errors at low equivalence ratios. At the same time, the impact of thermo-diffusive flame instabilities and the formation of cellular flames is very significant, captured by Tober's proposed Lewis number correction.

- *Can the BLF model, by coupling to CFD software, be extended to predict flashback limits in new burner concepts? Can the effect of an underlying adverse pressure gradient in diffuser geometries be separated from the effect of flame backpressure in the prediction of turbulent flow separation and flashback?*

The BLF model was coupled to a commercial CFD code to be able to apply it to general geometries. The performance of turbulence models for diffuser flows was studied. The Standard k-ω (SKW) and the Shear Stress Transport k-ω (SST) models perform very well. The Reynolds Stress Model (RSM) is also a suitable model to predict anisotropic turbulence, but should only be used for diffusers with small opening angles.
A fast and simple method to extract and use the flow data has been implemented. A way to fit the boundary layer mean velocity profile and produce customized separation criterions using the generalized Stratford criterion has been proposed and implemented showing good results. The BLF+CFD model performs very well for the channel case.
In the diffusers, the BLF+CFD has been implemented by separating the effects of the underlying pressure gradient and the backpressure effect of the flame. This gives results which agree partly with the experimental

conclusions of Eichler. The shape of the mean velocity profile does not seem to have a large effect on flashback propensity but including the underlying pressure gradient in the backpressure expression does increase flashback propensity markedly. The turbulent flame speed needs to be tuned to reproduce the flashback limits in the diffuser, suggesting that there is still an effect of the adverse pressure gradient that needs to be investigated further.

6.1. Recent research and the limits of the BLF model

Recently new numerical studies on confined BLF in channels have been carried out at TU Munich.

In 2018 Endres and Sattelmayer [17] used Large Eddy Simulation (LES) with finite rate chemistry to simulate the flush wall stabilized turbulent premixed hydrogen flame in a 3-dimensional channel. The flashback limits from Eichler were accurately reproduced. The authors observed recirculation regions in front of the flame without flashback occurring. Flashback only occurred when the recirculation height significantly exceeded the quenching distance of the flame.

In August 2019 Endres and Sattelmayer [18] published a similar LES study focusing on the effect of operational pressure on confined flashback, which had not been studied before. They found that the flashback propensity increases with increasing pressure. At the same time, the size of the averaged separation zone in front of the flame, averaged flow deflection and the average turbulent flame speed decrease which should decrease flashback propensity. The quenching distance however decreases with increasing pressure increasing flashback propensity. This suggests that confined flame flashback is more complex than what is currently assumed in the BLF model, although it is not confirmed by experiments yet. The BLF model assumes that flashback propensity is a sole function of the size and shape of the backpressure effect and the resulting flow deflection and does not take into account the role of the variable quenching distance.

Endres and Sattelmayer also claim that the Stratford criterion is not valid for flame induced flow separation. The criterion is derived for a uniform pressure profile across the height of the channel while, in front of the flame, the wall-normal pressure profile is not uniform. They also note that the pressure rise ahead of the flame is overestimated by the one-dimensional treatment of the two-dimensional backpressure effect. Table 6.1 shows the values of pressure rise from the study compared to Hoferichter's results. The approximate pressure values from this work using the generalized separation criterion results from Fig. 4.9 are also shown, showing that they are closer to the LES simulations.

Table 6.1: Values for the pressure increase in front of the channel confined flame in two iterations of the BLF model vs. the LES simulations by Endres and Sattelmayer [18].

	U_{bulk} [m/s]	**10**	**20**	**30**
	ϕ [-]	**0.38**	**0.55**	**0.7**
Endres & Sattelmayer (LES) [18]	**Δp [Pa]**	5.5	22.9	63.7
This work	**Δp [Pa]**	10	43	75
Hoferichter [24]	**Δp [Pa]**	32.2	106.4	214.3

6.2. Recommendations

Recent findings by TU Munich on the influence of pressure on confined boundary layer flashback indicate that the BLF model might need to be updated accordingly. While it does a good job of describing the effects of fuel composition and preheating, and is able to capture the effect of adverse pressure gradients with some tuning, it assumes increased flashback propensity is always linked to increased flow separation tendency. The role of the quenching distance should be the focus of new modelling efforts for validation at gas turbine relevant pressures. The effect of operational pressure needs to be further investigated.

The increased flashback propensity in terms of critical gradients in the diffuser flows versus the channel seems to be linked to differences in the time-resolved near-wall turbulence. Both reactive and isothermal diffuser flow should be studied further, experimentally and numerically, with the goal of describing the increased flashback propensity in adverse pressure gradient flow.

Bibliography

[1] Carbon capture and storage. URL en.wikipedia.org/wiki/Carbon_capture_and_storage.

[2] Natural gas and its advantages. URL shell.com/energy-and-innovation/natural-gas/natural-gas-and-its-advantages.html.

[3] HA technology now available at industry-first 64 percent efficiency, 2017. URL genewsroom.com/press-releases/ha-technology-now-available-industry-first-64-percent-efficiency.

[4] World energy balances: Overview. Technical report, International Energy Agency, 2019. URL webstore.iea.org/download/direct/2710?fileName=World_Energy_Balances_2019_Overview.pdf.

[5] John D. Anderson Jr. *Fundamentals of Aerodynamics*. McGraw-Hill, 2011.

[6] Georg Martin Baumgartner. *Flame Flashback in Premixed Hydrogen-Air Combustion Systems*. PhD thesis, 2014.

[7] J. K. Bechtold and M. Matalon. The dependence of the Markstein length on stoichiometry. *Combustion and Flame*, 2001. ISSN 00102180. doi: 10.1016/S0010-2180(01)00297-8.

[8] Ali Cemal Benim and Khawar J. Syed. *Flashback Mechanisms in Lean Premixed Gas Turbine Combustion*. 2014. ISBN 9780128008263. doi: 10.1016/C2013-0-18847-2.

[9] H. Blasius. Grenzschichten in Flussigkeiten mit kleiner Reibung, 1908. ISSN 00218979.

[10] K. N. C. Bray. Studies of the Turbulent Burning Velocity. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 1990. ISSN 1364-5021. doi: 10.1098/rspa.1990.0133.

[11] Eoin Burke, Felix Güthe, and Rory Monaghan. A comparison of turbulent flame speed correlations for hydrocarbon fuels at elevated pressures. 2016.

[12] Tuncer Cebeci, G. J. Mosinskis, and A. M. O. Smith. Calculation of Separation Points in Incompressible Turbulent Flows. *Journal of Aircraft*, 1972. ISSN 0021-8669. doi: 10.2514/3.59049.

[13] Stefan Dankelman, Marko Draskic, Lim Chi Ho, and Rashed Muslem. The effects of scaling of the FlameSheet combustor. Technical report, TU Delft, 2018.

[14] C. Eichler. Flame Flashback in Wall Boundary Layers of Premixed Combustion Systems. 2011. URL www.td.mw.tum.de/fileadmin/w00bso/www/Forschung/Dissertationen/Eichler.pdf.

[15] Christian Eichler, Georg Baumgartner, and Thomas Sattelmayer. Experimental Investigation of Turbulent Boundary Layer Flashback Limits for Premixed Hydrogen-Air Flames Confined in Ducts. *Journal of Engineering for Gas Turbines and Power*, 2012. ISSN 07424795. doi: 10.1115/1.4004149.

[16] Samy M. El-Behery and Mofreh H. Hamed. A comparative study of turbulence models performance for separating flow in a planar asymmetric diffuser. *Computers and Fluids*, 2011. ISSN 00457930. doi: 10.1016/j.compfluid.2011.01.009.

[17] A. Endres and T. Sattelmayer. Large Eddy simulation of confined turbulent boundary layer flashback of premixed hydrogen-air flames. *International Journal of Heat and Fluid Flow*, 72:151–160, aug 2018. ISSN 0142727X. doi: 10.1016/j.ijheatfluidflow.2018.06.002.

[18] Aaron Endres and Thomas Sattelmayer. Numerical Investigation of Pressure Influence on the Confined Turbulent Boundary Layer Flashback Process. *Fluids*, 4(3):146, aug 2019. doi: 10.3390/fluids4030146.

[19] Fluent. ANSYS Fluent 12.0 user's guide. *Ansys Inc*, 2009. ISSN ISO 9001:2008. doi: 10.1016/0140-3664(87)90311-2.

[20] S. Goldstein. *Modern Developments in Fluid Dynamics.* Oxford University Press, 1938.

[21] D.G. Goodwin, H.K. Moffat, and R.L. Speth. Cantera: An open-source suite of tools for problems involving chemical kinetics, thermodynamics, and transport processes., 2019. URL `www.cantera.org`.

[22] A. Gruber, J. H. Chen, D. Valiev, and C. K. Law. Direct numerical simulation of premixed flame boundary layer flashback in turbulent channel flow. *Journal of Fluid Mechanics*, 709:516–542, oct 2012. ISSN 0022-1120. doi: 10.1017/jfm.2012.345. URL `www.journals.cambridge.org/abstract_S002211201200345X`.

[23] Martin Hertzberg. Selective diffusional demixing: Occurrence and size of cellular flames, 1989. ISSN 03601285.

[24] Vera Hoferichter. Boundary Layer Flashback in Premixed Combustion Systems. 2017. URL `mediatum.ub.tum.de/doc/1336042/1336042.pdf`.

[25] Vera Hoferichter, Christoph Hirsch, and Thomas Sattelmayer. Prediction of Confined Flame Flashback Limits Using Boundary Layer Separation Theory. *Journal of Engineering for Gas Turbines and Power*, 139(2):021505, 2016. ISSN 0742-4795. doi: 10.1115/1.4034237. URL `gasturbinespower.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4034237`.

[26] Vera Hoferichter, Christoph Hirsch, Thomas Sattelmayer, Alireza Kalantari, Elliot Sullivan-Lewis, and Vincent McDonell. Comparison of Two Methods to Predict Boundary Layer Flashback Limits of Turbulent Hydrogen-Air Jet Flames. *Flow, Turbulence and Combustion*, 2018. ISSN 15731987. doi: 10.1007/s10494-017-9882-2.

[27] Fluent Inc. Fluent 6.3 - User's guide. Technical report, 2006.

[28] Christopher Jainski, Martin Rißmann, Suad Jakirlic, Benjamin Böhm, and Andreas Dreizler. Quenching of Premixed Flames at Cold Walls: Effects on the Local Flow Field. *Flow, Turbulence and Combustion*, 2018. ISSN 15731987. doi: 10.1007/s10494-017-9836-8.

[29] Satoshi Kadowaki. Flame Velocity of Cellular Flames at Low Lewis Numbers. *Combustion Science and Technology*, 162(1):223–234, jan 2001. ISSN 0010-2202. doi: 10.1080/00102200108952142. URL `www.tandfonline.com/doi/abs/10.1080/00102200108952142`.

[30] Alireza Kalantari and Vincent McDonell. Boundary layer flashback of non-swirling premixed flames: Mechanisms, fundamental research, and recent advances, 2017. ISSN 03601285.

[31] H. J. Kaltenbach, M. Fatica, R. Mittal, T. S. Lund, and P. Moin. Study of flow in a planar asymmetric diffuser using large-eddy simulation. *Journal of Fluid Mechanics*, 1999. ISSN 00221120. doi: 10.1017/S0022112099005054.

[32] Arnold M. Kuethe. *Foundations of aerodynamics : bases of aerodynamic design.* 1998. ISBN 0471129194. doi: 10.1016/S0360-3016(03)00222-0.

[33] Pijush K. Kundu, Ira M. Cohen, and David R. Dowling. Fluid Mechanics 6th Edition. *Fluid Mechanics*, 2016. ISSN 1098-6596. doi: 10.1016/B978-0-12-405935-1.01001-7.

[34] Bernard Lewis and Guenther von Elbe. Stability and Structure of Burner Flames. *The Journal of Chemical Physics*, 2005. ISSN 0021-9606. doi: 10.1063/1.1723808.

[35] Yu Chun Lin, Peter Jansohn, and Konstantinos Boulouchos. Turbulent flame speed for hydrogen-rich fuel gases at gas turbine relevant conditions. *International Journal of Hydrogen Energy*, 2014. ISSN 03603199. doi: 10.1016/j.ijhydene.2014.10.037.

[36] George H. Markstein. *Nonsteady Flame Propagation.* 1964.

[37] C. Meneveau and T. Poinsot. Stretching and quenching of flamelets in premixed turbulent combustion. *Combustion and Flame*, 1991. ISSN 00102180. doi: 10.1016/0010-2180(91)90126-V.

[38] Siva Muppala, Bhuvaneswaran Manickam, and Friedrich Dinkelacker. A Comparative Study of Different Reaction Models for Turbulent Methane/Hydrogen/Air Combustion. *Journal of Thermal Engineering*, 1: 367, 2015. doi: 10.18186/jte.60394.

[39] Yasutaka Nagano, Toshihiro Tsuji, and Tomoya Houra. Structure of turbulent boundary layer subjected to adverse pressure gradient. *International Journal of Heat and Fluid Flow*, 1998. ISSN 0142727X. doi: 10.1016/S0142-727X(98)10013-9.

[40] Marcus Ó Conaire, Henry J. Curran, John M. Simmie, William J. Pitz, and Charles K. Westbrook. A comprehensive modeling study of hydrogen oxidation. *International Journal of Chemical Kinetics*, 2004. ISSN 05388066. doi: 10.1002/kin.20036.

[41] Shinnosuke Obi. Experimental and Computational Study of Turbulent Separating Flow in an Asymmetric Plane Diffuser. 1993.

[42] R. Pecnik, G. Otero Rodriguez, and A. Patel. RANS channel, 2018. URL `github.com/Fluid-Dynamics-Of-Energy-Systems-Team/RANS_Channel`.

[43] Norbert Peters. *Turbulent combustion*. Cambridge University Press, 2000. ISBN 9780511612701.

[44] Norbert Peters. Lecture notes on combustion theory, 2010. URL `cefrc.princeton.edu/combustion-summer-school/archived-programs/2010-summer-school/lecture-notes`.

[45] Philip A.E. Pogge von Strandmann, Kevin W. Burton, Sandra O. Snæbjörnsdóttir, Bergur Sigfússon, Edda S. Aradóttir, Ingvi Gunnarsson, Helgi A. Alfredsson, Kiflom G. Mesfin, Eric H. Oelkers, and Sigurður R. Gislason. Rapid CO 2 mineralisation into calcite at the CarbFix storage site quantified using calcium isotopes. *Nature Communications*, 10(1), dec 2019. ISSN 20411723. doi: 10.1038/s41467-019-10003-8.

[46] Thierry Poinsot and Denis Veynante. *Theoretical and Numerical Combustion*. 2 edition, 2005. ISBN 1-930217-10-2.

[47] Stephen B. Pope. Turbulent Flows. *Measurement Science and Technology*, 2001. ISSN 0957-0233. doi: 10.1088/0957-0233/12/11/705.

[48] L.A. Meyer R.K. Pachauri. *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva, Switzerland, 2014. ISBN 9789291691432.

[49] S. Robinson. Coherent Motions In The Turbulent Boundary Layer. *Annual Review of Fluid Mechanics*, 1991. ISSN 00664189. doi: 10.1146/annurev.fluid.23.1.601.

[50] Herman Schlichting. Tech. Mem. Nat. Adv. Comm. Aero., Wash., no. 1218 (transl.). Technical report, 1941.

[51] Herman Schlichting. *Boundary-layer theory*. McGraw-Hill, 7 edition, 1979.

[52] Herman Schlichting and Klaus. Gersten. *Boundary-layer theory*. Springer, 9 edition, 2017.

[53] D. B. Spalding. A Single Formula for the “Law of the Wall”. *Journal of Applied Mechanics*, 2011. ISSN 00218936. doi: 10.1115/1.3641728.

[54] B. S. Stratford. Flow in the laminar boundary layer near separation. 1957. URL `naca.central.cranfield.ac.uk/reports/arc/rm/3002.pdf`.

[55] B. S. Stratford. The prediction of separation of the turbulent boundary layer. *Journal of Fluid Mechanics*, 1959. ISSN 14697645. doi: 10.1017/S0022112059000015.

[56] Peter J. Stuttaford, Stephen Jennings, Andrew Green, Ryan McMahon, Yan Chen, and Hany Rizkalla. Flamesheet combustor, 2003. URL `https://patents.google.com/patent/US6935116B2/en`.

[57] D. Tam. Advanced Fluid Dynamics. Course code ME45040, 2018.

[58] Luis Tay-Wo-Chong, M. Zellhuber T. Komarek, J. Lenz, C. Hirsch, and W. Polifke. Influence of strain and heat loss on flame stabilization in a non-adiabatic combustor. In *Proceedings of the European Combustion Meeting*, 2009.

[59] Joeri Tober. *Boundary layer flashback prediction of a low emissions full hydrogen burner for gas turbine applications.* Master thesis, TU Delft, 2019.

[60] OPRA Turbines. Hydrogen Subsidy Project Awarded, 2019. URL `https://www.opraturbines.com/hydrogen-subsidy-project-awarded/`.

[61] Stephen R Turns. *An introduction to combustion: concepts and applications.* 2000.

[62] J.P. van Buijtenen, Wilfried Visser, T. Tinga, S. Shakariyants, F. Montella, and J. Singh. Reader: Gas Turbines, WB4420/4421 - Thermodynamics and Gas Turbines, AE3-235, 2007.

[63] Various. Rankine-Hugoniot conditions. URL `en.wikipedia.org/wiki/Rankine-Hugoniot_conditions`.

[64] D. Veynante, J. Piana, J. M. Duclos, and C. Martel. Experimental analysis of flame surface density models for premixed turbulent combustion. *Symposium (International) on Combustion*, 1996. ISSN 00820784. doi: 10.1016/S0082-0784(96)80243-8.

[65] Frank M. White. *Viscous Fluid Flow.* McGraw-Hill, 2006.

Appendices

A

Appendix

A.1. Supplementary notes

A.1.1. Coefficients for polynomial fits

In the original confined boundary layer flashback model, Hoferichter [24] uses Eq. (3.24) to fit experimental data for streamwise turbulent velocity fluctuations u'. The coefficients are:

$$\begin{aligned} a_0 &= 2.661 \\ a_1 &= -7.211 \\ a_2 &= 7.600 \\ a_3 &= -2.900 \\ a_4 &= 0.472 \\ a_5 &= -0.028 \end{aligned}$$

A.1.2. Flame stretch with isotropic turbulence

The flame stretch rate κ is defined as the normalized temporal change of flame surface area A_F [24]:

$$\kappa = \frac{1}{A_F}\frac{dA_F}{dt} \tag{A.1}$$

Flame stretch due to flow strain is further divided into a mean component and turbulent component $\kappa_{\text{strain}} = \kappa_{\text{mean}} + \kappa_{\text{turb}}$ such that:

$$\kappa = \kappa_{\text{mean}} + \kappa_{\text{turb}} + \kappa_{\text{curv}} \tag{A.2}$$

Assuming isotropic turbulence Hoferichter uses the following simplified expression for κ_{mean} from Chong et al. [58]:

$$\kappa_{\text{mean}} = \frac{2}{3}\frac{\partial u_i}{\partial x_i} \tag{A.3}$$

By further assuming incompressible fully developed flow, κ_{mean} is zero. For the turbulent contribution to strain rate Hoferichter uses the Intermittent Turbulent Net Flame Stretch (ITNFS) model by Meneveau and Poinsot [37]:

$$\kappa_{\text{turb}} = \Gamma_K \frac{\epsilon}{k} \tag{A.4}$$

with turbulent dissipation rate $\epsilon = u'^3/\Lambda$ and turbulent kinetic energy $k = 3/2u'^2$:

$$\kappa_{\text{turb}} = \frac{2}{3}\Gamma_K \frac{u'}{\Lambda} \tag{A.5}$$

with the turbulence macroscale Λ based on the hydraulic diameter:

$$\Lambda = 0.07 D_h \tag{A.6}$$

valid for turbulent duct flows [24] and an efficiency function, also from Meneveau and Poinsot [37]:

$$\log_{10}(\Gamma_K) = \frac{-1}{s+0.4} e^{-(s+0.4)} + (1 - e^{-(s+0.4)}) \left(\frac{2}{3} \left(1 - 0.5 e^{\left(-(u'/S_{l,0})^{1/3} \right)} \right) s - 0.11 \right) \tag{A.7}$$

with

$$s = \log_{10}\left(\frac{\Lambda}{\delta_F} \right) \tag{A.8}$$

Finally, Hoferichter uses the following expression from Veynante et al. [64]:

$$\kappa_{\text{curv}} \approx S_{l,0} \frac{0.5 - c}{L} \tag{A.9}$$

where c is the Reynolds averaged reaction progress variable and L is the flame wrinkling length, given by Bray [10] as

$$L = \Lambda \frac{S_{l,0}}{u'} \tag{A.10}$$

resulting in

$$\kappa_{\text{curv}} \approx \frac{u' \left(\frac{1}{2} - c \right)}{\Lambda} = \frac{1}{2} \frac{u'}{\Lambda} \tag{A.11}$$

as the maximum absolute value since Hoferichter states that the maximum flame stretch rate will be found at the beginning and end of the reaction ($c = 0$ and $c = 1$). Hoferichter's final expression for the total flame stretch rate κ is then

$$\kappa = \frac{2}{3} \Gamma_K \frac{u'}{\Lambda} + \frac{1}{2} \frac{u'}{\Lambda} \tag{A.12}$$

A.1.3. Flame stretch with anisotropic turbulence

The original confined boundary layer flashback model from Hoferichter (cf. section 3.3) includes Eq. (A.3) for the mean strain rate:

$$\kappa_{\text{mean}} = \frac{2}{3} \frac{\partial u_i}{\partial x_i}$$

and Eq. (A.5) for the turbulent strain rate, assuming isotropic turbulence:

$$\kappa_{\text{turb}} = \Gamma_K \frac{\epsilon}{k} = \frac{2}{3} \Gamma_K \frac{u'}{\Lambda}$$

Tober [59] noted that the turbulence in the channel is anisotropic. Chong et al. define the strain rate as:

$$\kappa_{\text{strain}} = \kappa_{\text{mean}} + \kappa_{\text{turb}} \tag{A.13}$$

$$\kappa_{\text{mean}} = \left(\delta_{ij} - n_i n_j \right) \frac{\partial \overline{u}_i}{\partial x_j} \tag{A.14}$$

$$\kappa_{\text{turb}} = \left(\delta_{ij} - n_i n_j \right) \frac{\partial u'_i}{\partial x_j} \tag{A.15}$$

where $n_i n_j = \frac{\overline{u'_i u'_j}}{2k}$ resulting in

$$\begin{aligned}
\kappa_{\text{mean}} = \frac{\partial \overline{u_i}}{\partial x_i} - \frac{\overline{u'_i u'_j}}{2k} \frac{\partial \overline{u_i}}{\partial x_j} \\
= \left(\frac{\partial \overline{u}}{\partial x} + \frac{\partial \overline{v}}{\partial y} + \frac{\partial \overline{w}}{\partial z} \right) - \frac{1}{2k} \Bigg[\overline{u'u'} \frac{\partial \overline{u}}{\partial x} + \overline{u'v'} \frac{\partial \overline{u}}{\partial y} + \overline{u'w'} \frac{\partial \overline{u}}{\partial z} \\
+ \overline{v'u'} \frac{\partial \overline{v}}{\partial x} + \overline{v'v'} \frac{\partial \overline{v}}{\partial y} + \overline{v'w'} \frac{\partial \overline{v}}{\partial z} \\
+ \overline{w'u'} \frac{\partial \overline{w}}{\partial x} + \overline{w'v'} \frac{\partial \overline{w}}{\partial y} + \overline{w'w'} \frac{\partial \overline{w}}{\partial z} \Bigg]
\end{aligned} \tag{A.16}$$

By taking into account symmetry around the z-axis in the channel, incompressibility and that the flow is fully developed Tober crosses out terms and arrives at

$$\kappa_{mean} = -\frac{\overline{u'v'}}{2k}\frac{\partial \overline{u}}{\partial y} \tag{A.17}$$

The anisotropic turbulent strain rate can again be modeled using the Intermittent Turbulent Net Flame Stretch (ITNFS) model by Meneveau and Poinsot [37]:

$$\kappa_{turb} = \Gamma_K \frac{\epsilon}{k} \tag{A.18}$$

with the turbulent dissipation rate $\epsilon = \frac{u'v'w'}{\Lambda}$ and turbulent kinetic energy $k = \frac{1}{2}\left(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}\right)$.

A.1.4. Flame instabilities and flame speed of cellular flames

Two mechanisms are responsible for the instability of lean premixed hydrogen flames: the hydrodynamic Darrieus Landau (DL) instability and thermo-diffusive instabilities. The DL instability can be explained by the expansion of incoming reactant flow in front of the flame due to a temperature jump and flow expansion over the flame front. If the flame front is perturbed the reactant flow expands sooner where the flame front is convex, and diverges towards the concave areas. This causes the reactant velocity to decrease locally in front of the convex bulge but increase where the flow is concave, which affects the local flame propagation and drives the instability by curving the flame front even more.

Figure A.1 from Poinsot and Veynante [46] illustrates thermo-diffusive instability. It is caused by unequal

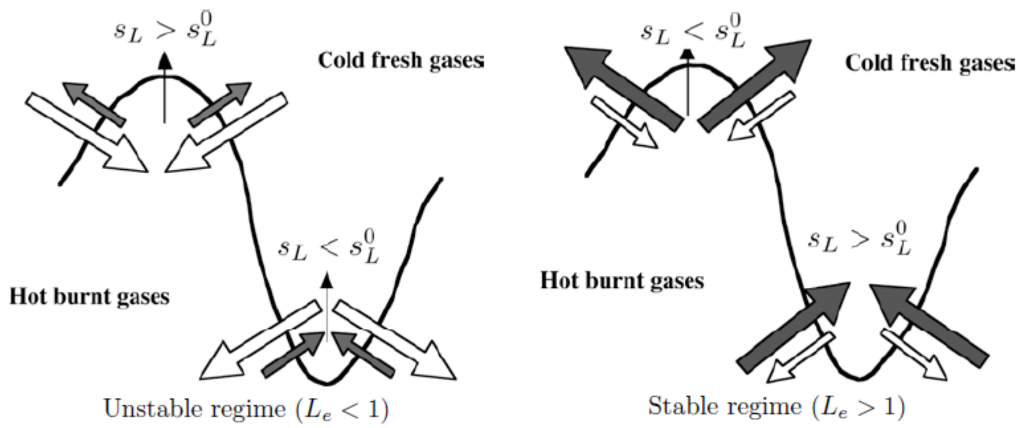


Figure A.1: Illustration of thermo-diffusive instabilities. The grey arrows represent diffused heat, the white arrows represent diffused mass. Source: Poinsot and Veynante [46].

thermal (α) and mass (D) diffusivities. The ratio between these two diffusivities is called the Lewis number:

$$\text{Le} = \frac{\alpha}{D}$$

At a convex part of the flame front, the heat flux into the preheat zone is diverging which lowers the flame speed. However, the diffusion of reactants from the preheat zone into the reaction zone is convergent which increases flame speed. The ratio of thermal diffusivity to mass diffusivity captured by the Lewis number will determine the net effect of the two competing effects of flame curvature. For Le < 1 the net result is an increase of laminar flame speed. For concave parts of the flame fronts, if Le < 1, the net result is a local decrease of laminar flame speed due to an opposite curvature. Therefore Le < 1 always results in growth of perturbations and an unstable flame front. If Le > 1 the flame front will instead straighten out and is therefore stable.

Tober mentions that that thermo-diffusive instabilities are more important than hydrodynamic instabilities for lean hydrogen-air mixtures with Le < 1. The Markstein number Ma = $\frac{L_M}{\delta_f}$ also plays a role since it determines the stability with regards to stretch rate. Negative Markstein numbers will contribute to instability by increasing the stretched laminar flame speed for positive stretch rates. Conversely, positive Markstein lengths contribute to a stable flame front [8].

In the case of lean hydrogen-air flames, a negative Markstein length and a Lewis number less than unity will both contribute to an unstable flame front. Tober notes that lean hydrogen-air mixtures will form a stable cellular flame structure. Stable cellular flame structures are formed where the most diffusive component in the fuel-air mixture is also the deficient component, according to Hertzberg [23]. In hydrogen-air mixtures the hydrogen is the most diffusive component and thus stable cellular flame structures will form at lean conditions. For most other fuels the most diffusive component is oxygen and in those cases, the instability will lead to a cellular flame structure only for rich mixtures.

The flame instabilities and the switch to a cellular flame structure will result in an increased turbulent burning velocity. To include this effect in the BLF model, Tober suggests a modification using a correlation from Kadowaki's data [29] given by Eq. (3.36). He calls this modification the Lewis number correction. Tober suggests that this data can be used in the range of $0.5 \leq \mathrm{Le} \leq 1$. Above $\mathrm{Le} = 1$ the lean hydrogen-air flames do not show a cellular flame structure. Below $\mathrm{Le} = 0.5$ Tober assumes that the cellular flame self-stabilizes and the trend levels off, so the value at $\mathrm{Le} = 0.5$ is used for $\mathrm{Le} \leq 0.5$.

$$S_{t,\mathrm{corrected}} = S_t \left[0.6052 \left(\frac{1}{\mathrm{Le}}^2 \right) - 1.1314 \left(\frac{1}{\mathrm{Le}} \right) + 1.5224 \right]$$

A.1.5. How to output profiles from Fluent

In section 5.1, the Fluent solution data was processed using a Python interpolation scheme written by Tober [59]. The solution data was interpolated from the mesh nodes to the desired cross-sectional profiles. In section 5.2 the interpolation is done in Fluent using the profile export option. This built-in method is fast and easy to use:

- In Fluent's post-processing environment: create a surface at the desired location.
- Choose File - Export - Profile, select your surface, select the desired values and save as .prof or .csv
- Change the file ending manually from .prof to .txt

A.2. Codes

A.2.1. Functions

Function: LFS.py
Author: J. Tober (2019)
Description: The function calculates the unstretched laminar flame speed based on Eq. (3.28) and uses coefficients tabulated by Hoferichter [24] in LaminarFlameSpeed.txt. The coefficients are based on experimental values for room temperature mixtures and one dimensional free flame simulations (in Cantera [21]) at preheated temperatures.

```
import numpy as np
import math
A = np.ndarray(shape=(20,25,5))

file_name="LaminarFlameSpeed"
f=open(file_name+".txt",'r+')
lines=f.readlines()

for l in range(len(lines)):
    lines_split = lines[l].split()
    P = math.floor(l/20)
    L = l - P*20
    A[L,0:25,P] = lines_split

def inter(phi,T,p):
    i = math.floor((phi - 0.35)/0.05)
    if i > -1:
        j = math.floor((T - 273)/25.)
        if p>=7:
            k=3
        else:
            k=math.floor((p-1)/2.)
        p_list = [1,3,5,7,20]
        i = int(i)
        j = int(j)
        k = int(k)
        S_l0 = A[i,j,k] + (phi-(0.35+i*0.05))/(0.05) * (A[i+1,j,k]-A[i,j,k])\
        + (T-(273+j*25.))/(25.) * (A[i,j+1,k]-A[i,j,k]) + (p-p_list[k])/\
```

```
        (p_list[k+1]-p_list[k]) * (A[i,j,k+1]-A[i,j,k])
    else:
        j = math.floor((T - 273)/25.)
        i=0
        if p>=7:
            k=3
        else:
            k=math.floor((p-1)/2.)
        p_list = [1,3,5,7,20]
        i = int(i)
        j = int(j)
        k = int(k)
        S_l0 = A[i,j,k] + (phi-(0.35+i*0.05))/(0.05) * (A[i+1,j,k]-A[i,j,k])\
        + (T-(273+j*25.))/(25.) * (A[i,j+1,k]-A[i,j,k]) + (p-p_list[k])/\
        (p_list[k+1]-p_list[k]) * (A[i,j,k+1]-A[i,j,k])
        if S_l0 < 0:
            phi = 0.3
            S_l0 = A[i,j,k] + (phi-(0.35+i*0.05))/(0.05) * \
            (A[i+1,j,k]-A[i,j,k]) + (T-(273+j*25.))/(25.) * \
            (A[i,j+1,k]-A[i,j,k]) + (p-p_list[k])/(p_list[k+1]-p_list[k]) \
            * (A[i,j,k+1]-A[i,j,k])
            S_l0 = 0.5 * S_l0
    return(S_l0)
```

Function: onedfs.py
Author: O.H. Bjornsson (2019)
Description: The function gets the unstretched laminar flame speed based on one dimensional free flame simulations using Cantera [21] and the reaction mechanism of Ó Conaire [40].

```
# -*- coding: utf-8 -*-
"""
Get one dimensional free flame flame speeds using the reaction mechanism
from O'Contaire
"""
import cantera as ct
def onedfs(phi,Tin,p):
    #reactants = 'H2:1.1, O2:1, AR:5'  # premixed gas composition
    reactants = 'H2:'+str(2*phi)+', O2:1, N2:3.76'
    #reactants = 'H2:'+str(2*phi)+', O2:1, AR:3.76'

    width = 0.03  # m
    loglevel = 1  # amount of diagnostic output (0 to 8)

    # IdealGasMix object used to compute mixture properties,
    # set to the state of the
    # upstream fuel-air mixture
    #gas = ct.Solution('h2o2.xml')
    #gas = ct.Solution('gri30.xml')
    #gas = ct.Solution('UCSD.xml')
    gas = ct.Solution(\'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
                      'H2 Flashback/OConaire Reaction Mechanism/chem.cti')
    gas.TPX = Tin, p, reactants

    # Set up flame object
    f = ct.FreeFlame(gas, width=width)
    f.set_refine_criteria(ratio=3, slope=0.06, curve=0.12)
    #f.show_solution()

    # Solve with mixture-averaged transport model
    #f.transport_model = 'Mix'

    #f.solve(loglevel=loglevel, auto=True)

    # Solve with the energy equation enabled
    #f.save('h2_adiabatic.xml', 'mix',\
    #  'solution with mixture-averaged transport')
    #f.show_solution()
    #print('mixture-averaged flamespeed = {0:7f} m/s'.format(f.u[0]))

    # Solve with multi-component transport properties
    f.transport_model = 'Multi'
    #f.solve(loglevel) # don't use 'auto' on subsequent solves
    f.solve(loglevel=loglevel, auto=True)
    #f.show_solution()
    #print('multicomponent flamespeed = {0:7f} m/s'.format(f.u[0]))
    #f.save('h2_adiabatic.xml','multi',\
    # 'solution with multicomponent transport')

    ## write the velocity, temperature, density,
    ## and mole fractions to a CSV file
    #f.write_csv('h2_adiabatic.csv', quiet=False)
    flamespeed = f.u[0]
    return(flamespeed)

onedfs(0.5,293,101325)
```

Function: FINALdataANISOTROPY.py
Author: J. Tober (2019)
Description: Function to import and sort solution data from Fluent. Based on a chosen x-location (streamwise) in the channel geometry, the function interpolates the solution data and outputs data on the cross-sectional profile. This code is meant for RSM solution data with anisotropic turbulence. Note that you can also interpolate the solution data to cross-sectional profile using the built in profile exporter in Fluent. The built in profile exporter was used in the final version of the BLF+CFD code.

```
#def CFDDATA(T_input,phi_input,data_number):
def CFDDATA(T_input,U_inlet):
#    T_input = 293
```

```python
#     phi_input = 0.35
#     data_number = '0020'

#     data_name = str(T_input)+'-'+str(int(100*phi_input))+'-0-'\
#     +str(data_number)

#     f=open('C:/Users/tober/Desktop/EPT/H2 Flashback/CFD COUPLING/EXTRA-RSM/\
#     '+str(T_input)+'K/'+data_name,'r+')
    f=open('C:/Users/O.H. Bjornsson/CFD Laptop/19-1-24 Test results/T'\
          +str(T_input)+'K/'+str(U_inlet)+'ms','r+')
    lines_data=f.readlines()


### Creating lists
    Lines = []
    NODE = []
    X = []
    Y = []
    PABS = []
    U = []
    K = []
    UURS = []
    VVRS = []
    WWRS = []
    UVRS = []
    EPS = []
#     S = []
    DUDX = []
    DVDX = []
    DUDY =[]
    DVDY = []
    DPDX = []
    DPDY = []

    for i in range(1,len(lines_data),1):
        Lines.append(lines_data[i].split(' '))
        Lines[i-1] = list(filter(None, Lines[i-1]))
        DPDY.append(float(Lines[i-1][-2]))
        DPDX.append(float(Lines[i-1][-3]))
        DVDY.append(float(Lines[i-1][-4]))
        DUDY.append(float(Lines[i-1][-5]))
        DVDX.append(float(Lines[i-1][-6]))
        DUDX.append(float(Lines[i-1][-7]))
#         S.append(float(Lines[i-1][-8]))
        EPS.append(float(Lines[i-1][-11]))
        UVRS.append(float(Lines[i-1][-12]))
        WWRS.append(float(Lines[i-1][-13]))
        VVRS.append(float(Lines[i-1][-14]))
        UURS.append(float(Lines[i-1][-15]))
        K.append(float(Lines[i-1][-16]))
        U.append(float(Lines[i-1][-19]))
        PABS.append(float(Lines[i-1][-21]))
        Y.append(float(Lines[i-1][-23]))
        X.append(float(Lines[i-1][-24]))
        NODE.append(float(Lines[i-1][-25]))


### Required data on list of points
    x_loc = 0.2
    number_of_points = 50
    XREQ = [x_loc] * number_of_points
    YREQ = list(range(5,number_of_points+5,1))
#    YREQ[:] = [j * 6.67*10**-5 for j in YREQ]
    YREQ[:] = [j * 1*10**-5 for j in YREQ]
    PABSREQ = []
    KREQ = []
    UUREQ = []
    VVREQ = []
    WWREQ = []
    UVREQ = []
    EPSREQ = []
    DUDXREQ = []
    DVDXREQ = []
    DUDYREQ =[]
    DVDYREQ = []
    DPDXREQ = []
    DPDYREQ = []

    import numpy as np
    for k in range(0,len(XREQ),1):
        D = []
        for l in range(0,len(NODE),1):
            d = ( (YREQ[k]-Y[l])**2 + (XREQ[k]-X[l])**2 )**0.5
            D.append(d)
        ZIP = list(zip(NODE,D))
        ZIPSORT = list(sorted(ZIP,key=lambda x: x[1]))
        X1 = X[int(ZIPSORT[0][0])-1]
        X2 = X[int(ZIPSORT[1][0])-1]
        Y1 = Y[int(ZIPSORT[0][0])-1]
        Y2 = Y[int(ZIPSORT[1][0])-1]
        VECTOR2R = np.array([XREQ[k]-X2,YREQ[k]-Y2])
        VECTOR21 = np.array([X1-X2,Y1-Y2])
        PART1 = np.inner(VECTOR2R, VECTOR21) / (np.linalg.norm(VECTOR21))**2
        PART2 = 1 - np.inner(VECTOR2R, VECTOR21) / \
        (np.linalg.norm(VECTOR21))**2
        KREQ.append(K[int(ZIPSORT[0][0])-1]*PART1 + \
                    K[int(ZIPSORT[1][0])-1]*PART2)
        PABSREQ.append(PABS[int(ZIPSORT[0][0])-1]*PART1 + \
                       PABS[int(ZIPSORT[1][0])-1]*PART2)
        UUREQ.append(UURS[int(ZIPSORT[0][0])-1]*PART1 + \
                     UURS[int(ZIPSORT[1][0])-1]*PART2)
        VVREQ.append(VVRS[int(ZIPSORT[0][0])-1]*PART1 + \
                     VVRS[int(ZIPSORT[1][0])-1]*PART2)
        WWREQ.append(WWRS[int(ZIPSORT[0][0])-1]*PART1 + \
                     WWRS[int(ZIPSORT[1][0])-1]*PART2)
        UVREQ.append(UVRS[int(ZIPSORT[0][0])-1]*PART1 + \
                     UVRS[int(ZIPSORT[1][0])-1]*PART2)
        EPSREQ.append(EPS[int(ZIPSORT[0][0])-1]*PART1 + \
                      EPS[int(ZIPSORT[1][0])-1]*PART2)
        DUDXREQ.append(DUDX[int(ZIPSORT[0][0])-1]*PART1 + \
                       DUDX[int(ZIPSORT[1][0])-1]*PART2)
        DVDXREQ.append(DVDX[int(ZIPSORT[0][0])-1]*PART1 + \
```

```
                                DVDX[int(ZIPSORT[1][0])-1]*PART2)
            DUDYREQ.append(DUDY[int(ZIPSORT[0][0])-1]*PART1 + \
                                DUDY[int(ZIPSORT[1][0])-1]*PART2)
            DVDYREQ.append(DVDY[int(ZIPSORT[0][0])-1]*PART1 + \
                                DVDY[int(ZIPSORT[1][0])-1]*PART2)
            DPDXREQ.append(DPDX[int(ZIPSORT[0][0])-1]*PART1 + \
                                DPDX[int(ZIPSORT[1][0])-1]*PART2)
            DPDYREQ.append(DPDY[int(ZIPSORT[0][0])-1]*PART1 + \
                                DPDY[int(ZIPSORT[1][0])-1]*PART2)


### Location of x_m, in front of location of interest,
###  also p_m and u_max centerline
    ZIP3 = list(zip(NODE,X,PABS))
    ZIPSORT3 = list(sorted(ZIP3,key=lambda x: x[1]))
    iii = 0
    while float(ZIPSORT3[iii][1]) - x_loc <0:
        iii = iii+1

    PABS_min = 100000000
    for jjj in range(0,iii,1):
        if ZIPSORT3[jjj][2] < PABS_min:
            PABS_min = ZIPSORT3[jjj][2]
            JJJ = jjj
    x_min = ZIPSORT3[JJJ][1]


### Centerline
    Xcenterline = [x_min] * 500
    YY = list(range(1,501,1))
    YY[:] = [j * 0.0175/500. for j in YY]
    U_centerline = 0

    for l in range(0,len(Xcenterline),1):
        D = []
        for k in range(0,len(NODE),1):
            d = ( (YY[l]-Y[k])**2 + (Xcenterline[l]-X[k])**2 )**0.5
            D.append(d)
        ZIP2 = list(zip(NODE,D))
        ZIPSORT2 = list(sorted(ZIP2,key=lambda x: x[1]))
        if (U[int(ZIPSORT2[0][0])-1]) > U_centerline:
            U_centerline = U[int(ZIPSORT2[0][0])-1]

    return(YREQ,KREQ,UUREQ,VVREQ,WWREQ,UVREQ,EPSREQ,PABSREQ,\
           DUDXREQ,DVDXREQ,DUDYREQ,DVDYREQ,DPDXREQ,DPDYREQ,\
           U_centerline,PABS_min,x_min,x_loc)
#     return(KREQ,UUREQ,VVREQ,WWREQ,UVREQ,EPSREQ,PABSREQ,DUDXREQ,\
#           DVDXREQ,DUDYREQ,DVDYREQ,DPDXREQ,DPDYREQ,x_loc)
```

A.2.2. BLF model with the generalized Stratford criterion

Code: BLFmodel generalizedcriterion.py

Author: O.H. Bjornsson (2019)

Description: BLF model including the generalized Stratford criterion from section 4.2.

```
import numpy as np
from scipy.optimize import fsolve # To solve non-linear equations
import cantera as ct # To get mixture properties
import LFS as LFS # Function to calculate laminar flame speed from polynomial
import matplotlib.pyplot as plt
from onedfs import onedfs # Function to simulate 1-d laminar flame speed

no = 24 # no of flashback points required from model for each case
def BLFmodel(C,PRINT,no): # Function to solve the non-linear BLF model

### Create empty lists
    T_ad_list = []; phi_list = []; S_l0_list = []; S_ls_list = []; Lm_list = []
    Le_list = []; LE_list = []; RHS_list = []; strat_list = []
    kappa_list = []; U_FB_bar_list = []; u_fluc_list = []
    kappa_ratio_list = []; Re_list = []; Ka_list = []; TI_x_list = []
    TI_y_list = []; TI_loc_list = []; flame_x_list = []; flame_y_list = []
    Local_Error_List = []; dpdxmax_list = []; cp_list = []
    u_fluc_tau_list = []; y_plus_list = []


### Starting the loops (mm for different cases, m for different phi)
    for mm in range(1,5,1):
        phi_varied = np.linspace(0.2,1,no)
        for stak in phi_varied:

            phi = stak
            if mm==1:
                T_u = 293
                GEOMETRY = 1
            if mm==2:
                T_u = 473
                GEOMETRY = 1
            if mm==3:
                T_u = 673
                GEOMETRY = 1
            if mm==4:
                T_u = 293
                GEOMETRY = 2

            p_u = 101325
            R = 8.314
            Ea = 125604. # Activation energy (mean value from literature)
            Le_O2 = 2.32
            Le_H2 = 0.33
            gamma2 = 1.

            h = 0.0175 # Height of the channel
            w = 0.157   # Width of the channel
            if GEOMETRY==1: # Channel

                C = 1.05                    # Used to override the C
```

```
                                           # taken as input to the function

                D_h = 4*(h*w)/2/(h+w)    # Hydraulic diameter
                #D_h = ((w*h)/3.14)**0.5 * 2# Incorrect hydraulic diameter
                                           # used in the previous iteration
                                           # of the BLF model
            if GEOMETRY==2: # Tube
                C = 1.05                   # Used to override the C
                D_h = 0.04                 # taken as input to the function

### An equilibrium reaction (with Cantera) results in burned properties
            # Uncomment to use gri30 reaction mechanism:
            #gas1 = ct.Solution('gri30.xml')

            # Uncomment to use O Conaire reaction mechanism:
            H2_RM_path = ('C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
            'H2 Flashback/OConaire Reaction Mechanism/chem.cti')
            gas1= ct.Solution(H2_RM_path)

            # Set up mixture at the appropriate equivalence ratio
            mix = 'H2:'+str(2*phi)+', O2:1, N2:3.76'
            gas1.transport_model = 'Multi'              # 'Multi' or 'Mix'
            gas1.X = mix
            gas1.TP = T_u,p_u
            rho_u = gas1.TD[1]
            cp_u = gas1.cp_mass
            lambda_u = gas1.thermal_conductivity
            LE = lambda_u / (np.dot((gas1.X),(gas1.mix_diff_coeffs_mole))\
                             *rho_u*cp_u)
            mu_u = gas1.viscosity
            thermal_diff_u = lambda_u / (rho_u * cp_u)
            gas1.equilibrate('HP', solver='gibbs')
            cp_b = gas1.cp_mass
            lambda_b = gas1.thermal_conductivity
            T_ad = gas1.T
            rho_b = gas1.TD[1]
            thermal_diff_b = lambda_b / (rho_b * cp_b)

            S_l0 = LFS.inter(phi,T_u,p_u*10**(-5)) # Uncomment for polynomial
            #S_l0 = onedfs(phi,T_u,p_u) # Uncomment to use 1-d flame simulation

            sigma = rho_u / rho_b
            gamma1 = sigma
            deltaf = 2*lambda_u / (rho_u * cp_u * S_l0)
            beta = Ea*(T_ad - T_u)/(R*T_ad**2)
            A = 1+ beta*(1/phi-1)
            Le = 1+ (Le_O2 - 1 + A*(Le_H2-1))/(1+A)
            alfa = gamma1 + 0.5*beta*(Le-1)*gamma2
            Lm = deltaf*(alfa-(sigma-1)*(gamma1/sigma)) # Markstein length
            l_t = 0.07 * D_h
            s = np.log10(l_t/deltaf)

            error = 10
            count = 0
            U_FB = 1

            while abs(error)>0.01:
                def equation1(eq1):
                    u_tau = eq1
                    if GEOMETRY ==1:
                        return ((U_FB-2.4*u_tau)/u_tau -\
                                (1/0.41 * np.log((h/2.*u_tau)\
                                                 /(mu_u/rho_u)) + 5 - 1/0.41))
                    if GEOMETRY ==2:
                        return (u_tau**2 - (0.03955 * (U_FB-2.4*u_tau)**(7/4.)\
                                            *(mu_u/rho_u)**(1/4.)\
                                            *(D_h)**(-1/4.)))
                u_tau = fsolve(equation1,0.1,xtol=1.49012e-2)
                S_t = 0
                imax = 50
                for i in range(5,imax,1):
                    y = i * (mu_u/(rho_u*u_tau))
                    u_fluc = u_tau*(2.661 - 7.211*np.log(u_tau*y*rho_u/mu_u)\
                                    + 7.600*np.log(u_tau*y*rho_u/mu_u)**2\
                                    - 2.900*np.log(u_tau*y*rho_u/mu_u)**3\
                                    + 0.472*np.log(u_tau*y*rho_u/mu_u)**4\
                                    - 0.028*np.log(u_tau*y*rho_u/mu_u)**5)

# MODIFICATION: Jainski's fluctuation
#                    u_fluc = u_tau*(-0.000000015613432*(u_tau*y*\
#                    rho_u/mu_u)**6 + 0.000002729809731*(u_tau*y*\
#                    rho_u/mu_u)**5
#                    - 0.000172250265584*(u_tau*\
#                    y*rho_u/mu_u)**4 + 0.0046443496*(u_tau*y*rho_u/mu_u)**3
#                    - 0.0485448388*(u_tau*y*rho_u/mu_u)**2\
#                    + 0.1590653579*(u_tau*y*rho_u/mu_u) + 1.9742528316)

                    Gamma = 10**(-1/(s+0.4)*np.exp(-(s+0.4))+(1-np.exp\
                                 (-(s+0.4)))*(2/3.*(1-1/2.*np.exp(-(u_fluc/\
                                 S_l0)**(1/3.)))*s-0.11 ))

# MODIFICATION: Anistropic flame stretch
                    v_fluc = u_tau*(-0.00052*(u_tau*y*rho_u/mu_u)**2 + \
                                    0.045873*(u_tau*y*rho_u/mu_u) - 0.014410)
                    w_fluc = u_tau*1.4*(-0.00052*(u_tau*y*rho_u/mu_u)**2\
                                        + 0.045873*(u_tau*y*rho_u/mu_u)\
                                        - 0.014410)
                    k = 1/2. * (u_fluc**2 + v_fluc**2 + w_fluc**2)
#                   kappa_mean = (-0.1*np.log10(U_FB*D_h*rho_u/mu_u)+0.91)\
#                   *u_fluc*v_fluc/(2*k) * (1/7.*U_FB*(y/(h/2.))\
#                   **(-6/7.)) * 2./h
                    kappa_turb = Gamma*u_fluc*v_fluc*w_fluc/(l_t*k)
                    kappa_mean = 0.
#                   kappa_turb = 2/3. * Gamma*u_fluc/l_t
                    kappa_s = 1/2.*u_fluc/l_t
                    kappa = (kappa_mean + kappa_turb + kappa_s)

# Calculate stretched laminar flame speed based on
# Markstein length Lm and flame stretch rate kappa
                    S_ls = S_l0 - kappa*Lm

## Apply turbulence flame speed closures
```

```
                    # With flame stretch
#                    S_t_new = (S_ls*(1+C*(u_fluc/S_ls)**0.5))
                    # Without flame stretch
                    S_t_new = (S_l0*(1+C*(u_fluc/S_l0)**0.5))

# MODIFICATION: Lin's correlation
#                    S_t_new = 10.5 * S_l0 * (Le**(-0.82)) * (u_fluc/S_l0)\
#                    **0.45 * (l_t/deltaf)**(-0.41) * (100000/p_u)**0.75\
#                    * (293/T_u)**(-1.33) #This is what Joeri wrote
#                    S_t_new = 10.5 * S_l0 * (Le**(-0.82)) * (u_fluc/S_l0)\
#                    **0.45 * (l_t/deltaf)**(-0.41) * (p_u/100000)**0.75\
#                    * (T_u/298)**(-1.33) #This is correct according to Lin
#
# Lin's correlation from Sachin where C has to be optimized to new stratford condition
#                    S_t_new = C * S_ls * (u_fluc/S_ls)**0.45 * (p_u/100000)**0.84\
#                    * (T_u/298)**0.4

# MODIFICATION: Lewis correction for elevated temperatures only
#                    if T_u > 300:
#                        if Le < 1.0 and Le >=0.50:
#                            S_t_new = (0.6052*(1/Le)**2 - 1.1314*(1/Le) + 1.5224) * S_t_new    # only valid down to phi = 0.5
#                        if Le< 0.50:
#                            S_t_new =  S_t_new * 1.678

# MODIFICATION: Lewis correction for all temperatures
                    if T_u > 200:
                        if Le < 1.0 and Le >=0.50:
                            S_t_new = (0.6052*(1/Le)**2 - 1.1314*(1/Le) +\
                            1.5224) * S_t_new # only valid down to phi = 0.5
                        if Le < 0.50:
                            S_t_new =  S_t_new * 1.678

                    if S_t_new > S_t:
                        S_t = S_t_new
                        Y = u_tau * y *rho_u / mu_u
                        kappa_FB, Gamma_FB, S_ls_FB = kappa,Gamma,S_ls
                        u_fluc_FB = u_fluc
                        S_l0_FB = S_l0
                        S_ls_FB = S_ls
                        Ka = (u_fluc / S_l0)**(3/2.) * (l_t/deltaf)**(-1/2.)
                        u_TAU = u_tau

#                    if mm==1 and m==7 and count==0:
#                        u_fluc_tau_list.append(u_fluc/u_tau)
#                        y_plus_list.append(y * u_tau * rho_u / mu_u)


### To get dp over flame front, use mass and momentum balance
                dp_max = rho_u * S_t**2 * (sigma - 1)
                dpdx = 0.

                # Hoferichter's simplified Stratford's
                # turbulent BL separation criterion:
#                U_FB_new = (((dp_max + dpdx*0.01)*(2*dp_max+dpdx*0.01)**0.5\
#                             / (0.39))**(2/3.) * 2/rho_u)**0.5

                # Generalized Stratford's criterion:
                if mm==4:   # Tube
                    n = 8   # Pick n for 1/n-th power law
                    de = 0.04/2 # Pick boundary layer thickness delta
                                # 2.120016677e-04 is what Hoferichter uses,
                                # which is 0.0175/82.54652044
                                # Logical choice is channel/pipe halfwidth

                else:       # Channel
                    n = 7
                    de = 0.0175/2

                # Calculate right hand side of the generalized criterion
                RHS =((3*(0.41*0.73)**4)/((n+1)*n**2))**0.25*(1-(3/(n+1)))\
                **(0.25*(n-2))
                # Calculate the centerline velocity U_0
                U_FB_new = (((dp_max/rho_u/0.01**2)**(n/4)*0.01**(0.5*(n-1))*2\
                             **(0.5*(0.5*n+1))*de**0.5)/RHS)**(2/n)

                # Define error for iteration of the non-linear system
                error = U_FB_new - U_FB
                U_FB = U_FB_new
                count = count+1

                if Ka > 1:
                    flame = 'Thin reaction zone'
                elif Ka < 1:
                    flame = 'Corrugated flamelets'
                elif Ka > 100:
                    flame = 'Distributed Reactions/Well-stirred Reactor'

            dpdxmax = dp_max/0.01

            # Calculate bulk velocity at flashback from the centerline velocity
            U_FB_bar = U_FB[0] - 2.4*u_TAU[0]

            if PRINT == 1:
                print ('T_u = ',T_u,'K    ','phi = ',round(phi,2)\
                       ,'    ','U_FB_bar = ',round(U_FB_bar,1),'m/s    ',
                       'u_fluc_FB/S_Ls_FB = ',round(u_fluc_FB[0]\
                       /S_ls_FB[0],1),'    ','l_t/deltaf = '\
                                                     ,round(l_t/deltaf,1),
                       '    ','y+_FB = ',round(Y[0],0),'    ','Re_h = ',\
                       round(U_FB_bar*h*rho_u/mu_u,0),'    ',flame)
                print ((Le**(-0.82)),(u_fluc/S_l0)**0.45,(l_t/deltaf)**(-0.41))
                print(' ')

### Fill lists -----------------------------------------------------------------
            Re_list.append(round(U_FB_bar*h*rho_u/mu_u,0))
            Ka_list.append(float(Ka))
            TI_x_list.append(float(((mu_u/rho_u)**3*l_t/(u_fluc_FB)**3)\
                                **(1/4.)/deltaf))
            TI_y_list.append(float(u_fluc_FB/(2*S_l0)))
            TI_loc_list.append(round(u_fluc_FB[0]/S_ls_FB[0],1))
            flame_x_list.append(l_t/deltaf)
            flame_y_list.append(u_fluc_FB/S_l0_FB)
```

```
            phi_list.append(phi)
            T_ad_list.append(T_ad)
            S_l0_list.append(S_l0)
            S_ls_list.append(S_ls)
            Lm_list.append(1000*Lm)
            Le_list.append(Le)
            LE_list.append(LE)
            kappa_list.append(kappa_FB/1000)
            U_FB_bar_list.append(U_FB_bar)
            u_fluc_list.append(u_fluc_FB)
            kappa_ratio_list.append(100 * 2/3.*Gamma_FB/(2/3.*Gamma_FB+1/2.))
            dpdxmax_list.append(dpdxmax)
            cp_list.append(2*dp_max/rho_u/U_FB**2)

### Define error to get best value for C, and returning outputs ---------------
            if mm==1:
                U_FB_Data = 50.8502675059 * phi - 7.2760562413
                Local_Error_List.append(abs(U_FB_bar-U_FB_Data)/35.28)
            if mm==2:
                U_FB_Data = 78.8293368674 * phi - 6.6335530372
                Local_Error_List.append(abs(U_FB_bar-U_FB_Data)/49.12)
            if mm==3:
                U_FB_Data = 48.0853332972*np.log(phi) + 93.2665966307
                Local_Error_List.append(abs(U_FB_bar-U_FB_Data)/84.69)
            if mm==4:
                U_FB_Data = 48.1158091256*  phi - 6.6807553719
                Local_Error_List.append(abs(U_FB_bar-U_FB_Data)/32.42)

    Error_total = sum(Local_Error_List)
    print('C = ',round(C,4),'Error = ',round(Error_total,4))
    return(D_h,Error_total,U_FB_bar_list,phi_list,dpdxmax_list,cp_list,\
           n,strat_list,RHS_list)

#-----------------------------------------------------------------------------
# -----------------Two ways of running the model------------------------------
# -Note that C can be overriden inside the function BLFmodel (lines 50 and 58)-
# ----------------------------------------------------------------------------

### Run the BLFmodel multiple times until best fitting C is found:
#Error_0 = 10000.
#delta_Error = 1.
#C = 0.79
#dc = 0.01
#while delta_Error > 0:
#    C = C + dc
#    Error_1,U_FB_bar_list,phi_list,dpdxmax_list,cp_list,n,strat_list,\
#    RHS_list = BLFmodel(C,0)
#    delta_Error = Error_0 - Error_1
#    Error_0 = Error_1
#C_optimum = round(C - dc,4)
#print(' ')
#print('Optimum C = ',C_optimum)
#print(' ')
#print('dpdxmaxlist = ', dpdxmax_list)
#Error_model,U_FB_bar_list,phi_list,dpdxmax_list,cp_list,n,strat_list,\
#RHS_list = BLFmodel(C_optimum,1,no)

### Run the BLFmodel once by choosing C
D_h,Error_model,U_FB_bar_list,phi_list,dpdxmax_list,cp_list,n,strat_list,\
RHS_list = BLFmodel(2,1,no)
###---------------------------------------------------------------------------
###---------------------------------------------------------------------------
###---------------------------------------------------------------------------

#%% Default values (from Hoferichter) and experimental results (Eichler) ------
Default_Model_phi = [0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9]

Default_Model_UFB = [ 8.45698472, 10.31915582, 12.63297894, 15.51082699,\
                     18.61602674,
       21.92871719, 25.17090436, 28.52621328, 31.52168034, 34.14735101,
       36.21113684, 38.31325107,  7.85085045, 12.59570915, 18.22011575,
       24.2874666 , 30.4117381 , 36.43550811, 42.11137157, 47.28936131,
       52.03341047, 56.27141839, 59.7308201 , 62.86691147, 23.53936365,
       32.25597853, 41.01390014, 49.42428701, 57.4092592 , 64.87625382,
       71.67542777, 77.94897827, 83.54016192, 88.4518415 , 92.47539058,
       96.14188181,  8.74907499, 10.36201519, 12.51505931, 15.23352964,
       18.17568597, 21.31713644, 24.38743799, 27.56425899, 30.39262171,
       32.86334311, 34.79180371, 36.75913529]

phi_data = []
U_FB_bar_data = []
file_name='C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/H2 Flashback'\
'/Codes Python/EichlerUFB3'
f=open(file_name+".txt",'r+')
lines_data=f.readlines()
for data in range(len(lines_data)):
    lines_split_data = lines_data[data].split('\t')
    phi_data.append(float(lines_split_data[0]))
    U_FB_bar_data.append(float(lines_split_data[1]))

### Results of the original Hoferichter model with Tober's modifications (2019)
Tober_phi = []
Tober_UFBBAR = []
datafile_path = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
'H2 Flashback/Codes Python/Eichler 0 CFD new stratford/'\
'JoeriModifiedResults_Channel_T293K.txt'
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
'H2 Flashback/Codes Python/Eichler 0 CFD new stratford/'\
'JoeriModifiedResults_Channel_T473K.txt'
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
```

```
'H2 Flashback/Codes Python/Eichler 0 CFD new stratford/'\
'JoeriModifiedResults_Channel_T673K.txt'
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
'H2 Flashback/Codes Python/Eichler 0 CFD new stratford/'\
'JoeriModifiedResults_Tube_T293K.txt'
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])


#%% Plotting --------------------------------------------------------------------
fig1, axs1 = plt.subplots(nrows=2, ncols=2, figsize=(12, 12))

C1 = 'b'

axs1[0, 0].plot(phi_data[67:67+39],U_FB_bar_data[67:67+39],\
    marker='+',c='k',linestyle='none',label='Experiments')
axs1[0, 0].plot(Default_Model_phi,Default_Model_UFB[0:12],\
    label='BLF model',c='k',marker='o',linestyle='none')
axs1[0, 0].plot(Tober_phi[0:12],Tober_UFBBAR[0:12],\
    label='Improved BLF model',c='r',marker='o',\
    linestyle='none',fillstyle='none')
axs1[0, 0].plot(phi_list[0:no],U_FB_bar_list[0:no],\
    label='Improved BLF model w) new criterion',\
    c='b',marker='s',markersize=8,linestyle=':',fillstyle='none')
axs1[0, 0].set_xlabel('$\phi$ (-)', fontsize=14)
axs1[0, 0].set_ylabel('$\overline{U}_{FB}$ (m/s)', fontsize=14)
axs1[0, 0].set_title('Channel, T=293 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,120]
NN = 4
axs1[0, 0].set_xlim(XLIM[0],XLIM[1])
axs1[0, 0].set_ylim(YLIM[0],YLIM[1])
axs1[0, 0].set_xticks=(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/\
    NN,step=(XLIM[1]-XLIM[0])/NN))
axs1[0, 0].set_yticks=(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/\
    NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
axs1[0, 0].grid(True)
axs1[0, 0].legend(loc=2,fontsize=12)

axs1[0, 1].plot(phi_list[0:no],U_FB_bar_list[no:2*no],c='b',\
    marker='s',markersize=8,linestyle=':',fillstyle='none')
axs1[0, 1].plot(phi_data[67+39:75+39],U_FB_bar_data[67+39:75+39],\
    marker='+',c='k',linestyle='none')
axs1[0, 1].plot(Default_Model_phi,Default_Model_UFB[12:24],c='k',\
    marker='o',linestyle='none')
axs1[0, 1].plot(Tober_phi[0:12],Tober_UFBBAR[12:24],c='r',\
    marker='o',linestyle='none',fillstyle='none')
axs1[0, 1].set_xlabel('$\phi$ (-)', fontsize=14)
axs1[0, 1].set_title('Channel, T=473 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,120]
NN = 4
axs1[0, 1].set_xlim(XLIM[0],XLIM[1])
axs1[0, 1].set_ylim(YLIM[0],YLIM[1])
axs1[0, 1].set_xticks=(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])\
    /NN,step=(XLIM[1]-XLIM[0])/NN))
axs1[0, 1].set_yticks=[]
axs1[0, 1].grid(True)

axs1[1, 0].plot(phi_list[0:no],U_FB_bar_list[2*no:3*no],\
    label='Improved BLF model w) new criterion',c='b',marker='s',\
    markersize=8,linestyle=':',fillstyle='none')
axs1[1, 0].plot(phi_data[75+39:86+39],U_FB_bar_data[75+39:86+39],\
    marker='+',c='k',linestyle='none',label='Experiments')
axs1[1, 0].plot(Default_Model_phi,Default_Model_UFB[24:36],\
    label='BLF model',c='k',marker='o',linestyle='none')
axs1[1, 0].plot(Tober_phi[0:12],Tober_UFBBAR[24:36],\
    label='Improved BLF model',c='r',marker='o',\
    linestyle='none',fillstyle='none')
axs1[1, 0].set_xlabel('$\phi$ (-)', fontsize=14)
axs1[1, 0].set_ylabel('$\overline{U}_{FB}$ (m/s)', fontsize=14)
axs1[1, 0].set_title('Channel, T=673 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,120]
NN = 4
axs1[1, 0].set_xlim(XLIM[0],XLIM[1])
axs1[1, 0].set_ylim(YLIM[0],YLIM[1])
axs1[1, 0].set_xticks=(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/\
    NN,step=(XLIM[1]-XLIM[0])/NN))
axs1[1, 0].set_yticks=[]
axs1[1, 0].grid(True)

axs1[1, 1].plot(phi_list[3*no:4*no],U_FB_bar_list[3*no:4*no],c='b',marker='s',\
    markersize=8,linestyle=':',fillstyle='none')
axs1[1, 1].plot(phi_data[0:67],U_FB_bar_data[0:67],marker='+',c='k',\
    linestyle='none')
axs1[1, 1].plot(Default_Model_phi,Default_Model_UFB[36:48],c='k',marker='o',\
    linestyle='none')
axs1[1, 1].plot(Tober_phi[0:12],Tober_UFBBAR[36:],c='r',marker='o',\
    linestyle='none',fillstyle='none')
axs1[1, 1].set_xlabel('$\phi$ (-)', fontsize=14)
axs1[1, 1].set_title('Tube, T=293 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,120]
NN = 4
axs1[1, 1].set_xlim(XLIM[0],XLIM[1])
axs1[1, 1].set_ylim(YLIM[0],YLIM[1])
axs1[1, 1].set_xticks=(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,\
    step=(XLIM[1]-XLIM[0])/NN))
axs1[1, 1].set_yticks=[]
axs1[1, 1].grid(True)

#%% Save figure as .png ---------------------------------------------------------
```

```
fname = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/H2 Flashback/'\
'Temporary Spyder Figures/blfresults.png'
fig1.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
        orientation='portrait', papertype=None, format=None,
        transparent=False, bbox_inches=None, pad_inches=0.1,
        frameon=None, metadata=None)
```

A.2.3. BLF+CFD model: Channel

Code: blfcfdmodel channel.py
Author: O.H. Bjornsson (2019)
Description: BLF model duplication using CFD inputs instead of empirical expressions for flow information. From section 5.1.

```
# -*- coding: utf-8 -*-
"""
BLF+CFD model, proof of concept code

"""
import numpy as np
import cantera as ct # Thermodynamics, chemical kinetics software
import LFS as LFS # Calculates laminar flame speed from Hoferichter polynomial
import matplotlib.pyplot as plt
import FINALdataANISOTROPY as dataa # Interpolates Fluent solution data (slow)
from scipy.optimize import fsolve

#%% Calculate BLF limits using Tobers's improvements WITH CFD DATA
#   instead of correlations

h = 0.0175; w = 0.157;  # Channel height, width
#D_h = ((w*h)/3.1415)**0.5 *2; # Hydraulic diameter from Tober code
D_h = 4*(w*h)/2/(w+h); # Correct hydraulic diameter

# Constants
R = 8.314; Ea = 125604; Le_O2 = 2.32; Le_H2 = 0.33; gamma1 = 1; gamma2 = 1.

# List of mixture preheat temperatures
T = [293, 473, 673]

# Lists of inlet bulk velocities at respective preheat temperatures
U = [(5,10,15,20,25,31,35,40),(15,20,25,30,35,45,50,55,60),\
     (40,50,60,70,80,90,100)]

#Create lists for results (equivalence ratio, bulk velocity, critical gradient)
phi_separation = [[],[],[]]
u_fb = [[],[],[]]
gc = [[],[],[]]


for i in range(0,len(T),1): # For all temperatures considered
    # Make temporary lists for results
    phi_separation2 = []
    u_fb2 = []
    # Make list of equivalence ratios
    phi_list = np.linspace(0.9, 0.2, 100)
    U_FB_bar_list = []
    U_inlet = U[i]

    for ii in range(0,len(U_inlet),1): # For all inlet bulk velocities
        STRATFORD = [] # Temporary list
        # Fetch interpolated data using CFDDATA from FINALdataANISOTROPY
        YREQ,KREQ,UUREQ,VVREQ,WWREQ,UVREQ,EPSREQ,PABSREQ,DUDXREQ,DVDXREQ,\
        DUDYREQ,DVDYREQ,DPDXREQ,DPDYREQ,U_centerline,PABS_min,x_min,x_loc =\
        dataa.CFDDATA(T[i], U_inlet[ii])
        last_lhs = 0
        countphiloop = 0
        for iii in phi_list: # For all equivalence ratios
            countphiloop = countphiloop + 1
        ### Set up mixture (GRI3.0 Works as well as O Conaire for this purpose)
            gas1 = ct.Solution('gri30.xml')
            mix = 'H2:'+str(2*iii)+', O2:1, N2:3.76'
            gas1.transport_model = 'Multi'
            gas1.X = mix
            gas1.TP = T[i],101325
            rho_u = gas1.TD[1]
            cp_u = gas1.cp_mass
            lambda_u = gas1.thermal_conductivity
            LE = lambda_u / (np.dot((gas1.X),(gas1.mix_diff_coeffs_mole))\
                            *rho_u*cp_u)
            mu_u = gas1.viscosity
            thermal_diff_u = lambda_u / (rho_u * cp_u)
            gas1.equilibrate('HP', solver='gibbs')
            cp_b = gas1.cp_mass
            lambda_b = gas1.thermal_conductivity
            T_ad = gas1.T
            rho_b = gas1.TD[1]
            thermal_diff_b = lambda_b / (rho_b * cp_b)

            Stratford_list = [] # Temporary list

            tauwall = mu_u*DUDYREQ[0]
            nu_u = mu_u/rho_u
            uurs_max = np.nanmax(UUREQ)
            uurs_max_index = np.nanargmax(UUREQ) # Find index of max. u_fluc
            jj = uurs_max_index

            p_u_CFD = PABSREQ[jj]
            u_fluc_CFD = abs(UUREQ[jj])**0.5

            S_l0 = LFS.inter(iii,T[i],p_u_CFD*10**(-5)) # Laminar flame speed
            sigma = rho_u / rho_b
            gamma1 = sigma
            deltaf = 2*lambda_u / (rho_u * cp_u * S_l0)
            beta = Ea*(T_ad - T[i])/(R*T_ad**2)
            A = 1+ beta*(1/iii -1)
```

```
            Le = 1+ (Le_O2 - 1 + A*(Le_H2-1))/(1+A)
            alfa = gamma1 + 0.5*beta*(Le-1)*gamma2
            Lm = deltaf*(alfa-(sigma-1)*(gamma1/sigma)) # Markstein length
            l_t = 0.07 * D_h


            s = np.log10(l_t/deltaf)
            Gamma = 10**(-1/(s+0.4)*np.exp(-(s+0.4))+(1-np.exp(-(s+0.4)))*\
                          (2/3.*(1-1/2.*np.exp(-(u_fluc_CFD/S_l0)\
                                                  **(1/3.)))*s-0.11 ))

#            kappa_mean = -1/(2*KREQ[jj]) * (UUREQ[jj]*DUDXREQ[jj]+UVREQ[jj]*\
#                                (DUDYREQ[jj]+DVDXREQ[jj])+VVREQ[jj]*DVDYREQ[jj])
#            kappa_t = Gamma*EPSREQ[jj]/KREQ[jj]
#            kappa_mean = (-0.1*np.log(velocity_INPUT[rr]*D_h*rho_u/mu_u)+\
#                           0.91)*UUREQ[jj]**0.5*VVREQ[jj]**\
#                           0.5/(2*KREQ[jj]) * DUDYREQ[jj]

            kappa_mean = 0.
            kappa_t = Gamma * (UUREQ[jj]**0.5*VVREQ[jj]**0.5*WWREQ[jj]**0.5)\
            / (l_t * KREQ[jj])
            kappa_s = 1/2.*u_fluc_CFD/l_t
            kappa = (kappa_t + kappa_mean + kappa_s)
            alpha_0 = 1.
            K = 10.
            S_ls = S_l0 - kappa*Lm      # Stretched laminar flame speed
            kappa_crit = alpha_0 * K * S_l0 / deltaf
            C = 0.84 # Tuning constant
            S_t = (S_ls*(1+C*(u_fluc_CFD/S_ls)**0.5)) # Turbulence closure

            # Apply Lewis number correction
            if T[i] > 200:
                    if Le < 1.0 and Le >=0.50:
                        S_t = (0.6052*(1/Le)**2 - 1.1314*(1/Le) + 1.5224) * S_t
                    if Le< 0.50:
                        S_t =  S_t * 1.678

            if kappa > kappa_crit:
#                       print(jj,kappa_crit,kappa)
                Stratford_list.append(0)

            # If flame stretch is within limits:
            else:
                dp_max = rho_u * S_t**2 * (sigma - 1) # dp over flame
                dpdx = DPDXREQ[jj]
                x = 0.01 # Assumed axial extent of backpressure area
                P = dp_max / 0.01**2 * x**2 + p_u_CFD
                dPdx = 2 * dp_max / 0.01**2 * x
                P_min = PABS_min
                CP = (P - P_min) / (0.5 * rho_u * U_centerline**2)
                dCPdx = dPdx / (0.5 * rho_u * U_centerline**2)
                n = 6
                de = 0.0175/2
                rhs = ((3*(0.41*0.73)**4)/((n+1)*n**2))**0.25*\
                (1-(3/(n+1)))**(0.25*(n-2)) # right hand side of gen. criterion
                lhs = CP**(0.25*(n-2))*(de*dCPdx)**(0.5) # left hand side

            if rhs <= last_lhs and lhs < rhs: # if tipping point is reached
                phi_separation[i].append(iii) # equivalence ratio at flashback
                gc[i].append(DUDYREQ[0]) # wall gradient at flashback
                u_fb[i].append(U_inlet[ii]) # bulk inlet velocity at flashback
                yplusloc = YREQ[jj]/(nu_u/(tauwall/rho_u)**0.5)

                print("\ng_w          y+          phi          rhs          lhs"\
                "          sls          sl0          cplim         cp           ")
                print("%.2E     %.2E     %.2f          %.3f         %.3f         %.2f"\
                "          %.2f          %.2f" \
                % (DUDYREQ[0],yplusloc,iii,rhs,lhs,S_ls,S_l0,(n-2)/(n+1),CP))
                break
            last_lhs = lhs

#%% Prepare validation data
### Results of the original model ----------------------------------------------
Default_Model_phi = [0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9]
Default_Model_UFB = [ 6.70798426,  8.82348554, 11.24479479, 14.19490623, \
                     17.3893729 ,
   20.82524271, 24.226564  , 27.77517374, 31.00090666, 33.88480434,
   36.21085847, 38.58727119,  7.09568617, 11.68560967, 17.18901776,
   23.21327254, 29.37823449, 35.52793146, 41.40220583, 46.84775837,
   51.89728874, 56.47327076, 60.28544781, 63.75527976, 22.546382  ,
   31.16507735, 39.94901987, 48.50328199, 56.73712225, 64.53647118,
   71.73508127, 78.45280993, 84.50951192, 89.88303058, 94.34449072,
   98.41211723,  8.12450355,  9.88003665, 12.11533307, 14.9098337 ,
   17.9397085 , 21.19220129, 24.39198498, 27.70699492, 30.68707647,
   33.30501874, 35.36029642, 37.44624207]

### Experimental data ------------------------------------------------------------
g=open('C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/H2 Flashback/Codes '\
       'Python/Eichler 0 CFD new stratford/EichlerUFB3.txt','r+')
lines_expdata=g.readlines()
phi_data = []
U_FB_bar_data = []

for dd in range(len(lines_expdata)):
    lines_split_expdata = lines_expdata[dd].split('\t')
    phi_data.append(float(lines_split_expdata[0]))
    U_FB_bar_data.append(float(lines_split_expdata[1]))
### Critical gradients from Eichler's 0 deg channel, fig. 4.8 in his PhD thesis
g_c_eichler293 = np.array([0.1074,0.1670,0.1670,0.2227,0.2266,0.3022,0.3062,\
                           0.3062,0.3976,0.3936,0.4930,0.4930,0.4891,0.5964,\
                           0.5964,0.5964,0.7157,0.7157,0.7197,0.8350,0.8509,\
                           0.8588,0.9781,0.9940,1.1412,1.1531,1.3121,1.3241,\
                           1.4871,1.5030,1.5149,1.5189,1.7336,1.7376,1.7495,\
                           1.7495,1.7654,1.7813,1.8012,1.8012])*1e+05
x_eichler293 = np.array([0.2913,0.3167  ,  0.3257   , 0.3346   , 0.3511   , \
                         0.3779  ,  0.4012   , 0.4053   , 0.4417    ,0.4445 \
                         ,0.4685 ,0.4761    ,0.4815  ,  0.4987 ,   0.5056   \
                         , 0.5166  ,  0.5330 ,   0.5440   , 0.5578  ,  0.5605\
                         , 0.5674  ,  0.5736, 0.6017 ,   0.6113    ,0.6470   \
                         , 0.6580   , 0.6999   , 0.7130   , 0.7507 ,   0.7590\
                         , 0.7734 ,   0.7851   , 0.8613 ,0.8757  ,  0.8997  ,\
                         0.9073  ,  0.9162  ,  0.9341   , 0.9602  ,  0.9602])
```

```python

g_c_eichler293_4 = np.array([0.831826401446652 ,  1.030741410488245,\
                             3.218806509945748 ,  4.285714285714284])*1e+04
x_eichler293_4 = np.array([0.250047755491882  , 0.250047755491882,\
                           0.290162368672397  , 0.290162368672397])

g_c_eichler293_2 = np.array([0.723327305605786,   0.813743218806509   ,\
                             2.368896925858950 ,  2.820976491862567  ,\
                             3.942133815551536    ,  4.683544303797468  ,\
                             6.148282097649185,   6.889692585895117  ,\
                             8.589511754068715])*1e+04
x_eichler293_2 = np.array([0.285959885386819,   0.302769818529131  ,\
                           0.333333333333333  , 0.363514804202483,\
                           0.363514804202483    ,  0.399808978032474   ,\
                           0.399808978032474  , 0.434574976122254  ,\
                           0.434574976122254])

### Channel confined BL flashback experimental
### data and results from previous models.

### Experimental data from Eichler (2011) ---------------------------------
g=open('C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/H2 Flashback/'\
       'Codes Python/EichlerUFB3.txt','r+')
lines=g.readlines()
Eichler_phi = []
Eichler_UFBBAR = []

for i in range(len(lines)):
    lines_split = lines[i].split('\t')
    Eichler_phi.append(float(lines_split[0]))
    Eichler_UFBBAR.append(float(lines_split[1]))

### Results of the original Hoferichter model (2017) ----------------------
Hoferichter_phi = [0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9]
Hoferichter_UFBBAR = [ 6.70798426,  8.82348554, 11.24479479, 14.19490623,\
                      17.3893729  ,
   20.82524271, 24.226564   , 27.77517374, 31.00090666, 33.88480434,
   36.21085847, 38.58727119,  7.09568617, 11.68560967, 17.18901776,
   23.21327254, 29.37823449, 35.52793146, 41.40220583, 46.84775837,
   51.89728874, 56.47327076, 60.28544781, 63.75527976, 22.546382   ,
   31.16507735, 39.94901987, 48.50328199, 56.73712225, 64.53647118,
   71.73508127, 78.45280993, 84.50951192, 89.88303058, 94.34449072,
   98.41211723,  8.12450355,  9.88003665, 12.11533307, 14.9098337  ,
   17.9397085  , 21.19220129, 24.39198498, 27.70699492, 30.68707647,
   33.30501874, 35.36029642, 37.44624207]

### Results of the original Hoferichter model with Tober's modifications (2019)
Tober_phi = []
Tober_UFBBAR = []
datafile_path = "C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/"\
"H2 Flashback/Codes Python/Eichler 0 CFD new stratford/"\
"JoeriModifiedResults_Channel_T293K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = "C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/"\
"H2 Flashback/Codes Python/Eichler 0 CFD new stratford/"\
"JoeriModifiedResults_Channel_T473K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = "C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/"\
"H2 Flashback/Codes Python/Eichler 0 CFD new stratford/"\
"JoeriModifiedResults_Channel_T673K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path = "C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/"\
"H2 Flashback/Codes Python/Eichler 0 CFD new stratford/"\
"JoeriModifiedResults_Tube_T293K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

#%% Plot results in terms of g_c  ---------------------------------------
fig, (sp1, sp2) = plt.subplots(nrows=1, ncols=2, figsize=(16, 8))

sp1.plot(x_eichler293,g_c_eichler293,label='0deg channel, Experiments',c='b',\
         marker='s',linestyle='none',fillstyle='full')
sp1.plot(phi_separation[0],gc[0],label='0deg channel, BLF+CFD model',c='b',\
         marker='s',linestyle='none',fillstyle='none')
sp1.plot(x_eichler293_2,g_c_eichler293_2,label='2deg diffuser, Experiments',\
         c='g',marker='d',linestyle='none',fillstyle='full')
sp1.plot(x_eichler293_4,g_c_eichler293_4,label='4deg diffuser, Experiments',\
         c='r',marker='^',linestyle='none',fillstyle='full')

sp1.set_xlabel('$\phi$ [-]', fontsize=14)
sp1.set_ylabel('g [1/s]', fontsize=14)
sp1.set_title('Channel and diffusers, critical gradient',fontsize=14)
XLIM = [0,1]
YLIM = [0,300000]
NN = 4
sp1.set_xlim(XLIM[0],XLIM[1])
sp1.set_ylim(YLIM[0],YLIM[1])
sp1.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
sp1.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
sp1.grid(True)
sp1.legend(loc=2,fontsize=12)

```

```python
#%% Plot results in terms of U_FB_bar at inlet of channel/diffuser -----------
black = 'k'
red = 'r'

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

ax1.plot(Eichler_phi[67:67+39],Eichler_UFBBAR[67:67+39],\
        label='Experiments',marker='+',c='k',linestyle='none')
ax1.plot(Hoferichter_phi[0:12],Hoferichter_UFBBAR[0:12],\
        label='BLF model',c=black,marker='o',linestyle='none',\
        fillstyle='full')
ax1.plot(Tober_phi[0:12],Tober_UFBBAR[0:12],label='Improved BLF model',\
        c='r',marker='o',linestyle='none',fillstyle='none')
ax1.plot(phi_separation[0],u_fb[0],label='BLF+CFD model',\
        c='b',marker='s',markersize=8,linestyle=':',fillstyle='none')

ax1.set_xlabel('$\phi$ (-)', fontsize=14)
ax1.set_ylabel('$\overline{U}_{FB}$ (m/s)', fontsize=14)
ax1.set_title('Channel, T=293 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,160]
NN = 4
ax1.set_xlim(XLIM[0],XLIM[1])
ax1.set_ylim(YLIM[0],YLIM[1])
ax1.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
ax1.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
ax1.grid(True)
ax1.legend(loc=2,fontsize=12)

ax2.plot(Eichler_phi[67+39:75+39],Eichler_UFBBAR[67+39:75+39],\
        label='Experiments',marker='+',c='k',linestyle='none')
ax2.plot(Hoferichter_phi,Hoferichter_UFBBAR[12:24],label='BLF model',c='k',\
        marker='o',linestyle='none')
ax2.plot(Tober_phi[0:12],Tober_UFBBAR[12:24],label='Improved BLF model',c='r',\
        marker='o',linestyle='none',fillstyle='none')
ax2.plot(phi_separation[1],u_fb[1],label='BLF+CFD model',c='b',\
        marker='s',markersize=8,linestyle=':',fillstyle='none')
ax2.set_xlabel('$\phi$ (-)', fontsize=14)
ax2.set_title('Channel, T=473 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,160]
NN = 4
ax2.set_xlim(XLIM[0],XLIM[1])
ax2.set_ylim(YLIM[0],YLIM[1])
ax2.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
ax2.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
ax2.grid(True)
ax2.yaxis.tick_right()

fig, (ax3, ax4) = plt.subplots(nrows=1, ncols=2, figsize=(12, 6))

ax3.plot(Eichler_phi[75+39:86+39],Eichler_UFBBAR[75+39:86+39],\
        label='Experiments',marker='+',c='k',linestyle='none')
ax3.plot(Hoferichter_phi,Hoferichter_UFBBAR[24:36],c='k',\
        label='BLF model',marker='o',linestyle='none')
ax3.plot(Tober_phi[0:12],Tober_UFBBAR[24:36],label='Improved BLF model',\
        c='r',marker='o',linestyle='none',fillstyle='none')
ax3.plot(phi_separation[2],u_fb[2],label='BLF+CFD model',c='b',marker='s',\
        markersize=8,linestyle=':',fillstyle='none')
ax3.set_xlabel('$\phi$ (-)', fontsize=14)
ax3.set_ylabel('$\overline{U}_{FB}$ (m/s)', fontsize=14)
ax3.set_title('Channel, T=673 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,160]
NN = 4
ax3.set_xlim(XLIM[0],XLIM[1])
ax3.set_ylim(YLIM[0],YLIM[1])
ax3.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
ax3.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
ax3.grid(True)

ax4.plot(Eichler_phi[0:67],Eichler_UFBBAR[0:67],marker='+',c='k',\
        linestyle='none')
ax4.plot(Hoferichter_phi,Hoferichter_UFBBAR[36:48],c='k',marker='o',\
        linestyle='none')
ax4.plot(Tober_phi[0:12],Tober_UFBBAR[36:],c='r',marker='o',linestyle='none',\
        fillstyle='none')
ax4.set_xlabel('$\phi$ (-)', fontsize=14)
ax4.set_title('Tube, T=293 K',fontsize=14)
XLIM = [0.2,1]
YLIM = [0,120]
NN = 4
ax4.set_xlim(XLIM[0],XLIM[1])
ax4.set_ylim(YLIM[0],YLIM[1])
ax4.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
ax4.set_yticks=[]
ax4.grid(True)
```

A.2.4. BLF+CFD model: Final code including velocity profile fitting

Code: blfcfdmodel finalversion.py
Author: O.H. Bjornsson (2019)
Description: BLF+CFD model including automatic fitting of the mean velocity profile for extended application to adverse pressure gradient flow. From section 5.2.

```python
# -*- coding: utf-8 -*-
"""
BLF+CFD model, final version with 1/n-th power law fitting
```

```python
@author: O.H. Bjornsson

Based on Hoferichter (DOI: 10.1115/1.4034237) and
Tober (http://resolver.tudelft.nl/uuid:29260da8-c1e9-4ffb-932b-121ce0326752)

Written for confined boundary layer flashback prediction in a
0deg channel and 2deg and 4deg planar asymmetric diffusers.

Input: Flow profiles extracted from Fluent 19.0
"""
import numpy as np
import cantera as ct
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
from _onedfs import onedfs # To calculate laminar flame speed using Cantera
import _LFS as LFS # To calculate laminar flame speed from polynomial
import copy

#%% Specify the path to the BLFCFDmodel folder, include a slash / at the end.
### i.e. '.../BLFCFDmodel/' not '.../BLFCFDmodel'
basepath = 'C:/Users/O.H. Bjornsson/Google Drive/Drive Delft/'\
'H2 Flashback/Codes Python/BLFCFDmodel/'
#%% Tuning parameter C for the effect of
### turbulence fluctuations on turbulent flame speed
C = {}
C[0] = 1.1
C[2] = 1.4
C[4] = 2.8

#%% Tuning parameter raising the dp effect of the flame.
C2 = {}
C2[0] = 1
C2[2] = 1
C2[4] = 1

### First create a dictionary for results for each of phi (equivalence ratio),
### gc (critical gradient), ufbbar (inlet bulk velocity at flashback) and
### ufbbarlocal (local bulk velocity at flashback).

# Create the dictionary for results in terms of phi at flashback.
phi_dict = {}

# The dictionary has an entry for each geometry studied (0, 2 and 4 degree)
# channel/diffusers
phi_dict[0] = {};
phi_dict[2] = {};
phi_dict[4] = {};

# Add the studied x-positions in the respective geometry
# (e.g. x = 1.25m in the channel, x = 0.125, 0.250, 0.375 in the diffusers)
phi_dict[0][1.25] = []
phi_dict[2][0.125] = [];    phi_dict[2][0.250] = [];    phi_dict[2][0.375] = []
phi_dict[4][0.125] = [];    phi_dict[4][0.250] = [];    phi_dict[4][0.375] = []

# Copy the phi dictionary to have a similar dictionary for other results
gc_dict = copy.deepcopy(phi_dict);
ufbbar_dict = copy.deepcopy(phi_dict)
ufbbarlocal_dict = copy.deepcopy(phi_dict)

### Create dictionary with the bulk inlet velocities studied with CFD
### for each geometry studied
ubulk_dict = {}
ubulk_dict[0] = [5,10,15,20,25,30,35,40,45]
ubulk_dict[2] = [10,15,20,25,30,35,40,45,50,55,70]
ubulk_dict[4] = [25,35,50,55,60,65,70,75,80,85]

### Data paths to the CFD profiles exported from Fluent
### For each geometry, there is a Workbench 19.0 case file with the Fluent
### files. The exported profiles are kept there, sorted by inlet bulk velocity.
datapath_dict = {}
datapath_dict[0] = basepath+'0degProfiles/'
datapath_dict[2] = basepath+'2degProfiles/'
datapath_dict[4] = basepath+'4degProfiles/'

#%% Solve the BLF+CFD model for all geometries, x-positions and bulk velocities
for geometry in phi_dict:
    for xposition in phi_dict[geometry]:
        for ubulk in ubulk_dict[geometry]:
            # Create the final datapath to the CFD profile studied
            xpositionstring = '%.3f' % xposition
            turbulencemodel = 'RSM'
            equivalenceratio = 'avgphi'
            ubulkstring = '%.0f' % ubulk
            datapath = datapath_dict[geometry]+turbulencemodel+'_'\
            +equivalenceratio+'_'+ubulkstring+'ms/x='+xpositionstring+'.txt'

            # Open the data path to the CFD profile and sort the data into
            # a dictionary named 'data', with keywords that Fluent gives.
            f = open(datapath, 'r')

            data = {}
            lines = f.readlines()
            f.close()

            count = 0
            for l in lines:
                if l[0] == '(':
                    if count != 0:
                        string = l[1:-1]
                        data[string] = []
                else:
                    if l[0] != ')':
                        data[string].append(float(l))
                count = count + 1

            # Send the data to numpy arrays so it can be worked with.
            x = np.array(data['x'])
            y = np.array(data['y'])
            p = np.array(data['pressure'])
            pabs = np.array(data['absolute-pressure'])
```

```
        rho = np.array(data['density'])
        vel = np.array(data['velocity-magnitude'])
        u = np.array(data['x-velocity'])
        v = np.array(data['y-velocity'])
        cellRe = np.array(data['cell-reynolds-number'])
        k = np.array(data['turb-kinetic-energy'])
        eps = np.array(data['turb-diss-rate'])
        Returb = np.array(data['turb-reynolds-number-rey'])
        mu = np.array(data['viscosity-lam'])
        taux = np.array(data['x-wall-shear'])
        tauy = np.array(data['y-wall-shear'])
        dudx = np.array(data['dx-velocity-dx'])
        dvdx = np.array(data['dy-velocity-dx'])
        dudy = np.array(data['dx-velocity-dy'])
        dvdy = np.array(data['dy-velocity-dy'])
        dpdx = np.array(data['dp-dx'])
        dpdy = np.array(data['dp-dy'])

        # If the data is "upside down" which sometimes happens in Fluent,
        # we want to correct that by flipping it. We want the y-coordinate
        # vector to be increasing from index 0.
        if y[0] > y[-1]:
            x = np.flipud(x)
            y = np.flipud(y)
            p = np.flipud(p)
            pabs = np.flipud(pabs)
            rho = np.flipud(rho)
            vel = np.flipud(vel)
            u = np.flipud(u)
            v = np.flipud(v)
            cellRe = np.flipud(cellRe)
            k = np.flipud(k)
            eps = np.flipud(eps)
            Returb = np.flipud(Returb)
            mu = np.flipud(mu)
            taux = np.flipud(taux)
            tauy = np.flipud(tauy)
            dudx = np.flipud(dudx)
            dvdx = np.flipud(dvdx)
            dudy = np.flipud(dudy)
            dvdy = np.flipud(dvdy)
            dpdx = np.flipud(dpdx)
            dpdy = np.flipud(dpdy)

        # If we use the RSM turbulence model we get anisotropic
        # turbulence. In that case, import the reynolds stresses
        if turbulencemodel == 'RSM':
            uurs = np.array(data['uu-reynolds-stress'])
            vvrs = np.array(data['vv-reynolds-stress'])
            wwrs = np.array(data['ww-reynolds-stress'])
            uvrs = np.array(data['uv-reynolds-stress'])
            if y[0] > y[-1]:
                uurs = np.flipud(uurs)
                vvrs = np.flipud(vvrs)
                wwrs = np.flipud(wwrs)
                uvrs = np.flipud(uvrs)
        if turbulencemodel == 'SKW':
            uurs = 2*(k.copy())

        print('\n\n\nU inlet: %0.3f, X: %0.3f, H: %0.3f\n' \
              % (ubulk,x[0],max(y)))

        # Fit the 1/nth power law to the mean velocity profile with n

        h = max(y)  # Height of duct
        ubulklocal = ubulk*0.0175/h     # Local bulk velocity
        w = 0.157;  # Width of channel
        n_list = np.linspace(3, 9, 100) # List of n's to check
        rho_u = np.average(rho)
        nu_u = np.average(mu)/np.average(rho)
        mu_u = np.average(mu)
        tauwall = taux[0]
        yplus = y*(tauwall/rho_u)**0.5*rho_u/mu_u # Calculate yplus
        utau = (tauwall/rho_u)**0.5 # Calculate friction velocity

        # Find location of maximum turbulence fluctuations in the
        # viscous/transitioning boundary layer
        uurs_max = np.nanmax(uurs[40 > yplus])
        uurs_max_index = np.nanargmax(uurs[40 > yplus])
        jj = uurs_max_index


        # Choose the boundary layer thickness delta in the 1/n-th power law
        # Delta should be somewhere in the outer layer, e.g. between
        # h/10 and h/2.
        de = h/6
        u_0 = u[np.nanargmin(abs(y-de))] # Finds u_0 at location of delta

        #  Choose between two least squares methods:
        method = '2' #   1 for best fit above y+ lowerbound
                     #   2 for best fit between y+ lowerbound-upperbound
        lowerb = 30  #  (y+ = 30 is inner extent of fully turbulent layer)
        upperb = 50  #  (y+ = 50 chosen as some value close to y+ 30)

        if method == '1':
            logic = yplus > lowerb

        if method == '2':
            logic1 = yplus > lowerb
            logic2 = upperb > yplus
            logic = logic1*logic2

        yturb = y[logic] # finds y coordinates in selected region
        u_nlaw = np.zeros((len(n_list),len(yturb)))

        #Y plus vector above 30 (turbulent region)
        yplusturb = yturb*(tauwall/rho_u)**0.5*rho_u/mu_u
        for i in range(0,len(n_list),1):
            u_nlaw[i] = u_0*(yturb/(de))**(1/n_list[i])
        sd = (u_nlaw-u[logic])**2
        minindex = np.argmin(sd.sum(axis=1))

```

```
        # Find best fitting n constant
        # between y+ = lowerbound and upperbound
        n = n_list[minindex]

        # Right hand side of generalized Stratford criterion
        rhs = ((3*(0.41*0.73)**4)/((n+1)*n**2))\
        **0.25*(1-(3/(n+1)))**(0.25*(n-2))

        print("Best fit: n = %0.2f\n" % n)

        plt.rc('xtick',labelsize=16)
        plt.rc('ytick',labelsize=16)

        pic1, mndl = plt.subplots(nrows=1, ncols=2, figsize=(10, 4.5))

        logic1 = yplus > lowerb
        logic2 = de > y
        logic = logic1*logic2

        yfit = y[logic] # finds y coordinates in selected region
        yfitplus = yfit*(tauwall/rho_u)**0.5*rho_u/mu_u

        mndl[0].semilogx(yplus,u/utau,label=\
                    'CFD ('+turbulencemodel+')',\
                    linestyle='--',linewidth=4)
        mndl[0].semilogx(yfitplus,u_0*(yfit/(de))\
                    **(1/n)/utau,\
                    label='1/n-th-law, n = %0.2f'% n,linewidth=4)
        mndl[0].set_xlabel('$y^+\ [-]$',fontsize=20)
        mndl[0].set_ylabel('$u^{+}\ [-]$',fontsize=20)
        mndl[0].grid(True)
        if geometry == 2 or geometry == 4:
            mndl[0].set_title(str(geometry)+\
                'deg duct - $\overline{U}_{inlet}$ = '+\
                str(ubulk)+' m/s - x/L = %0.1f' %\
                (xposition/0.5),fontsize=20)
        if geometry == 0:
            mndl[0].set_title(str(geometry)+\
                'deg duct - $\overline{U}_{inlet}$ = '+str(ubulk)+\
                ' m/s',fontsize=20)
        mndl[1].plot(y,u,label='CFD ('+turbulencemodel+')',\
            linestyle='--',linewidth=4)
        mndl[1].plot(yfit,u_0*(yfit/de)**\
                (1/n),label='1/n-th-law, n = %0.2f'% n,linewidth=4)
        mndl[1].legend(loc=4,fontsize=16)
        mndl[1].set_xlabel('$y\ [m]$',fontsize=20)
        mndl[1].set_ylabel('$u\ [m/s]$',fontsize=20)
        mndl[1].grid(True)

        if geometry == 0:
            bg = 0.005
        if geometry == 4 or geometry == 2:
            bg = 0.01
        sm = bg/5
        mndl[1].xaxis.set_major_locator(ticker.MultipleLocator(bg))
        mndl[1].xaxis.set_minor_locator(ticker.MultipleLocator(sm))

        #%% Save figure as .png ---------------------------------------
        fname = \
        basepath+'Images/Outer layer fits/'+str(geometry)\
        +'/%0.3f' % xposition +'/'+str(ubulk)+'.png'
        plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
        orientation='landscape', papertype=None, format=None,
        transparent=False, bbox_inches=None, pad_inches=0.1,
        frameon=None, metadata=None)

        # Plot other important things as well,
        # like UU Reynolds stress

        if turbulencemodel == 'RSM':
            plt.figure(3)
            plt.plot(y,vvrs,label='vv rs')
            plt.plot(y,wwrs,label='ww rs')
            plt.figure(4)
            plt.semilogx(yplus,vvrs,label='vv rs')
            plt.semilogx(yplus,wwrs,label='ww rs')
        plt.figure(3)
        plt.plot(y,uurs,label='uu rs')
        plt.legend(loc=4,fontsize=16)
        plt.ylabel('$u^2_i\ [m^2/s^2]$',fontsize=18)
        plt.xlabel('$y\ [m]$',fontsize=18)
        plt.figure(4)
        plt.semilogx(yplus,uurs,label='uu rs')
        plt.legend(loc=4,fontsize=16)
        plt.ylabel('$u^2_i\ [m^2/s^2]$',fontsize=18)
        plt.xlabel('$y^+\ [-]$',fontsize=18)
        plt.show()

        # Set up mixture using Cantera and check for flashback at every
        # equivalence ratio in phi_list
        phi_list = np.linspace(1,0.2,100)
        last_lhs = 0
        countphiloop = 0
        for phi in phi_list:
            countphiloop = countphiloop + 1
            T = 293         # In the diffuser cases we only look at room T
            P = 101325      # BLF model is only validated at 1 atm for now

            #gas1 = ct.Solution('gri30.xml') # Can use GRI30 reaction mech.

            # Can also use reaction mechanism of O Conaire:
            # http://www.nuigalway.ie/media/researchcentres/
            # combustionchemistrycentre/files/mechanismdownloads/
            # hydrogen/H2_reaction_v1a.dat
            # http://dx.doi.org/10.1002/kin.20036
            gas1= ct.Solution(\
                basepath+'OConaire Reaction Mechanism/chem.cti')
            # Set up mixture based on equivalence ratio phi
            mix = 'H2:'+str(2*phi)+', O2:1, N2:3.76' # Simplified h2-air
            gas1.transport_model = 'Multi'      # 'Multi' or 'Mix'
            gas1.X = mix
            gas1.TP = T,P
```

```
                rho_u = gas1.TD[1]
                cp_u = gas1.cp_mass
                lambda_u = gas1.thermal_conductivity
                LE = lambda_u / (np.dot((gas1.X),\
                                    (gas1.mix_diff_coeffs_mole))*rho_u*cp_u)
                mu_u = gas1.viscosity
                nu_u = mu_u/rho_u
                thermal_diff_u = lambda_u / (rho_u * cp_u)
                gas1.equilibrate('HP', solver='gibbs')
                cp_b = gas1.cp_mass
                lambda_b = gas1.thermal_conductivity
                T_ad = gas1.T
                rho_b = gas1.TD[1]
                thermal_diff_b = lambda_b / (rho_b * cp_b)

                # Some constants required for the model
                R = 8.314;
                Ea = 125604; # Activation energy, mean value from literature
                Le_O2 = 2.32;
                Le_H2 = 0.33;
                gamma2 = 1;
                D_h = 4*w*h/2/(h+w); # Hydraulic diameter

#                # Find location of maximum turbulence fluctuations in the
#                # viscous/transitioning boundary layer
#                yplus = y/(nu_u/(tauwall/rho_u)**0.5)
#                uurs_max = np.nanmax(uurs[40 > yplus])
#                uurs_max_index = np.nanargmax(uurs[40 > yplus])
#                jj = uurs_max_index

                p_u_CFD = pabs[jj]
                u_fluc_CFD = abs(uurs[jj])**0.5

                # Two methods to calculate laminar flame speed:
                ##  First method is from Tober, calculates the LFS from
                ##  a polynomial. Coefficients are given in Hoferichter's
                ##  PhD thesis (2017):
                S_l0 = LFS.inter(phi,T,p_u_CFD*10**(-5))
                ##  Second method uses Cantera to simulate a 1-d flat flame.
                ##  This method gives slightly different results at room T
                ##  and takes much longer to compute:
#                S_l0 = onedfs(phi,T,p_u_CFD,basepath)

                if turbulencemodel == 'RSM':
                    sigma = rho_u / rho_b
                    gamma1 = sigma
                    deltaf = 2*lambda_u / (rho_u * cp_u * S_l0)
                    beta = Ea*(T_ad - T)/(R*T_ad**2)
                    A = 1+ beta*(1/phi-1)
                    Le = 1+ (Le_O2 - 1 + A*(Le_H2-1))/(1+A)
                    alfa = gamma1 + 0.5*beta*(Le-1)*gamma2
                    Lm = deltaf*(alfa-(sigma-1)*(gamma1/sigma))#MarksteinLength
                    l_t = 0.07 * D_h
                    s = np.log10(l_t/deltaf)
                    Gamma = 10**(-1/(s+0.4)*np.exp(-(s+0.4))+\
                                 (1-np.exp(-(s+0.4)))*(2/3.*(1-1/2.*np.exp\
                                 (-(u_fluc_CFD/S_l0)**(1/3.)))*s-0.11 ))
                    kappa_mean = 0.
                    kappa_t = Gamma * (uurs[jj]**0.5*vvrs[jj]**\
                                       0.5*wwrs[jj]**0.5)/ (l_t * k[jj])

                    kappa_s = 1/2.*u_fluc_CFD/l_t
                    # Calculate flame stretch rate
                    kappa = (kappa_t + kappa_mean + kappa_s)
                    alpha_0 = 1.
                    K = 10.
                    # Calculate stretched laminar flame speed
                    S_ls = S_l0 - kappa*Lm
                    kappa_crit = alpha_0 * K * S_l0 / deltaf


                # Damkohler turbulence closure using stretched lfs
                #S_t = (S_ls*(1+C_0deg*(u_fluc_CFD/S_ls)**0.5))
                # Damkohler closure using unstretched laminar flame speed
                S_t = (S_l0*(1+C[geometry]*(u_fluc_CFD/S_l0)**0.5))

                # Use Lewis number correction
                if T > 200:
                        if Le < 1.0 and Le >= 0.50:
                            S_t = (0.6052*(1/Le)**2 - 1.1314*(1/Le) + 1.5224) \
                                * S_t    # only valid down to phi = 0.5
                        if Le < 0.50:
                            S_t =  S_t * 1.678
                if kappa > kappa_crit:
                    print('critical flame stretch')
                    print(kappa_crit,kappa)
                else:
                    # Calculate pressure jump over flame front
                    #dp_max = rho_u * S_t**2 * (sigma - 1)

                    # Calculate pressure jump over flame front
                    # and include a tuning constant based on the
                    # PDF/CDF of turbulence streamwise velocity fluctuation
                    dp_max = C2[geometry] * rho_u * S_t**2 * (sigma - 1)

                    dpdx_loc = dpdx[jj]
                    # Assume axial extend of backpressure profile
                    x = 0.01
                    # Calculate P, dPdx, coefficient of pressure CP etc.
                    P = dp_max / x**2 * x**2 + p_u_CFD #+ dpdx_loc * x
                    dPdx = 2 * dp_max / x**2 * x #+ dpdx_loc
                    P_min = pabs[jj] # = p_u_CFD
                    CP = (P - P_min) / (0.5 * rho_u * u_0**2)
                    dCPdx = dPdx / (0.5 * rho_u * u_0**2)
                    # Calculate left hand side of generalized Stratford crit.
                    lhs = CP**(0.25*(n-2))*(de*dCPdx)**(0.5)

                    yplusloc = y[jj]/(nu_u/(tauwall/rho_u)**0.5)

                    # If this is the tipping point of the Stratford criterion,
                    # record the equivalence ratio phi, critical gradient,
                    # and the inlet and local bulk velocities.
```

```python
                    if rhs <= last_lhs and lhs < rhs: # and kappa < kappa_crit:
                        phi_dict[geometry][xposition].append(phi)
                        gc_dict[geometry][xposition].append(dudy[0])
                        ufbbar_dict[geometry][xposition].append(ubulk)
                        ufbbarlocal_dict[geometry][xposition].\
                        append(ubulklocal)
                        print("\ng_w          y+            phi           rhs"\
                              "          lhs           cplim        cp            ")
                        print("%.2E     %.2E     %.2f          %.3f          %.3f"\
                              "        %.2f          %.2f" \
                              % (dudy[0],yplusloc,phi,rhs,lhs,(n-2)/(n+1),CP))
                        break
                    last_lhs = lhs


#%% Critical gradients from Eichler's 0 deg channel, fig. 4.8 in his PhD thesis
g_c_eichler293 = np.array(\
[0.1074,0.1670,0.1670,0.2227,0.2266,0.3022,0.3062,0.3062,0.3976,0.3936,0.4930,\
 0.4930,0.4891,0.5964,0.5964,0.5964,0.7157,0.7157,0.7197,0.8350,0.8509,0.8588,\
 0.9781,0.9940,1.1412,1.1531,1.3121,1.3241,1.4871,1.5030,1.5149,1.5189,1.7336,\
 1.7376,1.7495,1.7495,1.7654,1.7813,1.8012,1.8012])*1e+05
x_eichler293 = np.array(\
[0.2913,0.3167   ,   0.3257    , 0.3346    , 0.3511    , 0.3779   ,   0.4012    ,\
 0.4053    , 0.4417     ,0.4445     ,0.4685  ,0.4761     ,0.4815   ,   0.4987  ,\
   0.5056    , 0.5166   ,   0.5330  ,    0.5440    , 0.5578   ,   0.5605    , 0.5674   ,\
   0.5736, 0.6017  ,    0.6113     ,0.6470    , 0.6580    , 0.6999    , 0.7130    ,\
   0.7507  ,    0.7590    , 0.7734  ,    0.7851    , 0.8613  ,0.8757   ,   0.8997   ,\
   0.9073   ,   0.9162   ,   0.9341    , 0.9602   ,   0.9602])

g_c_eichler293_4 = np.array(\
[0.831826401446652  ,   1.030741410488245,    3.218806509945748  ,\
  4.285714285714284])*1e+04
x_eichler293_4 = np.array(\
[0.250047755491882   , 0.250047755491882,    0.290162368672397   ,\
 0.290162368672397])

g_c_eichler293_2 = np.array(\
[0.723327305605786,    0.813743218806509    , 2.368896925858950  ,\
  2.820976491862567  ,   3.942133815551536     ,   4.683544303797468  ,\
  6.148282097649185,    6.889692585895117  ,   8.589511754068715])*1e+04
x_eichler293_2 = np.array(\
[0.285959885386819,    0.302769818529131  ,   0.333333333333333   ,\
 0.363514804202483,    0.363514804202483      ,   0.399808978032474   ,\
 0.399808978032474   , 0.434574976122254  ,   0.434574976122254])

### Experimental data from Eichler (2011)
g=open(basepath+'EichlerUFB3.txt','r+')
lines=g.readlines()
Eichler_phi = []
Eichler_UFBBAR = []

for i in range(len(lines)):
    lines_split = lines[i].split('\t')
    Eichler_phi.append(float(lines_split[0]))
    Eichler_UFBBAR.append(float(lines_split[1]))

### Results of the original Hoferichter model (2017)
Hoferichter_phi = [0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9]
Hoferichter_UFBBAR = [ 6.70798426,  8.82348554, 11.24479479, 14.19490623,\
                      17.3893729 ,
   20.82524271, 24.226564   , 27.77517374, 31.00090666, 33.88480434,
   36.21085847, 38.58727119,  7.09568617, 11.68560967, 17.18901776,
   23.21327254, 29.37823449, 35.52793146, 41.40220583, 46.84775837,
   51.89728874, 56.47327076, 60.28544781, 63.75527976, 22.546382   ,
   31.16507735, 39.94901987, 48.50328199, 56.73712225, 64.53647118,
   71.73508127, 78.45280993, 84.50951192, 89.88303058, 94.34449072,
   98.41211723,  8.12450355,  9.88003665, 12.11533307, 14.9098337 ,
   17.9397085 , 21.19220129, 24.39198498, 27.70699492, 30.68707647,
   33.30501874, 35.36029642, 37.44624207]

### Results of the original Hoferichter model with Tober's modifications (2019)
Tober_phi = []
Tober_UFBBAR = []
datafile_path =\
 basepath+"ToberResults/"\
       "JoeriModifiedResults_Channel_T293K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path =\
basepath+"ToberResults/"\
 "JoeriModifiedResults_Channel_T473K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path =\
basepath+"ToberResults/"\
 "JoeriModifiedResults_Channel_T673K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

datafile_path =\
basepath+"ToberResults/"\
 "JoeriModifiedResults_Tube_T293K.txt"
with open(datafile_path, 'r+') as datafile_id:
    data = np.loadtxt(datafile_id)
    for i in range(0,len(data)):
        Tober_phi.append(data[i][0])
        Tober_UFBBAR.append(data[i][1])

#%% Plot results in terms of g_c
plt.rc('xtick',labelsize=18)
plt.rc('ytick',labelsize=18)
```

```
fig2 , axs1 = plt.subplots(nrows=1, ncols=1, figsize=(12, 6))

axs1.plot(x_eichler293 ,g_c_eichler293 ,label=\
          '0deg channel, Experiments (Eichler 2011)',c='b',marker='s',\
          linestyle=\
          'none',markersize=10,fillstyle='full')
axs1.plot(x_eichler293_2 ,g_c_eichler293_2 ,label=\
          '2deg diffuser, Experiments (Eichler 2011)',c='g',marker='d',\
          linestyle=\
          'none',markersize=10,fillstyle='full')
axs1.plot(x_eichler293_4 ,g_c_eichler293_4 ,label=\
          '4deg diffuser, Experiments (Eichler 2011)',c='r',marker='^',\
          linestyle=\
          'none',markersize=10,fillstyle='full')

axs1.plot(phi_dict[0][1.25],gc_dict[0][1.25],label=\
          '0deg channel, C = %0.1f' % C[0],c='b',marker='s',\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[2][0.125],gc_dict[2][0.125],label=\
          '2deg diffuser, C = %0.1f, x/L = 1/4' % C[2],c='g',marker="h",\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[2][0.250],gc_dict[2][0.250],label=\
          '2deg diffuser, C = %0.1f, x/L = 1/2' % C[2],c='g',marker="p",\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[2][0.375],gc_dict[2][0.375],label=\
          '2deg diffuser, C = %0.1f, x/L = 3/4' % C[2],c='g',marker='d',\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[4][0.125],gc_dict[4][0.125],label=\
          '4deg diffuser, C = %0.1f, x/L = 1/4' % C[4],c='r',marker="<",\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[4][0.250],gc_dict[4][0.250],label=\
          '4deg diffuser, C = %0.1f, x/L = 1/2' % C[4],c='r',marker=">",\
          markersize=10,linestyle='none',fillstyle='none')
axs1.plot(phi_dict[4][0.375],gc_dict[4][0.375],label=\
          '4deg diffuser, C = %0.1f, x/L = 3/4' % C[4],c='r',marker='^',\
          markersize=10,linestyle='none',fillstyle='none')


axs1.set_xlabel('$\phi$ [-]', fontsize=20)
axs1.set_ylabel('g$_c$ [1/s]', fontsize=20)
axs1.set_title('T = 293K',fontsize=20)
XLIM = [0.2,1.2]
YLIM = [0,150000]
NN = 4
axs1.set_xlim(XLIM[0],XLIM[1])
axs1.set_ylim(YLIM[0],YLIM[1])

axs1.grid(True)
axs1.legend(loc=4,fontsize=14)

bg = 0.2
sm = bg/5
axs1.xaxis.set_major_locator(ticker.MultipleLocator(bg))
axs1.xaxis.set_minor_locator(ticker.MultipleLocator(sm))

bg = 5e4
sm = bg/5
axs1.yaxis.set_major_locator(ticker.MultipleLocator(bg))
axs1.yaxis.set_minor_locator(ticker.MultipleLocator(sm))

axs1.ticklabel_format(axis='y', style='sci', scilimits=(0,0))
# Save figure as .png ----------------------------------------------------------------
fname = \
basepath+'Images/Model results/gc_zoom.png'
plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
orientation='portrait', papertype=None, format=None,
transparent=False, bbox_inches=None, pad_inches=0.1,
frameon=None, metadata=None)

#%% Plot results in terms of U_FB_bar locally in channel/diffuser
plt.rc('xtick',labelsize=18)
plt.rc('ytick',labelsize=18)
black = 'k'
red = 'r'

fig3 , axs3 = plt.subplots(nrows=1, ncols=1, figsize=(6, 6))

axs3.plot(Eichler_phi[67:67+39],Eichler_UFBBAR[67:67+39],\
          label='0deg channel, Experiments by Eichler (2011)',marker='s',\
          c='b',markersize=8,fillstyle='full',linestyle='none')

axs3.plot(phi_dict[0][1.25],ufbbarlocal_dict[0][1.25],\
          label='0deg channel, C = %0.1f' % C[0],c='b',marker='s',\
          markersize=10,linestyle='none',fillstyle='none')

axs3.plot(phi_dict[2][0.125],ufbbarlocal_dict[2][0.125],\
          label='2deg diffuser, C = %0.1f, x/L = 1/4' % C[2],c='g',marker='h',\
          markersize=10,linestyle='none',fillstyle='none')
axs3.plot(phi_dict[2][0.250],ufbbarlocal_dict[2][0.250],\
          label='2deg diffuser, C = %0.1f, x/L = 2/4' % C[2],c='g',marker='p',\
          markersize=10,linestyle='none',fillstyle='none')
axs3.plot(phi_dict[2][0.375],ufbbarlocal_dict[2][0.375],\
          label='2deg diffuser, C = %0.1f, x/L = 3/4' % C[2],c='g',marker='d',\
          markersize=10,linestyle='none',fillstyle='none')
axs3.plot(phi_dict[4][0.125],ufbbarlocal_dict[4][0.125],\
          label='4deg diffuser, C = %0.1f, x/L = 1/4' % C[4],c='r',marker='<',\
          markersize=10,linestyle='none',fillstyle='none')
axs3.plot(phi_dict[4][0.250],ufbbarlocal_dict[4][0.250],\
          label='4deg diffuser, C = %0.1f, x/L = 2/4' % C[4],c='r',marker='>',\
          markersize=10,linestyle='none',fillstyle='none')
axs3.plot(phi_dict[4][0.375],ufbbarlocal_dict[4][0.375],\
          label='4deg diffuser, C = %0.1f, x/L = 3/4' % C[4],c='r',marker='^',\
          markersize=10,linestyle='none',fillstyle='none')

axs3.set_xlabel('$\phi$ (-)', fontsize=20)
axs3.set_ylabel('$\overline{U}_{FB,local}$ (m/s)', fontsize=20)
axs3.set_title('T=293 K',fontsize=20)
XLIM = [0.2,1]
YLIM = [0,90]
NN = 4
axs3.set_xlim(XLIM[0],XLIM[1])
axs3.set_ylim(YLIM[0],YLIM[1])
```

```
axs3.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
axs3.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
axs3.grid(True)

# Save figure as .png -----------------------------------------------------------
fname = \
basepath+'Images/Model results/ufbbar_local.png'
plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
orientation='portrait', papertype=None, format=None,
transparent=False, bbox_inches=None, pad_inches=0.1,
frameon=None, metadata=None)

#%% Plot results in terms of U_FB_bar at inlet of channel/diffuser
plt.rc('xtick',labelsize=18)
plt.rc('ytick',labelsize=18)
black = 'k'
red = 'r'

fig2, axs2 = plt.subplots(nrows=1, ncols=1, figsize=(6, 6))

axs2.plot(Eichler_phi[67:67+39],Eichler_UFBBAR[67:67+39],\
        label='0deg channel, Experiments by Eichler (2011)',marker='s',\
        markersize=8,c='b',fillstyle='full',linestyle='none')

axs2.plot(phi_dict[0][1.25],ufbbar_dict[0][1.25],\
        label='0deg channel, this work',c='b',marker='s',\
        markersize=10,linestyle='none',fillstyle='none')


axs2.plot(phi_dict[2][0.125],ufbbar_dict[2][0.125],label=\
        '2deg diffuser, this work, x/L = 1/4',c='g',marker='h',markersize=10,\
        linestyle='none',fillstyle='none')
axs2.plot(phi_dict[2][0.250],ufbbar_dict[2][0.250],label=\
        '2deg diffuser, this work, x/L = 2/4',c='g',marker='p',markersize=10,\
        linestyle='none',fillstyle='none')
axs2.plot(phi_dict[2][0.375],ufbbar_dict[2][0.375],label=\
        '2deg diffuser, this work, x/L = 3/4',c='g',marker='d',markersize=10,\
        linestyle='none',fillstyle='none')
axs2.plot(phi_dict[4][0.125],ufbbar_dict[4][0.125],label=\
        '4deg diffuser, this work, x/L = 1/4',c='r',marker='<',\
        markersize=10,linestyle='none',fillstyle='none')
axs2.plot(phi_dict[4][0.250],ufbbar_dict[4][0.250],label=\
        '4deg diffuser, this work, x/L = 2/4',c='r',marker='>',\
        markersize=10,linestyle='none',fillstyle='none')
axs2.plot(phi_dict[4][0.375],ufbbar_dict[4][0.375],label=\
        '4deg diffuser, this work, x/L = 3/4',c='r',marker='^',\
        markersize=10,linestyle='none',fillstyle='none')


axs2.set_xlabel('$\phi$ (-)', fontsize=20)
axs2.set_ylabel('$\overline{U}_{FB,inlet}$ (m/s)', fontsize=20)
axs2.set_title('T=293 K',fontsize=20)
XLIM = [0.2,1]
YLIM = [0,90]
NN = 4
axs2.set_xlim(XLIM[0],XLIM[1])
axs2.set_ylim(YLIM[0],YLIM[1])
axs2.set_xticks=\
(np.arange(XLIM[0],XLIM[1]+(XLIM[1]-XLIM[0])/NN,step=(XLIM[1]-XLIM[0])/NN))
axs2.set_yticks=\
(np.arange(YLIM[0],YLIM[1]+(YLIM[1]-YLIM[0])/NN/2,step=(YLIM[1]-YLIM[0])/NN/2))
axs2.grid(True)

# Save figure as .png -----------------------------------------------------------
fname = \
basepath+'Images/Model results/ufbbar_inlet.png'
plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
orientation='portrait', papertype=None, format=None,
transparent=False, bbox_inches=None, pad_inches=0.1,
frameon=None, metadata=None)
```