



Master Thesis, ${\rm c971820}$

Computational Fluid Dynamics in Microfluidic Systems

Laurits Højgaard Olesen



Supervisor: Henrik Bruus

Mikroelektronik Centret (MIC) Technical University of Denmark

31 July 2003

ii

Abstract

Computer simulations are an indispensable tool in microfluidics because of the lack of analytical solutions. We have developed a simulation tool in MATLAB based on the finite element method (FEM). At the present stage, the tool can be employed to handle general second order partial differential equations in two dimensions.

The tool has been applied to model incompressible laminar flow, and it has been tested on the classical problem of flow over a backward-facing step.

Also the tool has been employed to model non-Newtonian blood flow in a microchannel, in relation to an experiment performed by Lennart Bitsch, using micro particle-image velocimetry (μ PIV) measurements to map out the velocity profile of blood flowing in a thin glass capillary. The joint work has resulted in two conference proceedings [1, 2] and a paper submitted to *Experiments in Fluids* [3].

Finally we have considered the problem of electroosmotic flow in a pore system on the submicron scale where the Debye layer overlap is non-neglible. At the time of writing the results are still preliminary, yet they demonstrate that our FEM tool is general enough to accomodate for more complex problems occuring in the field of microfluidics.

Copyright by Laurits Højgaard Olesen All rights reserved

Published by: Mikroelektronik Centret Technical University of Denmark DK-2800 Lyngby

ISBN: 87-89935-70-5

Resumé

Computer simuleringer er et uundværligt værktøj i mikrofluidik på grund af manglen på analytiske løsninger. Vi har udviklet et simuleringsværktøj i MATLAB baseret på finite element metoden (FEM). På dets nuværende stade kan værktøjet benyttes til at behandle generelle anden ordens partielle differentialligninger i to dimensioner.

Værktøjet er anvendt til at modellere inkompressibel laminar strømning, og dette er testet på det klassiske problem med strømning over et nedadgående trin.

Værktøjet er også anvendt til at modellere ikke-Newtonsk strømning af blod i en mikrokanal, i forbindelse med et eksperiment udført af Lennart Bitsch, hvor micro particle-image velocimetry (μ PIV) målinger er benyttet til at bestemme hastighedsprofilen for blod strømmende i en tynd glaskapillar. Arbejdet har resulteret i to konference proceedings [1, 2] og en artikel indsendt til *Experiments in Fluids* [3].

Endelig har vi betragtet elektroosmotisk strømning i et poresystem på submikron skala, hvor man ikke kan se bort fra overlap af Debye lagene. I skrivende stund er resultaterne stadig usikre, men viser dog at vores FEM værktøj er generelt nok til at kunne beskrive mere komplekse problemer, der optræder indenfor mikrofluidikken.

Preface

The present master thesis is submitted in candidacy to the cand. polyt. title at the Technical University of Denmark. The work has been carried out at Mikroelektronik Centret, MIC in the microfluidic theory and simulation group (MIFTS) under supervision of Henrik Bruus from September 2002 to July 2003.

I would like to thank Henrik Bruus for dedicated effort as supervisor. Also I would like to thank Jess Michelsen for his advice and discussions during the project, and Lennart Bitsch for a good collaboration.

> Laurits Højgaard Olesen Mikroelektronik Centret (MIC) Technical University of Denmark 31 July 2003

Contents

Li	st of	figures			xiv
\mathbf{Li}	st of	tables			$\mathbf{x}\mathbf{v}$
\mathbf{Li}	st of	symbols		х	vii
1	Intr	oduction			1
	1.1	μ PIV paper submitted to <i>Experiments in Fluids</i>	•	•	2
2	Bas	ic hydrodynamics			5
	2.1	Physics of fluids			5
		2.1.1 The continuity equation			5
		2.1.2 The momentum equation			6
		2.1.3 The energy equation			8
	2.2	Poiseuille flow			8
		2.2.1 Flow in a circular capillary			9
		2.2.2 Flow in rectangular channel			10
		2.2.3 Flow in triangular and Gaussian shaped channels .	•	•	14
3	The	finite element method			17
	3.1	Discretization			17
	3.2	Weak solutions			19
		3.2.1 The Galerkin method			21
		3.2.2 The Lax-Milgram theorem			22
	3.3	Finite elements			23
		3.3.1 Lagrange elements			24
		3.3.2 Convergence properties			25
	3.4	Quadrature			29^{-3}
	-	3.4.1 Analytical expressions	_		32
		3.4.2 Gauss-Legendre quadrature			35
	3.5	Implementation			39
	5.0	3.5.1 Mesh data structure			40
		3.5.2 Basis functions	•	•	41
		3.5.3 Quadrature revisited			41
		······································	-		

	3.5.4 Solution of linear system of equations	42 45
4	Application of EEM to hydrodynamics	17
4	4.1 Discretization of the Navier Steles equation	17
	4.1 Discretization of the Navier-Stokes equation	±1 10
	4.2 The Babuška-Brezzi inf-sup condition	19 52
	4.3 The pressure gradient projection method (PGP)	54
5	Flow over backward-facing step 5	59
	5.1 Regular mesh refinement	30
	5.2 Adaptive mesh refinement $\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	31
	5.3 Bifurcation	34
6	Non-Newtonian flow	67
	6.1 Non-Newtonian liquids	37
	$6.1.1 \text{Blood} \dots \dots \dots \dots \dots \dots \dots \dots \dots $	38
	6.2 Blood flow in straight channel	70
	6.3 The μ PIV experiment	71
	6.4 Summary	75
7	Electroosmotic flow 7	7
	7.1 The Debye layer	78
	7.2 EOF in porous structure \ldots \ldots \ldots \ldots \ldots \ldots	30
	7.2.1 Pore geometry \ldots \ldots \ldots \ldots \ldots	32
8	Conclusion 8	37
\mathbf{A}	Computation of curvature 8	39
ъ		
в	Computation of streamlines 9	13
\mathbf{C}	Matlab function headers	17
	C.1 gaussgeom	<i>)</i> 7
	C.2 elm2sd	<i>)</i> 7
	C.3 halfmesh) 8
	C.4 prolong	98 20
	$C.5 \text{ Lagrange } \dots $	99 20
	$0.0 \text{ mglln} \dots 0.0 \text{ mglln}$	99 20
	C_{s}	90 10
	$\begin{array}{c} \text{O} \text{graval} \\ \begin{array}{c} \text{O} \text{graval} \\ \end{array} $	שע 19
	C = 10 neut	⊿ו או
	C 11 flow 2 dngn	יסי אר
	C 12 gcontour)4
	0.12 Boomoont) T

$C.13 \; \texttt{strmfunc}$	105
D Paper submitted to Experiments in Fluids	107

List of Figures

2.1	Velocity profile in a 3×1 rectangular channel	11
2.2	Maximal velocity vs. aspect ratio for various channel geometries	13
2.3	Average velocity vs. aspect ratio for various channel geomeries. \ldots .	13
2.4	Cross-section of microchannel produced in PMMA using the laser abla-	
	tion technique	14
2.5	Finite element mesh and contour plot of velocity profile in Gaussian	
	shaped channel.	16
3.1	Computational grid for the finite difference method	18
3.2	Computational mesh for the finite element method. \ldots	18
3.3	Lagrange element basis functions for triangular elements. \ldots .	26
3.4	Lagrange element basis functions for quadrangular elements	27
3.5	Mapping from the $xy\mbox{-}{\rm plane}$ to the integration coordinate $\xi\eta\mbox{-}{\rm plane}$	30
4.1	Geometry of channel with a bend.	51
4.2	Flow in channel with a 90° bend using same basis functions for both	
	velocity and pressure. \ldots	51
4.3	One-dimensional analogy to unstable pressure mode	52
4.4	Flow in channel with a 90° bend using quadratic velocity and linear	
	pressure basis functions.	53
4.5	Flow in channel with a 90° bend using the PGP method. $\hfill \hfill \h$	56
4.6	Convergence in the H^1 norm	57
5.1	Channel with backward-facing step	60
5.2	Solution for $Re = 100$ after three regular mesh refinements	61
5.3	Solution for $Re = 100$ after two adaptive mesh refinements	63
5.4	Solution for $Re = 100$ after six adaptive mesh refinements	63
5.5	Solution for $Re = 500$ after ten adaptive mesh refinements	65
5.6	Simulated results for location of detachment and reattachment point as	
	function of Reynolds number	66
5.7	Experimental results for location of detachment and reattachment point as function of Reynolds number. <i>Reproduced from</i> [19]	66
6.1	Measurement of the apparent viscosity of blood as function of shear rate.	70

6.2	Velocity profiles obtained with power law viscosity model and with fit to	
	experimental data for blood viscosity	72
6.3	Cross section of glass capillary used in $\mu {\rm PIV}$ experiment. $\hfill \hfill \hfil$	73
7.1	Pore geometry	83
7.2	Solution at zero applied potential.	84
7.3	Solution with applied potential of 5.3 mV	85

List of Tables

3.1	Terms in bivariate polynomial	24
3.2	Gaussian quadrature rules	37
5.1	Convergence for $Re = 100$ with regular mesh refinement	62
5.2	Convergence for $Re = 100$ with adaptive mesh refinement	64

List of symbols

Symbol	Description	Unit
$\overline{c_i}$	Molar concentration	$mol m^{-3}$
	- or number density	m^{-3}
$D_{\mathrm{mass},i}$	Mass diffusion coefficient	$m^2 s^{-1}$
e	Elementary charge	$1.602 \times 10^{-19} \text{ C}$
f	Body force density	${ m N~m^{-3}}$
G	Pressure gradient	Pa m^{-1}
g	Gravity	$ m N~kg^{-1}$
$oldsymbol{J}_i$	Molar flux vector	$mol \ s^{-1} \ m^{-2}$
k_B	Boltzmann constant	$1.381 \times 10^{-23} \text{ J K}^{-1}$
\boldsymbol{n}	Surface outward normal	
p	Pressure	$\rm N~m^{-2}$
\overline{Q}	Volume flow rate	$m^{3} s^{-1}$
T	Temperature	K
v	Velocity vector	${\rm m~s^{-1}}$
$oldsymbol{x}$	Position vector	m
z_i	Number of charges	
ϵ	Dielectric constant	$C V^{-1} m^{-1}$
ϵ_0	Permittivity of vacuum	$8.854 \times 10^{-12} \mathrm{C} \mathrm{V}^{-1} \mathrm{m}^{-1}$
$\dot{\gamma}$	Shear rate	s^{-1}
$\dot{\gamma}$	Magnitude of shear rate	s^{-1}
λ_D	Debye length	m
μ	Dynamic viscosity	$kg m^{-1} s^{-1}$
$\mu_{ m eo}$	Electroosmotic mobility	$m^2 V^{-1} s^{-1}$
μ_i	Mobility	$m^2 V^{-1} s^{-1}$
ϕ	Electrostatic potential	V
ho	Mass density	${ m kg}~{ m m}^{-3}$
σ	Cauchy stress tensor	$\rm N~m^{-2}$
au	Deviatoric stress tensor	${ m N~m^{-2}}$
ζ	Zeta potential	V

Symbol	Description
\mathcal{L}	Differential operator
u	Solution
f	Source term
Ω	Computational domain
$\partial \Omega$	Domain boundary
\mathbb{H}	Function space
\mathbb{H}^h	Finite dimensional function space
h	Mesh size
v,w	Members of function space
$arphi_k$	kth basis function
p	Accuracy
$\langle \cdot, \cdot angle$	Inner product

Chapter 1

Introduction

Microfluidics and the concept of micro total analysis systems (μ TAS) is a new and promising technology expected to revolutionize chemical and medical analysis systems.

It is envisaged to miniaturize all components involved in a chemical analysis and integrate them on a single microchip to form a so-called labon-a-chip system. There are many advantages to such a system, including very low sample consumption, high degree of portability, and possibility of cheap mass production by use of standard microtechnology batch processing.

At MIC several groups are working on different aspects of the lab-ona-chip realization. The microfluidic theory and simulation group (MIFTS), within which the present work has been carried out, has its focus on combining theory and simulation efforts to obtain a thorough understanding of the basic physical principles involved in microfluidic systems.

The main goals of the present project were first the development of an in-house general simulation tool for modelling of different problems arising in microfluidics. Next the tool was to be applied to simulate blood flow in microchannels as discussed below, and more generally it was to be applied to electroosmotic flow problems investigating e.g. electrochemical effects at the electrodes.

Now, one may indeed ask why we would want to develop our own software for simulation when powerful commercial software such as CFD-ACE+ and COVENTOR exist and is already in use within MIFTS. However the commercial programs do have limitations as to what they can handle, and it is typically rather difficult if not impossible to work around such limitations.¹ With in-house software it should be possible to modify the program to accomodate any specific requirements.

The tool that I have developed is based on the finite element method

¹E.g. in COVENTOR it has been found that internal solid walls specified in a geometry do act as barries for convection but not for diffusion of a dissolved species – thus effectively making it impossible to model electrodes isolated from the flow. [4]

(FEM) which is a powerful technique, in particular for problems involving complex geometries. Actually the FEM was new to me, and it took some time to get into all aspects of the method. For the implementation of the tool I initially worked with the C programming language; however I found it cumbersome and little flexible, in particular since the outline of the tool was not well defined at this early stage. Therefore it was chosen to translate the code into MATLAB since debugging and testing is typically much faster in this environment. Also it allowed to rely on MATLAB's built-in linear solvers and on its many graphical routines for postprocessing of the solution. Furthermore because of the more compact and simple syntax I believe it will be easier for future users to modify and extend the MATLAB program than an equivalent one written in C.

While the initial approach was coded specifically for flow problems, during the project I have continuously been extending the capabilities and tried to abstract the tool to be able to handle general partial differential equation problems. In that sense it is heading towards a form that is similiar to that of FEMLAB, which is a commercial generic FEM program that runs on top of MATLAB.

1.1 μ PIV paper submitted to *Exp. Fluids*

During the project I have been applying the FEM tool to model blood flow in a thin glass capillary in connection to a series of experiments performed by Lennart Bitsch. The experiments were so-called micro particle-image velocimetry (μ PIV) measurements, monitoring the motion of individual blood cells in order to observe the velocity profile in the capillary. Also I have been involved with parts of the data analysis and in calculations on the optics in the experimental setup. This joint work has resulted in two conference proceedings [1, 2] and a paper submitted to *Experiments in Fluids* [3]. The experiment is described in some detail in Chap. 6 and further I have included the paper in Appendix D.

The structure of the thesis is as follows:

- In Chap. 2 on basic hydrodynamics the governing equations of fluid dynamics are stated and we discuss flow in a long straight channel.
- In Chap. 3 we give an overview of the finite element method with particular focus on some aspects central to our implementation of the method.
- In Chap. 4 the FEM is applied to the incompressible flow problem; the most straightforward approach turns out to yield a solution with a spurious pressure oscillation, and two ways of dealing with this problem are discussed.

- In Chap. 5 we test our implementation on a classical problem of flow over a backwards facing step.
- Then in Chap. 6 we discuss non-Newtonian flow and blood flow in particular in relation to the μ PIV experiment.
- In Chap. 7 the basic concepts regarding electroosmotic flow are introduced and our first results applying the FEM tool to this problem presented.

Chapter 2

Basic hydrodynamics

In this chapter we state the governing equations for fluid motion which are the continuity equation, the Navier-Stokes equation, and the energy equation. Especially the Navier-Stokes equation and the Cauchy stress tensor are discussed in some detail. While the same derivation is found in any continuum physics textbook, we include it here because we in Chap. 6 are going to discuss non-Newtonian liquids, blood in particular, in relation to which a proper definition of the stress tensor and the shear rate is important.

The second part of the chapter is devoted to analysis of Poiseuille flow, that is, flow in long straight channels of various cross sections. The Poiseuille flow is treated, in part because experimental colleagues at MIC were interested in details on the flow resistance in their microchannels; and in part it is treated to have a background for discussing the experiments on blood flow in microchannels that we present in Chap. 6.

2.1 Physics of fluids

A fluid ultimately consist of molecules but we describe the system in terms of continuous fields such as the mass density $\rho(\boldsymbol{x})$ and the velocity field $\boldsymbol{v}(\boldsymbol{x})$. We also speak of material particles which should be thought of as small volumes of fluid that travel along with the flow, always containing the same fluid molecules, that is, neglecting diffusion.

2.1.1 The continuity equation

We consider a material particle of position $\boldsymbol{x}(t)$ and of volume $\mathcal{V}(t)$ riding along with the flow. The mass m of the particle can then be expressed as the local density of the fluid times the particle volume $m = \rho(\boldsymbol{x}, t)\mathcal{V}(t)$. The rate of change of this quantity is determined from

$$\frac{d}{dt} \Big(\rho[\boldsymbol{x}(t), t] \, \mathcal{V}(t) \Big) = \frac{\partial \rho}{\partial t} \mathcal{V} + \boldsymbol{\nabla} \rho \cdot \frac{d\boldsymbol{x}}{dt} \mathcal{V} + \rho \frac{d\mathcal{V}}{dt} = \Big[\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) \Big] \mathcal{V}, \quad (2.1)$$

where the second equality follows from the definition of the velocity field $d\boldsymbol{x}/dt = \boldsymbol{v}$ and the change in particle volume is expressed by the divergence of the velocity field lines $d\mathcal{V}/dt = (\boldsymbol{\nabla} \cdot \boldsymbol{v})\mathcal{V}$. Now since the mass of the particle is conserved we thus arrive at the equation of continuity

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0. \tag{2.2}$$

For constant density this reduces to the incompressibility constraint

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0. \tag{2.3}$$

Notice however that the opposite is not true – with a divergence free velocity field and a density constant in time the continuity equation only implies that the density is constant along streamlines.

2.1.2 The momentum equation

Again we consider a material particle riding along with the flow. The particle momentum is then p = mv, with rate of change

$$\frac{d\boldsymbol{p}}{dt} = m\frac{d}{dt} \left(\boldsymbol{v}[\boldsymbol{x}(t), t] \right) = m \left(\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{v} \right).$$
(2.4)

According to Newtons second law this is equal to the sum of the external forces. For a particular particle those are the contact force from the surrounding fluid $\nabla \cdot \sigma \mathcal{V}$, where σ is the Cauchy stress tensor, together with the external body forces acting on the particle $f\mathcal{V}$ where f is the body force density, e.g. gravity ρg . Thus we arrive at Cauchy's equation of motion

$$\rho \left[\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla}) \boldsymbol{v} \right] = \boldsymbol{f} + \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}.$$
(2.5)

The stress tensor σ is defined such that the contact force \mathcal{F} that the material on one side of a small surface patch of area A and outward normal n feels from the material on the other side of the patch is

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\sigma} \cdot \boldsymbol{n} \boldsymbol{A},\tag{2.6}$$

and it can be shown that $\boldsymbol{\sigma}$ is symmetric, that is $\sigma_{ij} = \sigma_{ji}$. Conventionally the stress tensor is split according to

$$\sigma_{ij} = -p\,\delta_{ij} + \tau_{ij},\tag{2.7}$$

where δ_{ij} is the Kronecker delta and p is the hydrodynamic pressure while τ_{ij} is the so-called deviatoric stress tensor. By definition the hydrodynamic pressure is given in terms of the trace of the stress tensor as

$$p = -\frac{1}{3} \sum_{k} \sigma_{kk} = -\frac{1}{3} \operatorname{Tr} \boldsymbol{\sigma}, \qquad (2.8)$$

and for a fluid at hydrostatic equilibrium this coincides with the hydrostatic pressure since fluids at rest cannot sustain shear stress. Hence the deviatoric stress tensor

$$\tau_{ij} = \sigma_{ij} + \delta_{ij}p \tag{2.9}$$

is said to be pressure free since by construction $\text{Tr} \tau = 0$. With these definitions the Cauchy equation of motion reads

$$\rho \left[\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla}) \boldsymbol{v} \right] = -\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot \boldsymbol{\tau} + \boldsymbol{f}.$$
(2.10)

Newtonian fluids

As mentioned a fluid at rest cannot sustain shear stress, and generally we associate the shear forces with friction between different regions of the fluid flowing at different speed, that is, the deviatoric stress tensor should depend on the velocity gradients in the flow. As a measure of the velocity gradients in the flow we define the shear rate tensor $\dot{\gamma}$

$$\dot{\gamma}_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) - \delta_{ij} \frac{1}{3} \sum_k \frac{\partial v_k}{\partial x_k} .$$
(2.11)

In a Newtonian fluid the deviatoric stress tensor τ is simply proportional to the shear rate, with

$$\boldsymbol{\tau} = 2\mu \dot{\boldsymbol{\gamma}},\tag{2.12}$$

where μ is called the dynamic viscosity. The viscous force in Eq. (2.10) is then obtained as the divergence of this

$$\boldsymbol{\nabla} \cdot \boldsymbol{\tau} = \mu \big[\boldsymbol{\nabla}^2 \boldsymbol{v} + \frac{1}{3} \boldsymbol{\nabla} (\boldsymbol{\nabla} \cdot \boldsymbol{v}) \big], \qquad (2.13)$$

and assuming incompressible flow $\nabla \cdot \boldsymbol{v} = 0$, we obtain the Navier-Stokes equation in its perhaps most familiar form

$$\rho \left[\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla}) \boldsymbol{v} \right] = -\boldsymbol{\nabla} p + \mu \boldsymbol{\nabla}^2 \boldsymbol{v} + \boldsymbol{f}.$$
(2.14)

Non-Newtonian fluids

While simple liquids such as water and ethanol behave as Newtonian liquids, more complex ones such as blood, toothpaste and polymer solutions do not. There are several ways that a fluid can deviate from Newtonian behaviour and we shall discuss this in detail in Chap. 6. The most simple case is the socalled generalized Newtonian liquids for which the viscosity is not a simple constant but depends on the magnitude of the velocity gradients present in the flow. Thus defining the magnitude of the shear rate as

$$\dot{\gamma} = \sqrt{\frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \dot{\gamma}_{ij} \dot{\gamma}_{ji}} , \qquad (2.15)$$

a generalized Newtonian liquid follows a constitutive equation of the form $\tau = 2\mu(\dot{\gamma})\dot{\gamma}$.

2.1.3 The energy equation

We define the field $\mathcal{U}(\boldsymbol{x},t)$ as the local total energy per unit mass. Then the energy of a material particle can be written as $\mathcal{E} = \rho \mathcal{U} \mathcal{V}$. The particle energy increases from work done by the external forces $dW = (\boldsymbol{f} + \boldsymbol{\nabla} \cdot \boldsymbol{\sigma})\mathcal{V} \cdot \boldsymbol{v}dt$ and from heat sources $q\mathcal{V}$ acting inside the particle while it decreases from the heat flux \boldsymbol{q} out of the particle surface. Thus

$$\frac{d}{dt} \left(\rho \, \mathcal{U} \mathcal{V} \right) = (\boldsymbol{f} + \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}) \mathcal{V} \cdot \boldsymbol{v} + q \mathcal{V} - \int_{\mathcal{S}} \mathrm{d}S \, \boldsymbol{n} \cdot \boldsymbol{q}.$$
(2.16)

By Fouriers law the heat flux is $q = -k\nabla T$ where T is the temperature and k is the thermal conductivity. Then applying Gauss' theorem to the surface integral and cancelling out \mathcal{V} we arrive at

$$\frac{\partial(\rho \mathcal{U})}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathcal{U} \boldsymbol{v}) = (\boldsymbol{f} + \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}) \cdot \boldsymbol{v} + q + \boldsymbol{\nabla} \cdot (k \boldsymbol{\nabla} T).$$
(2.17)

For an incompressible Newtonian fluid with constant density and isotropic thermal conductivity this reduces to

$$\rho\left(\frac{\partial \mathcal{U}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \mathcal{U}\right) = \boldsymbol{f} \cdot \boldsymbol{v} + \frac{1}{2}\mu \boldsymbol{\nabla}^2(\boldsymbol{v} \cdot \boldsymbol{v}) + q + k\boldsymbol{\nabla}^2 T.$$
(2.18)

Finally the internal energy per unit mass u is obtained by subtracting the kinetic energy per unit mass, thus $u = \mathcal{U} - \frac{1}{2}v^2$.

2.2 Poiseuille flow

Flow in a circular capillary or pipe is a simple example where the analytical solution for the velocity field can be obtained. The relation between applied pressure and flow rate in a cylindrical pipe is know as Hagen-Poiseuille's law and is of great importance because of the common industial use of cylindrical pipelines.

In silicon based microfluidic systems channels are rather of rectangular cross section, and on microchips fabricated in polymers by the laser ablation technique the channels tend to have a cross section that can be described as a triangular or Gaussian shaped trench.

Experimental colleagues at MIC were in particular interested in knowing the hydraulic resistance of such channels. This problem provided a neat startup problem for the FEM tools developed during the present project work.

We consider generally a steady laminar incompressible flow in a long straight channel extending in the z-direction and of some arbitrary cross section in the xy-plane. For symmetry reasons the velocity field must be of the form $\boldsymbol{v} = v_z(x, y)\boldsymbol{e}_z$ and further assuming no external body forces the pressure must be $p(z) = p_0 - G z$ where p_0 and G are constants. Inserting this into the Navier-Stokes equation Eq. (2.14) we obtain

$$G + \mu \nabla^2 v_z(x, y) = 0, \qquad (2.19)$$

that is, a Poisson problem in two dimensions for v_z with constant source term $-G/\mu$. The boundary condition that v_z should satisfy at the channel wall is the no-slip condition flow where the velocity at the walls is required to be zero¹.

2.2.1 Flow in a circular capillary

In the case of a channel of cylindrical cross section a description in cylindrical coordinates (r, ϕ, z) is adequate. Since the problem is axisymmetric the velocity field cannot depend on angle; thus $\boldsymbol{v} = v_z(r)\hat{\boldsymbol{z}}$ and the Navier-Stokes equation reduces to

$$G + \mu \nabla^2 v_z = G + \mu \Big[\frac{1}{r} \frac{\partial}{\partial r} \Big(r \frac{\partial}{\partial r} \Big) \Big] v_z(r) = 0, \qquad (2.20)$$

which has the general solution $v_z = -\frac{G}{4\mu}r^2 + A\ln r + B$. Infinite velocity at the channel center at r = 0 is unphysical so A = 0, and the no-slip condition at the channel wall at r = R yields $B = GR^2/4\mu$, so

$$v_z(r) = \frac{G}{4\mu}(R^2 - r^2).$$
(2.21)

That is, the velocity profile is a paraboloid with a maximal velocity proportional to the pressure gradient G, the inverse of the viscosity and the square of the channel radius.

The total volume flow rate Q is obtained by integrating the velocity field across the channel yielding

$$Q = \int_0^R \mathrm{d}r \ r \int_0^{2\pi} \mathrm{d}\phi \ v_z(r) = \frac{\pi G R^4}{8\mu}.$$
 (2.22)

Over a channel of length L the pressure drop is $\Delta p = GL$, so

$$Q = \frac{\pi R^4}{8\mu L} \Delta p \tag{2.23}$$

which is the famous Hagen-Poiseuille's law. Also the hydraulic resistance is defined as $R_{\rm hyd} = \Delta p/Q = 8\mu L/\pi R^4$ and we observe the drastic decrease in resistance with channel radius.

¹The no-slip condition applies to viscous flow and derives from the assumption that because of surface roughness or friction the fluid particles immediately next to the wall will have their momentum relaxed effectively towards that of the wall; thus in the case of moving walls no-slip implies zero fluid velocity relative to the wall. In extremely small channels or rarefied gas flows where channel dimensions become comparable to the particle mean free path, corrections to the no-slip condition are needed.

2.2.2 Flow in rectangular channel

We now consider flow in a channel of rectangular cross section of width W and depth H. We shall discuss how the velocity profile in the channel depends on the aspect ratio $\alpha = H/W$, and we first consider the limiting case of an infinitely wide channel, that is, of zero aspect ratio.

The problem is translationally invariant in the x-direction and as for the circular capillary the problem reduces to a simple ordinary differential equation

$$G + \mu \nabla^2 v_z^{\infty} = G + \mu \frac{\partial^2}{\partial y^2} v_z^{\infty}(y) = 0.$$
 (2.24)

Imposing the no-slip condition at the channel lid at y = H and at the bottom at y = 0 yields the solution

$$v_z^{\infty}(y) = \frac{G}{2\mu}y(H-y).$$
 (2.25)

The velocity profile is a parabola, with a maximal velocity at y = H/2 of $GH^2/8\mu$, that is, twice as large as the maximal velocity in a circular capillary of diameter H. Qualitatively this is easily understood since in the cylindrical geometry there are walls all around the channel slowing down/braking up the flow, whereas between the infinite plates there are only walls on two sides.

Finally the volume flow rate per unit width of the channel is

$$\frac{Q}{W} = \int_0^H dy \, v_z(y) = \frac{GH^3}{12\mu}.$$
(2.26)

Fourier sine-series solution

In a channel of finite width no simple analytical solution can be obtained. However the geometry is simple enough that we can obtain a semi-analytical result in terms of a Fourier sine series from which e.g. the dependence on aspect ratio can be extracted.

We define the computational domain as $(x, y) = [0, W] \times [0, H]$ for which an appropriate trial solution for the velocity, satisfying the no-slip boundary condition, is a Fourier sine-series of the form

$$v_z(x,y) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} a_{nm} \sin\left(\frac{n\pi x}{W}\right) \, \sin\left(\frac{m\pi y}{H}\right) \tag{2.27}$$

Inserting this into the Poisson equation Eq. (2.19) we obtain

$$\sum_{n,m}^{\infty} a_{nm} \pi^2 \left(\frac{n^2}{W^2} + \frac{m^2}{H^2}\right) \sin\left(\frac{n\pi x}{W}\right) \\ \sin\left(\frac{m\pi y}{H}\right) = \frac{G}{\mu}, \qquad (2.28)$$





Figure 2.1: Velocity profile in a 3×1 rectangular channel in units of G/μ . In the central part of the channel the profile is rather flat in the *x*-direction and almost parabolic in the *y*-direction. The maximal velocity is $v_{z,\max} = 0.1227G/\mu$ which is only 2% below the $G/8\mu$ of the unit depth infinitely wide channel.

from which the expansion coefficients a_{nm} can be determined by multiplying on both sides with $\sin(p\pi x/W)\sin(q\pi y/H)$ for some integer p and q and integrating over the channel. Then, since the sine's are orthogonal

$$a_{pq} = \frac{16G}{\mu\pi^4} \times \begin{cases} \left[pq \left(\frac{p^2}{W^2} + \frac{q^2}{H^2} \right) \right]^{-1} & \text{for both } p \text{ and } q \text{ odd,} \\ 0 & \text{otherwise.} \end{cases}$$
(2.29)

Thus

$$v_z(x,y) = \frac{16G}{\mu\pi^4} \sum_{\text{odd}\,n,m}^{\infty} \left[nm \left(\frac{n^2}{W^2} + \frac{m^2}{H^2} \right) \right]^{-1} \sin \left(\frac{n\pi x}{W} \right) \sin \left(\frac{m\pi y}{H} \right) \quad (2.30)$$

Again the total volume flow rate is obtained by integrating v_z across the channel yielding

$$Q = \frac{64G}{\mu\pi^6} WH \sum_{\text{odd } n,m}^{\infty} \left[n^2 m^2 \left(\frac{n^2}{W^2} + \frac{m^2}{H^2} \right) \right]^{-1}.$$
 (2.31)

Fig. 2.1 shows the velocity profile in a channel of width W = 3 and depth H = 1 as calculated by Eq. (2.30) truncated at n = m = 100. It is noticed that the velocity in the central part of the channel resembles the parabolic profile of an infinitely wide channel very much, the maximal velocity being only 2% below the $GH^2/8\mu$ of the fully parabolic one, Eq. (2.25). Thus the central part of a rectangular channel is 'far' from the sides even at this relatively modest aspect ratio.

Generally it is the smaller of the channel dimensions W and H that determines the maximal velocity. Assuming H < W and introducing $\alpha =$ H/W in Eq. (2.30) we obtain

$$v_{z,\max} = \frac{16G}{\mu\pi^4} H^2 \sum_{\text{odd } n,m}^{\infty} \frac{1}{nm(n^2\alpha^2 + m^2)} \quad .$$
 (2.32)

Fig. 2.2 shows the dependence of the maximal velocity on aspect ratio for the rectangular channel. Also results for a number of other geometries to be handled in the following sections are shown for comparision. The velocity is given in units of GL^2/μ where L is the smaller of the channel width and depth. We notice that the graph is symmetric around $\alpha = H/W = 1$ which is obvious since interchanging W and H yields the same rectangular channel only rotated by 90°.

Again Fig. 2.2 demonstrates how the velocity profile at the central part of the channel as represented by the maximal velocity is dominated by the nearest walls. Beyond $\alpha = 3$ the far sides are largely not 'felt' at the center where only the top and bottom walls determine the flow.

Fig. 2.3 shows the aspect ratio dependence of the total volume flow rate in the channel. The results shown are normalized with the channel area and thus actually define the average velocity \bar{v}_z in the channel

$$\frac{Q}{A} = \frac{1}{\int_{\Box} dx dy} \int_{\Box} dx dy \ v_z(x, y) \equiv \bar{v}_z$$
(2.33)

When comparing with the result for the channel maximal velocity, we see that the average velocity tends rather slowly towards the average velocity in an infinitely wide channel \bar{v}_z^{∞} . To understand this phenomenon, observe that roughly a wide channel can be divided into a central region of width W - 2H where the profile is almost as in an infinitely wide channel, and two side regions each of width H where the average velocity is only half \bar{v}_z^{∞} , see Fig. 2.1. Then the total average expected is

$$\bar{v}_z \approx \frac{1}{W} \Big[\bar{v}_z^{\infty} (W - 2H) + \frac{1}{2} \bar{v}_z^{\infty} 2H \Big] = \bar{v}_z^{\infty} (1 - \alpha).$$
 (2.34)

Thus the convergence is only linear for $\alpha \to 0$ which corresponds to an exponential approach to \bar{v}_z^{∞} in the logarithmic plot of Fig. 2.3. This would be revealed as a straight line with slope one in a double logarithmic plot of $\bar{v}_z^{\infty} - \bar{v}_z$ vs. α and actually is though we do not present the plot here.

In conclusion the velocity profile in the central part of the channel is a local property of the flow which is well approximated by the profile of an infinitely wide channel even at moderate aspect ratio. On the other hand the average velocity is an integrated global property for which the influence of the side regions drops out only as the proportion of the area of the side regions to the total area.



Figure 2.2: Maximal velocity $v_{z,\max}$ vs. aspect ratio $\alpha = H/W$ for various channel geometries. The velocity is given in units of GL^2/μ where G is the pressure gradient in the channel, μ is the dynamic viscosity, and L is the smaller of the channel width W and depth H. All curves tend towards the level of the maximal velocity in an infinitely wide rectangular channel at very small or large aspect ratio. However in both the triangular and Gaussian shaped cases the convergence towards $v_{z,\max}^{\infty}$ for $\alpha \to 0$ is faster than that for $\alpha \to \infty$.



Figure 2.3: Volume flow rate per unit area Q/A, or equivalently channel average velocity \bar{v}_z , vs. aspect ratio α . For the triangular channels the average velocity tends towards half that of the infinitely wide rectangular channel both for high and low aspect ratio, though it tends faster for low aspect ratio.



Figure 2.4: Cross-section of microchannel produced in PMMA using the laser ablation technique.

2.2.3 Flow in triangular and Gaussian shaped channels

In a microfluidic channel network fabricated in polymer by the laser ablation technique the channels tend to have a cross section that can be described as a triangular or Gaussian shaped trench, see Fig. 2.4

The geometry of the triangular and Gaussian shaped channels renders the solution of Eq. (2.19) in terms of Fourier expansion rather difficult. Instead we employ the finite element method (FEM) which is described in detail in Chap. 3. Actually the velocity profile in a Gaussian shaped channel was the first real problem handled with the FEM tool that was developed during the present project. While initially the tool was geared only for the simplest case of triangle elements with linear interpolation, the results presented in Figs. 2.2 and 2.3 for triangular and Gaussian shaped channels were obtained with quadratic interpolation. This is so because it yields more accurate result, in particular for the average velocity. With quadratic interpolation the analytical parabolic solution in an infinitely wide channel is obtained using just two triangle elements!



We consider first a triangular channel of width W and depth H. In the limiting cases of very small or large aspect ratio $\alpha = H/W$ we can describe the flow locally as the flow between two infinite planes separated by a distance that varies along the channel.

In the case of small aspect ratio the local channel depth is described by h(x) = H(1-2|x|/W). Thus from Eq. (2.25) we expect the velocity profile to follow approximately as $v_z(x,y) \simeq Gy[h(x) - y]/2\mu$. The maximal velocity would be found where the channel is deepest with $v_{z,\max} \to GH^2/8\mu$ for

 $\alpha \rightarrow 0$ and the total volume flow rate determined as

$$Q = \int_{\Delta} dx dy \, v_z(x, y)$$

$$\to \frac{G}{2\mu} \, 2 \int_0^{W/2} dx \, \int_0^{h(x)} dy \, y \big(h(x) - y \big) = \frac{GWH^3}{48\mu}, \qquad (2.35)$$

from which $\bar{v}_z = Q/A \to GH^2/24\mu$. Thus the average velocity in the wide triangular channel is half that of the wide rectangular one as is observed in Fig. 2.2 for small α . That is, while the velocity in the narrow regions of the channel decrease like h^2 , the area of those regions decrease linearly in h and the average remains one half of that of the constant depth case.

The same analysis carries through for large aspect ratio channels using w(y) = W(1 - y/H) and yields the same asymptotic results as for the small aspect ratio channel. However we notice from Figs. 2.2 and 2.3 that the results for $\alpha \to 0$ converge faster to the asymptotic value than for $\alpha \to \infty$; this is intuitively correct since the central largest velocity part of the wide channel resembles better a set of infinite planes than does the top largest velocity part of the deep channel where a wall is nearby present.

The Gaussian shape

What is actually meant by 'Gaussian' is that the channel has some kind of bell shape, and the first such shape that comes to mind is the Gaussian, which is described solely in terms of a width σ and a depth H, i.e. $He^{-x^2/2\sigma^2}$. However the true Gaussian has tails that, though small, continue infinitely. And that is not what is produced experimentally. Therefore a cut-off is introduced, chosen, arbitrarily, at a distance of 2σ and thus the channel has a total width of 4σ . The bell is further renormalized to a depth of H, that is

$$h(x) = H \frac{e^{-x^2/2\sigma^2} - e^{-2}}{1 - e^{-2}}.$$
(2.36)

and the channel interior is described by $-2\sigma < x < 2\sigma$ and 0 < y < h(x). Fig. 2.5 shows an example of a finite element computational mesh automatically generated in MATLAB together with a contour plot of the solution for the velocity profile in the channel.

When the data for Fig. 2.2 and 2.3 were generated computational meshes of equivalent finesse were used, that is, the elements were approximately even-length sided and the element size was chosen small enough that the meshes had at least 8 elements across the smaller of the channel depth and half width. The maximal number of elements required was 8092.





Figure 2.5: a) Finite element mesh in Gaussian shaped channel of width W = 1.0and depth H = 0.8. Because of mirror symmetry only half of the channel is included in the computation. The computation employs quadratic interpolation, that is, within each triangle element the solution is described as a quadratic polynomial in x and y, see Chap. 3. b) Contour plot of the velocity profile. The maximal velocity is $v_{z,\text{max}} = 2.78 \times 10^{-2} G/\mu$ and the spacing of the contour lines $\Delta v_z = 2 \times 10^{-3} G/\mu$.

Chapter 3

The finite element method

In this chapter we introduce the finite element method (FEM) for solving partial differential equations. The method is a popular tool for simulation of problems in many branches of physics and engineering, in particular in structural mechanics and stress analysis, for which it was originally developed. Indeed many textbooks introduce FEM entirely within the framework of structural mechanics and discuss in detail such concepts as element stiffness and deformability properties. We rather consider the method as a generic device for handling partial differential equations, though we touch only briefly on the error estimates and convergence analysis. Also we discuss the actual implementation of the method in MATLAB.

3.1 Discretization

We consider a problem described in terms of a partial differential equation of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{3.1}$$

where \mathcal{L} is a differential operator describing the physical behaviour of the system, f is the source or forcing term, while Ω is the computational domain on which we are seeking the solution u. Further a set of boundary conditions is provided for u on the boundary $\partial \Omega$ in order to describe how the system interacts with the environment in which it is embedded; or in mathematical terms to ensure that the problem is well-posed.

The finite element method as we present it in this chapter can be applied for problems with differential operators of any order. However what we have in mind is typically a second order differential equation since those are the ones encountered most often in physics and engineering. Indeed our favourite model problem is the Poisson equation

$$-\nabla^2 u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega$$
(3.2)

which is supplemented by either Dirichlet boundary conditions where the solution is prescribed on the boundary $u = a(\mathbf{x})$ for $\mathbf{x} \in \partial \Omega$ or Neumann boundary conditions where the outward normal derivative of u is specified $(\mathbf{n} \cdot \nabla) u = b(\mathbf{x})$ for $\mathbf{x} \in \partial \Omega$ or a combination of the two.

The problem as stated involves an infinite number of points $\boldsymbol{x} \in \Omega$ which makes it somewhat impractical to handle on a digital computer. Thus we need to discretize it, and perhaps the most obvious way to do this is to discretize space. That is, we could for example introduce a grid of points $\boldsymbol{x}_{i,j}$ as shown in Fig. 3.1 and represent the solution by its values at the grid points $u_{i,j}$. Also the differential equation could be discretized using so-called finite difference schemes, e.g. for the Laplace operator ∇^2 on a square grid,

$$\left[\boldsymbol{\nabla}^{2} u\right]_{i,j} \simeq \frac{1}{h^{2}} \left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} \right), \qquad (3.3)$$

where h is the grid point spacing. Thus we would obtain a set of algebraic equations to solve for the unknowns $u_{i,j}$.



Figure 3.1: The finite difference method. A computational grid showing a grid point $x_{i,j}$ (•) and neighbors (•).

Figure 3.2: The finite element method. Division of the domain Ω into a mesh of finite elements; showing a piecewise linear function $\varphi_j(\boldsymbol{x})$ with compact support.

In the finite element method the discretization takes a different route in that one represents the solution as a linear combination of some basis functions φ_{ℓ} defined on the whole domain Ω

$$u(\boldsymbol{x}) = \sum_{\ell} u_{\ell} \varphi_{\ell}(\boldsymbol{x}). \tag{3.4}$$

The discretization is then introduced by choosing only a finite set of basis functions φ_{ℓ} , rather than a complete and therefore infinite set.

The question is now what kind of functions to choose for the basis. A well known choice is Fourier expansion using plane waves with the characteristics that they are infinitely smooth and mutually orthogonal, and that they extend across the entire domain.

The particular choice of the finite element method is somewhat opposite as one chooses basis functions φ_{ℓ} that are local, in the sense that each φ_{ℓ} have
a rather small support. Moreover one typically chooses basis functions that are only as smooth as what is required as minimum for u to be considered as a candidate solution to the problem.

As an example, Fig. 3.2 shows the partitioning of a domain into a mesh of small triangular elements together with a pyramid-like function with the following properties: 1) it is continuous, 2) it is linear inside each element, and 3) it has value 1 at the mesh node x_j while it is zero at all other mesh nodes. By construction this hat-like *chapeau* function has a very small support since it is nonzero only within the elements immediately surrounding the *j*'th node. Similarly one can construct functions centered on the other nodes of the mesh.

These localized functions can be thought of as smeared out delta functions and we observe that they actually constitute a basis for the space of continuous functions that are linear within each element. Also in Eq. (3.4) the expansion coefficients u_{ℓ} are nothing but the value of u at \boldsymbol{x}_{ℓ} , that is, the expansion with the chapeau functions corresponds to an *elementwise linear interpolation* of the function values u_{ℓ} .

3.2 Weak solutions

Considering the problem of Eq. (3.1) the classical way of defining a solution is to require that it satisfies the equality $\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and further that it fulfills the boundary conditions. However, whereas for initial value problems for ordinary differential equations the existence of a unique classical solution is ensured if only the problem satisfies the so-called Lipschitz condition, no such condition exist for boundary value problems for partial differential equations [5]. But, depending on the operator \mathcal{L} it *is* actually possible to prove the existence and uniqueness of a so-called *weak* solution to the problem, a concept that we shall now introduce.

Suppose \mathbb{H} is an infinite-dimensional function space that is rich enough to include in its closure all functions that may be of interest as candidates for the solution u. For all $v \in \mathbb{H}$ we define the defect $d(v) = \mathcal{L}v - f$. Then we say that w is a *weak solution* to the problem Eq. (3.1) provided that the defect d(w) is orthogonal to the entire function space, that is, if

$$\langle v, d(w) \rangle = 0 \quad \text{for every } v \in \mathbb{H}.$$
 (3.5)

Here $\langle \cdot, \cdot \rangle$ denotes the Euclidian inner product in \mathbb{H} defined as $\langle f, g \rangle = \int_{\Omega} d\boldsymbol{x} f(\boldsymbol{x}) g(\boldsymbol{x})$ for functions $f, g \in \mathbb{H}$. Also the inner product naturally induces the norm $\|f\| = \sqrt{\langle f, f \rangle}$.

Obviously if u is a solution to the problem in the classical sense then the defect is the zero function and thus u is also a weak solution. However the converse is not necessarily true! The objects in a function space \mathbb{H} are not really functions but rather so-called equivalence classes of functions. If the

norm of the difference between two functions $||v_1 - v_2||$ is zero then they are said to belong to the same equivalence class and therefore correspond to the same object in \mathbb{H} . E.g. the functions

$$v_1(\boldsymbol{x}) = 0$$
 and $v_2(\boldsymbol{x}) = \begin{cases} 1 & \text{for } \boldsymbol{x} = 0, \\ 0 & \text{else} \end{cases}$ (3.6)

differ only at a single point which is not enough to make a difference on integral in the norm $\|\cdot\|$. The point here is that we should not fuss about the equation not being exactly satisfied at a few points in Ω .

Now we turn to the question of what functions to include in the space \mathbb{H} , that is, which kind of functions to consider as candidates for the solution u. In the case of a second order differential equation then for u to be a classical solution it should at least be a C^1 function, that is, it should at least have a continuous slope since otherwise the second derivatives would become infinite at the slope discontinuities. However the definition of a weak solution w for e.g. the Poisson problem Eq. (3.2) as $\langle v, -\nabla^2 w - f \rangle = 0$ makes perfect sense even if v and w are only C^0 functions that are continuous and piecewise differentiable since any delta function emerging from the second derivative at a slope discontinuity is immediately caught by the integral in the inner product $\langle \cdot, \cdot \rangle$. Formally we express this in terms of a partial integration as

$$\int_{\Omega} \mathrm{d}\boldsymbol{x} \, v \boldsymbol{\nabla}^2 w = \int_{\partial \Omega} \mathrm{d}\boldsymbol{s} \, v \left(\boldsymbol{n} \cdot \boldsymbol{\nabla} \right) w - \int_{\Omega} \mathrm{d}\boldsymbol{x} \, \boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} w \, . \tag{3.7}$$

Introducing the notation $\langle \boldsymbol{f}; \boldsymbol{g} \rangle = \int_{\Omega} d\boldsymbol{x} \, \boldsymbol{f} \cdot \boldsymbol{g}$ we write the last term as $\langle \boldsymbol{\nabla} v; \boldsymbol{\nabla} w \rangle$ and in order for this area integral to exist the functions v and w need only be continuous and piecewise differentiable. Thus if we express w as a linear combination of chapeau basis functions as that shown in Fig. 3.2 then the slope of w is ill-defined on all the triangle sides in the mesh – but this does not trouble the area integral since the offending triangle sides do not add up to a region of finite area. It *does* trouble the boundary integral in Eq. (3.7) though since the entire boundary is made up from triangle sides meaning that the integrand $(\boldsymbol{n} \cdot \boldsymbol{\nabla})w$ is ill-defined on all $\partial\Omega$.

The way to handle the boundary integral is to invoke the boundary conditions. In the Neumann case we wish the solution to have a specific outward normal derivative $(\boldsymbol{n} \cdot \boldsymbol{\nabla})u = b(\boldsymbol{x})$ whereas the value of u on the boundary is left to be determined. Thus we naturally substitute b in place of $(\boldsymbol{n} \cdot \boldsymbol{\nabla})w$ in the boundary integral over the Neumann part of the boundary to enforce the Neumann condition. In the Dirichlet case we rather wish to specify the solution as $u = a(\boldsymbol{x})$ on the boundary. This is enforced either by setting $u_{\ell} = u(\boldsymbol{x}_{\ell}) = a(\boldsymbol{x}_{\ell})$ where \boldsymbol{x}_{ℓ} is a mesh node on the boundary and u_{ℓ} is the expansion coefficient for the basis function centered on \boldsymbol{x}_{ℓ} assuming we choose a basis such as the chapeau functions that interpolates nodal values. Or we may define a subproblem on the Dirichlet boundary similiar to that defined on the interior of Ω

$$\int_{\partial \Omega_a} \mathrm{d}s \, v \left[u - a(\boldsymbol{x}) \right] = 0 \tag{3.8}$$

where $\partial \Omega_a$ is the Dirichlet part of the boundary and v belongs to a set of one-dimensional basis functions defined on $\partial \Omega_a$. In any case we neglect to solve the problem of Eq. (3.1) on the Dirichlet boundary which is done by excluding from \mathbb{H} all functions v that are non-zero on $\partial \Omega_a$.

We are not sure which of the two strategies for enforing the Dirichlet boundary conditions is better. We feel though that the second is more consistent with the FEM approach to the main problem Eq. (3.1), in particular for systems of coupled equations where we may wish to specify both Dirichlet and Neumann boundary conditions on the same part of the boundary for some combination of the solution components. In the commercial software package FEMLAB the first strategy is applied as default, but the second is also available.

Generally when faced with an *n*'th order differential operator it is possible to reduce the smoothness requirements for the function space \mathbb{H} to roughly half *n* by repeated partial integration. E.g. for problems involving the biharmonic operator ∇^4 the members must be at least C^1 functions rather than C^0 in order for the expression $\langle v, \nabla^4 w \rangle$ in the definition of a weak solution to make sense for $v, w \in \mathbb{H}$.

3.2.1 The Galerkin method

The Galerkin method is a method of discretization starting from the definition of weak solutions. We start out choosing a subspace $\mathbb{H}^h \subset \mathbb{H}$ spanned by n linearly independent functions from \mathbb{H}^1 . Then we determine the expansion coefficients u_ℓ in the discrete solution $u^h = \sum_{\ell=1}^n u_\ell \varphi_\ell$ where $\varphi_\ell \in \mathbb{H}^h$ such that the defect $\mathcal{L}u^h - f$ is orthogonal to \mathbb{H}^h

$$\langle \varphi_k, \mathcal{L}u^h - f \rangle = 0, \quad k = 1, 2, \dots, n.$$
 (3.9)

Thus the problem has been reduced to a system of n equations to solve for the n unknowns u_{ℓ} . Provided the operator \mathcal{L} is linear then so is the system of equations

$$\sum_{\ell=1}^{n} \langle \varphi_k, \mathcal{L}\varphi_\ell \rangle u_\ell = \langle \varphi_k, f \rangle, \quad k = 1, 2, \dots, n,$$
(3.10)

or in matrix notation simply

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{3.11}$$

¹The label h on the space \mathbb{H}^h and the discrete solution $u^h \in \mathbb{H}^h$ is meant to refer to the typical size of the elements in the mesh used to construct the basis functions.

with the matrix elements $\mathbf{K}_{k\ell} = \langle \varphi_k, \mathcal{L} \varphi_\ell \rangle$ and right hand side $\mathbf{f}_k = \langle \varphi_k, f \rangle$. Often **K** is termed the *stiffness matrix*.

In order to obtain an accurate solution it is often necessary to choose a rather large basis set, that is large n. This means that the stiffness matrix being $n \times n$ becomes huge! However, it is also *sparse* as the individual matrix element $\mathbf{K}_{k\ell}$ is computed as an integral across Ω of the overlap between φ_k and $\mathcal{L}\varphi_\ell$. Since φ_k is zero outside the elements immediately surrounding the node \mathbf{x}_k and similarly for φ_ℓ we deduce that unless the k'th and ℓ 'th node are actually immediate neighbors then the overlap is zero. In a triangulated mesh a typical node has about six neighbors which means that in the k'th row of \mathbf{K} only about seven matrix elements including \mathbf{K}_{kk} are nonzero.

When implementing the finite element method it is important to exploit sparsity. First, the memory required to store the full matrix including all the zero entries might easily exceed the physical memory of an ordinary workstation even for moderate accuracy in a complex system. Second there are special solution techniques available for sparse matrices that reduce the number of arithmetic operations required to solve the linear system from $\mathcal{O}(n^3)$ for straight-forward Gaussian elimination to something much lower. These include both direct methods where the variables and equations are reordered before the Gaussian elimination to reduce the bandwidth of the matrix, and iterative methods such as the conjugate gradients method and the multigrid method of which the latter is unique in that it can find the solution to the system in $\mathcal{O}(n)$ operations. We shall return to this subject in Sec. 3.5.4.

Finally we note that the Galerkin method is not special to the finite element method but is also used e.g. in spectral methods.

3.2.2 The Lax-Milgram theorem

The Lax-Milgram theorem is important in FEM since it states the existence and uniqueness of a solution for a large class of problems of interest.

Before we can state the theorem we need to introduce a new norm $\|\cdot\|_{H^m}$ in our function space \mathbb{H} , in addition to the L^2 norm $\|\cdot\|$ that we already discussed. The new norm is called a *Sobolev norm* and it measures both the function and its derivatives up to some order m

$$\|v\|_{H^m} = \left(\sum_{k=0}^m \left\|\frac{\partial^k v}{\partial \boldsymbol{x}^k}\right\|^2\right)^{1/2}.$$
(3.12)

In the case of a second order differential operator which is our primary interest this reduces to

$$\|v\|_{H^{1}} = \left(\|v\|^{2} + \|\nabla v\|^{2}\right)^{1/2} = \left(\langle v, v \rangle + \langle \nabla v; \nabla v \rangle\right)^{1/2}.$$
 (3.13)

With this definition we can be more explicit about what functions we want to include in the function space \mathbb{H} . If we are solving a problem with a differential operator \mathcal{L} that involves differentiation up to order 2m, then for all $v \in \mathbb{H}$ we must require $||v||_{H^m} < \infty$ in order for the definition of a weak solution in Eq. (3.5) to make sense.

The Lax-Milgram theorem requires the operator \mathcal{L} to be linear and also bounded $|\langle \mathcal{L}v, w \rangle| \leq \beta ||v||_H ||w||_H$ and coercive $\langle \mathcal{L}v, v \rangle \geq \kappa ||v||_H^2$ for some constants β and κ .² It then states that the problem

$$\langle v, \mathcal{L}w - f \rangle = 0, \quad \text{for all } v \in \mathbb{H}$$
 (3.14)

has got a unique solution in \mathbb{H} . Further the Céa lemma states that the solution u^h that is obtained by applying the Galerkin method Eq. (3.9) with some subspace $\mathbb{H}^h \subset \mathbb{H}$ satisfies

$$||u - u^h||_H \le \frac{\beta}{\kappa} \min_{v \in \mathbb{H}^h} ||u - v||_H,$$
 (3.15)

that is, the distance from the true solution u to the discrete solution u^h is no more than a factor of β/κ from the best approximation to u that can be made in \mathbb{H}^h . This is a strong result since it reduces the question of error analysis to a simple question of approximability. In Sec. 3.3 we extend on the idea of the chapeau functions to construct basis functions that are well suited for approximation and in Sec. 3.3.2 we discuss on the basis of Eq. (3.15) the convergence properties of the resulting numerical schemes.

3.3 Finite elements

The device for constructing a basis set in the finite element method is the partitioning of the domain Ω into a set of smaller domains called elements, as indicated in Fig. 3.2. In two dimensions the most obvious choices are either triangular or quadrangular elements. The basis functions are chosen to be polynomial within each element and are then matched at element boundaries to ensure continuity and possibly also higher order smoothness.

The basis is said to be of accuracy p if within each element the basis functions span the space \mathbb{P}_p of polynomials of degree $\leq p$ on that element. Also the basis is said to be of smoothness q if the basis functions globally belong to the $C^{q-1}(\Omega)$ class of functions, that is, if they are q-1 times smoothly differentiable over the entire domain Ω . Thus the chapeau functions discussed in Sec. 3.1 that were observed to span the space of continuous piecewise linears are of accuracy p = 1 since within each element any linear function can be interpolated while their first derivatives jump at element boundaries such that they are only C^0 functions on Ω and q = 1.

²It is not always straightforward to prove coercivity and boundedness. Still our favourite operator $-\nabla^2$ is bounded and coercive for most domains of interest [5].

Terms in bivariate polynomial	$\dim \mathbb{P}_p^2$
1	1
x y	3
x^2 xy y^2	6
$x^3 \ x^2 y \ x y^2 \ y^3$	10
$x^4 \ x^3 y \ x^2 y^2 \ x y^3 \ y^4$	15

Table 3.1: Terms in bivariate polynomials of total degree p zero through four. In general $\dim \mathbb{P}_p^2$ is related to p as $\dim \mathbb{P}_p^2 = \frac{1}{2}(p+1)(p+2)$.

3.3.1 Lagrange elements

We wish to generalize the construct of the chapeau functions to basis functions of higher accuracy p, but stay with smoothness q = 1 only since C^0 bases are well suited for solving second order differential equations. We shall proceed by defining in a coordinate-free way a basis set that spans \mathbb{P}_p inside the individual elements. Further the element basis is constructed in such a way that when the elements are subsequently patched together the element basis functions naturally combine to form C^0 functions on the entire domain Ω .

Triangular elements

First we focus on triangular elements for which we choose the element basis to be simply the Lagrange interpolation polynomials for a set of equispaced interpolation points or nodes in the element.

We recall the concept of Lagrange interpolation in one dimension: given a set of p+1 distinct points x_0, x_1, \ldots, x_p on the *x*-axis, the Lagrange interpolation polynomials $\varphi_k \in \mathbb{P}_p^1$ are defined as

$$\phi_k(x) = \prod_{j \neq k} \frac{x - x_j}{x_k - x_j}, \quad k = 0, 1 \dots, p,$$
(3.16)

where $\phi_k(x_k) = 1$ while $\phi_k(x_j) = 0$ for all $j \neq k$. Then for a set of function values $f_0, f_1 \dots, f_p$ there exist a unique polynomial $\tilde{f}(x)$ of degree p that interpolates the values f_j , that is, such that

$$f(x_j) = f_j \quad \text{for } j = 0, 1..., p,$$
 (3.17)

and this polynomial can be written explicitly as $\tilde{f}(x) = \sum_j \phi_j(x) f_j$. It is obvious that all p + 1 Lagrange polynomials are linearly independent and thus that they provide a basis for the space \mathbb{P}_p^1 of all polynomials of degree $\leq p$ in one variable since dim $\mathbb{P}_p^1 = p + 1$.

In two dimensions we find that the maximal number of terms in a general polynomial of degree p in x and y grows to dim $\mathbb{P}_p^2 = \frac{1}{2}(p+1)(p+2)$; this is

indicated in Tab. 3.1 showing the various terms in bivariate polynomials of increasing degree. We obtain an element basis for \mathbb{P}_p^2 simply by introducing dim \mathbb{P}_p^2 nodes in the element and require that each basis function be zero at all element nodes except one.

Fig. 3.3 shows the standard interpolation patterns employed for linear, quadratic, and cubic interpolation, that is, accuracy p = 1, 2, and 3 respectively. Notice that we place p + 1 nodes on each element edge with a distribution that is symmetric about the edge midpoint. This ensures that when the elements are patched together, neighboring elements always agree on the position of the p + 1 nodes on their common edge. The edge being a straight line segment the basis functions there behave as univariate polynomials of order p, and thus the interpolation of p + 1 points is unique and continuity is ensured of the global basis functions obtained by combining element basis functions associated with the various nodes.

Quadrangular elements

Turning to quadrangular elements we find that it is not possible to define basis functions that are globally continuous and e.g. linear within each element. However it *is* possible to construct a basis set with basis functions that are uniquely determined by being linear on all four element edges and zero at all element corner nodes except one where they take on value 1. This is called bilinear interpolation and an example is shown in the top row of Fig. 3.4. Similiarly biquadratic and bicubic element basis functions can be defined. Generally the quadrangle Lagrange element basis with *p*'th degree interpolation contains $(p + 1)^2$ basis functions whereas we saw that $\dim \mathbb{P}_p^2 = \frac{1}{2}(p+1)(p+2)$. Thus the basis functions span a space of higher dimension than \mathbb{P}_p^2 . This does not necessarily mean that the basis spans \mathbb{P}_p^2 – but it actually *does*.

3.3.2 Convergence properties

We now return to the Céa lemma Eq. (3.15) which states that the error of the FEM solution is bounded in terms of the best approximation that can be made to the true solution from \mathbb{H}^h . If we consider for the moment only a single element Ω_{α} with Lagrange element basis functions of accuracy order p, then a good approximation to u could be obtained as a p'th degree Taylor polynomial $u_{\text{taylor},p}$. Then by Taylors remainder formula, for all points inside the element the error would be bounded by

$$|u - u_{\text{taylor},p}| \le \frac{1}{(p+1)!} |u^{(p+1)}| h^{p+1}$$
(3.18)

where $u^{(p+1)}$ is the p+1'th derivative of u for some point inside the element and h is the diameter of the smallest circumscribed circle. Now, we cannot



Figure 3.3: Interpolation patterns on the triangle and examples of different types of Lagrange element basis functions. First row corresponds to linear interpolation, second row to quadratic, and third row to cubic, with accuracy p = 1, 2, and 3 respectively. First column shows the positions of the element nodes. The element basis functions are determined such that they are zero at all element nodes except one where they take on the value 1. Column two to four shows examples of element basis functions that are associated with an element corner node (vertex), egde node, and interior node respectively. When the individual element is pathced together with its neighbors to form global basis functions in Ω , element basis functions associated with shared nodes are combined to ensure C^0 smoothness. Thus interior node type basis functions are supported only within a single element, whereas edge node type basis functions are supported in the two elements that share the common edge, and vertex node type basis functions are supported in all elements that share the common vertex. Notice that while the resulting global basis is continuous the basis functions will generally have jumps in the derivative normal to element boundaries.



Figure 3.4: Bilinear (first row), biquadratic (second row) and bicubic (third row) interpolation patterns on the quadrangle and examples of different types of Lagrange element basis functions. The corresponding accuracies are p = 1, 2, and 3, respectively.

actually choose the approximation that freely since it has to be continuous across element boundaries. However a similiar result holds for a p'th degree interpolation polynomial $u_{\text{interp},p}$ over a given set of points. Then for each element

$$\|u - u_{\text{interp},p}\|_{H^i} \le Ch^{p+1-i} \|u^{(p+1)}\|$$
(3.19)

for $0 \leq i \leq p+1$, where *C* is some constant [6]. For a second order operator we measure the error in the H^1 norm and then the interpolation polynomial approximates *u* at least as good as $Ch^p ||u^{(p+1)}||$. According to the Céa lemma the error of the FEM solution is no more than a factor of β/κ times this value. Moreover it is typically found that if the error is measured in the L^2 norm instead, another power of *h* is gained reaching $||u - u^h|| \propto h^{p+1}$.

Equipped with this result we can discuss two distinct routes towards an accurate FEM solution – namely the h-method and the p-method. In the h-method one successively refines the elements to obtain smaller h and a more accurate solution. Refining the mesh by dividing every element into four smaller element and thereby halfing h is called regular mesh refinement.

Notice however that it would be good policy to refine the mesh more where $u^{(p+1)}$ is large; estimation of where this occurs and subsequent refinement at these locations is termed adaptive mesh refinement. There exist several techniques for estimating the error – and some simpler ones that do not estimate the error but only indicate in which elements it seems to be larger than in the others. We have implemented adaptive mesh refinement using a particularly simple such error indicator proposed in [7] for the Poisson equation $-\nabla^2 u = f$. A set of element error indicators e_{α} are computed as

$$e_{\alpha} = \left(|\Omega_{\alpha}| \| \boldsymbol{\nabla}^2 u^h + f \|_{\Omega_{\alpha}}^2 + \sum_{\Gamma \in \partial \Omega_{\alpha}} |\Gamma| \| \lfloor \partial_n u^h \rfloor \|_{\Gamma}^2 \right)^{1/2}$$
(3.20)

where Γ denotes the sides of the element Ω_{α} , $|\Gamma|$ is the side length, $|\Omega_{\alpha}|$ is the element area, and the symbol $\lfloor \partial_n u^h \rfloor$ indicates the jump in normal derivative for the FEM solution u^h from this element to its neighbours. The idea is simple: if either the solution does not satisfy the governing equation in the interior of the element or if it has jumps in the slope from one element to another, then the local resolution is not sufficient and the mesh is refined.

In the *p*-method the mesh is refined not by decreasing the mesh size h but by increasing the order of accuracy p. Notice that the number of nodes in the final mesh is the same after decreasing the mesh size from h to h/a as after increasing the accuracy from p to p + a, c.f. Fig. 3.3. However in the *h*-method the error would drop a factor of a^{-p} whereas in the *p*-method it would drop a factor of h^{-a} , that is, exponentially fast! There is a prize to be payed though – in the *h*-method only few basis functions overlap as they have small support thus the system matrices are sparse, whereas in the *p*-method all the basis functions for the nodes inside a particular element overlap leading to rather dense matrices which makes the solution more

expensive. Further the *p*-method fails if the solution has a singularity in one of its higher derivatives, since then the error bound becomes infinite. This being said, there exist methods for shielding of singularities by surrounding it with one or a few layers of small elements, with the effect that the bulk of the solution remains accurate [7]. We have not investigated these matters further, though.

3.4 Quadrature

In order to solve the discretized problem Eq. (3.9) obtained with the Galerkin method we need to evaluate the inner products

$$\langle \varphi_k, \mathcal{L}u - f \rangle = \int_{\Omega} \mathrm{d}\boldsymbol{x} \, \varphi_k(\boldsymbol{x}) [\mathcal{L}u(\boldsymbol{x}) - f(\boldsymbol{x})].$$
 (3.21)

Since the basis functions are defined to be piecewise linear or generally piecewise polynomial within each element it is natural to divide the integral into the sum of the contributions from the individual elements Ω_{α}

$$\int_{\Omega} \mathrm{d}\boldsymbol{x} \,\varphi_k[\mathcal{L}\boldsymbol{u} - \boldsymbol{f}] = \sum_{\alpha} \int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \,\varphi_k[\mathcal{L}\boldsymbol{u} - \boldsymbol{f}]. \tag{3.22}$$

Here the α sum needs only range over the elements in the support of φ_k since it is zero elsewhere.

The evaluation of the integrals is further simplified by mapping the arbitrary triangular or quadrangular integration domains in the *xy*-plane to a simple reference right-angled triangle or square element in the integration coordinate $\xi\eta$ -plane, see Fig. 3.4. Given a coordinate transform $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$ with $\boldsymbol{\xi} = (\xi, \eta)$ that maps the reference domain \mathcal{A} onto Ω_{α} , then integrals over Ω_{α} transform simply as

$$\int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \, g(\boldsymbol{x}) = \int_{\mathcal{A}} \mathrm{d}\boldsymbol{\xi} \, g\left(\boldsymbol{x}(\boldsymbol{\xi})\right) \left| \boldsymbol{J}(\boldsymbol{\xi}) \right|, \tag{3.23}$$

where |J| is the determinant of the Jacobian matrix $J = \begin{bmatrix} \frac{\partial x}{\partial \xi} \end{bmatrix}$ for the coordinate transform $x(\xi)$.

The triangular reference element is chosen as the right-angled triangle domain

$$\mathcal{A} = \left\{ \boldsymbol{\xi} \mid 0 < \boldsymbol{\xi} < 1, \ 0 < \eta < 1 - \boldsymbol{\xi} \right\}.$$
(3.24)

The simplest coordinate transform that maps \mathcal{A} onto an arbitrary triangle is obtained using the reference element linear basis functions that are simply

$$\chi_1(\xi,\eta) = 1 - \xi - \eta \tag{3.25a}$$

$$\chi_2(\xi,\eta) = \xi \tag{3.25b}$$

$$\chi_3(\xi,\eta) = \eta. \tag{3.25c}$$



Figure 3.5: Mapping of arbitrary triangular and quadrangular elements to the reference elements \mathcal{A} in the integration coordinate $\xi\eta$ -plane. We always do the numbering [1, 2, 3 and 4] of the element corner nodes in counterclockwise order.

With this definition we can take

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{i=1}^{3} \boldsymbol{x}_i \, \chi_i, \qquad (3.26)$$

where $\boldsymbol{x}_i = (x_i, y_i)$ is the *i*'th corner node in Ω_{α} . Since the transform is linear, it maps the reference element edges to straight lines in the *xy*-plane connecting the points $\boldsymbol{x}_1, \boldsymbol{x}_2$ and \boldsymbol{x}_3 , and it is easily verified that $\Omega_{\alpha} = \boldsymbol{x}(\mathcal{A})$.

The quadrangular reference element is chosen as the square domain

$$\mathcal{A} = \left\{ \boldsymbol{\xi} \mid -1 < \boldsymbol{\xi} < 1, \ -1 < \eta < 1 \right\}.$$
(3.27)

Introducing the bilinear element basis functions on the reference element

$$\chi_1(\xi,\eta) = (1-\xi)(1-\eta)/4 \tag{3.28a}$$

$$\chi_2(\xi,\eta) = (1+\xi)(1-\eta)/4$$
 (3.28b)

$$\chi_3(\xi,\eta) = (1+\xi)(1+\eta)/4 \tag{3.28c}$$

$$\chi_4(\xi,\eta) = (1-\xi)(1+\eta)/4,$$
 (3.28d)

the coordinate transform is taken as

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{i=1}^{4} \boldsymbol{x}_i \, \chi_i. \tag{3.29}$$

Since the transform is linear along vertical and horizontal lines in the $\xi\eta$ plane we find again that the reference element edges are mapped to straight lines in the *xy*-plane connecting the element corners \boldsymbol{x}_1 , \boldsymbol{x}_2 , \boldsymbol{x}_3 and \boldsymbol{x}_4 , and further that $\Omega_{\alpha} = \boldsymbol{x}(\mathcal{A})$ provided the corner nodes \boldsymbol{x}_i are numbered in counterclockwise order.

Notice that the coordinate transforms introduced map the nodes of an equispaced interpolation pattern in \mathcal{A} to the correct corresponding pattern in Ω_{α} . Therefore if $\varphi_k(\boldsymbol{x})$ is the Lagrange element basis function associated with \boldsymbol{x}_k in Ω_{α} , and $\phi_k(\boldsymbol{\xi})$ is that associated with $\boldsymbol{\xi}_k$ in \mathcal{A} and $\boldsymbol{x}_k = \boldsymbol{x}(\boldsymbol{\xi}_k)$, then

$$\varphi_k(\boldsymbol{x}(\boldsymbol{\xi})) = \phi_k(\boldsymbol{\xi}). \tag{3.30}$$

Or, employing the inverse transform $\boldsymbol{\xi}(\boldsymbol{x})$

$$\varphi_k(\boldsymbol{x}) = \phi_k(\boldsymbol{\xi}(\boldsymbol{x})). \tag{3.31}$$

These are important relations: in order to evaluate $\varphi_k(\boldsymbol{x})$ with $\boldsymbol{x} \in \Omega_{\alpha}$ we need not know the polynomial coefficients for φ_k on Ω_{α} – rather we can simply map from Ω_{α} to \mathcal{A} and employ an expression for ϕ_k that is common to all elements. Moreover it is in fact $\varphi_k(\boldsymbol{x}(\boldsymbol{\xi})) = \phi_k(\boldsymbol{\xi})$ that enters Eq. (3.23); this is nice in particular for quadrangular elements since for those $\boldsymbol{x}(\boldsymbol{\xi})$ is non-linear and the construction of the inverse transform $\boldsymbol{\xi}(\boldsymbol{x})$ is not simply straightforward.

The integrand in Eq. (3.23) often depends on the gradient of φ_k , which is not directly available but needs to be computed from

$$\frac{\partial \varphi_k(\boldsymbol{x})}{\partial x} = \frac{\partial \phi_k}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial \phi_k}{\partial \eta} \frac{\partial \eta}{\partial x} , \qquad (3.32)$$

and similarly for $\partial \varphi_k / \partial y$. The terms $\partial \xi / \partial x$ and $\partial \eta / \partial x$ are recognized as elements in the inverse of the Jacobian matrix introduced above

$$\boldsymbol{J}^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{|\boldsymbol{J}|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{bmatrix},$$
(3.33)

from which we obtain the gradient

$$\begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}.$$
 (3.34)

For second order differential operators we generally apply partial integration to the second order terms to reduce the smoothness requirements on the basis, in which case the integrand in Eq. (3.21) does not depend on the curvature $\nabla^2 \varphi_k$ of the basis functions. However the element local error indicators introduced in Sec. 3.3.2 for adaptive mesh refinement *do* depend on curvature. The formulas required to compute $\nabla^2 \varphi_k$ are somewhat more complex than Eq. (3.34) and they are presented in Appendix A.

Finally we mention that if a set of higher order reference element basis functions are applied for the coordinate transform than the linear or bilinear ones discussed above then we can map the reference element \mathcal{A} onto a triangle or quadrangle element with curved sides. E.g. for a triangle element we could employ quadratic basis functions for the coordinate transform – then in addition to the triangle corner nodes we should decide on the position of some point on each of the triangles sides. The quadratic transform then maps the regular straightsided reference domain \mathcal{A} onto a triangular domain in the xy-plane whose sides are quadratic curves that interpolate both the side endpoints and the extra point along the side. This feature is important in particular for problems where the boundary of the domain Ω is curved. If high order basis functions are used to represent the solution while the boundary of the computational domain is approximated as being piecewise linear then the solution accuracy may degrade close to the boundary. Better results are generally obtained using the same basis functions for both the coordinate transform and the representation of the solution, in which case on speaks of *isoparametric* elements. However, in most of the problems that we have been considering we have found that it was sufficient to use the simple coordinate transforms.

3.4.1 Analytical expressions

In this section we include some simple formulas for element integrals that can be performed analytically. The formulas are instructive and also straightforward to implement in a computer code as we demonstrate in a short example. Thus if the theory and the concepts introduced so far seems complicated and difficult to apply in practice, we wish to show that it is actually not!

We consider the simple case of triangular elements with straight sides so that we can employ the linear coordinate transform in Eqs. (3.25). Then the Jacobian matrix is constant such that the determinant $|\mathbf{J}|$ can be moved outside the integral in Eq. (3.23). Additionally we assume that the integrand $g(\mathbf{x}(\boldsymbol{\xi}))$ does not depend explicitly on x and y – this is the case when it depends only on the basis functions as stated in Eq. (3.30). Finally we choose the basis functions φ_k to be piecewise linear since in this case the results are particularly simple.

By differentiation of Eq. (3.25) with respect to ξ and η we obtain the Jacobian

$$\boldsymbol{J} = \begin{bmatrix} \Delta x_{12} & \Delta x_{13} \\ \Delta y_{12} & \Delta y_{13} \end{bmatrix}$$
(3.35)

where $\Delta x_{ij} = x_j - x_i$ and similarly for y. We compute the area of an element Ω_{α} by setting g = 1 in Eq. (3.23) and obtain

$$A_{\alpha} = \int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} = \int_{\mathcal{A}} \mathrm{d}\boldsymbol{\xi} \, |\boldsymbol{J}| = \frac{1}{2} \big[\Delta x_{12} \Delta y_{13} - \Delta x_{13} \Delta y_{12} \big]. \tag{3.36}$$

The volume below a basis function is found to be

$$\int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \,\varphi_k(\boldsymbol{x}) = \frac{1}{3} A_{\alpha} \tag{3.37}$$

which should be well-known from solid geometry. Since we have chosen basis functions that are piecewise linear, their gradient is constant within the element, and we compute it using Eq. (3.34). Then the contribution to the

stiffness matrix $\mathbf{K}_{k\ell} = \langle \nabla \varphi_k; \nabla \varphi_\ell \rangle$ that we obtain for the negative Laplace operator $-\nabla^2$ after integration by parts is from the particular element Ω_{α}

$$\left[\mathbf{K}_{k\ell}\right]_{\Omega_{\alpha}} = \int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \, \boldsymbol{\nabla}\varphi_k \cdot \boldsymbol{\nabla}\varphi_\ell = A_{\alpha} \, \boldsymbol{\nabla}\varphi_k \cdot \boldsymbol{\nabla}\varphi_\ell \,. \tag{3.38}$$

Another operator encountered frequently is the unit operator 1. Projection onto φ_k yields

$$\langle \varphi_k, 1u \rangle = \langle \varphi_k, u \rangle = \sum_{\ell} \langle \varphi_k, \varphi_\ell \rangle u_\ell$$
 (3.39)

for $u = \sum_{\ell} u_{\ell} \varphi_{\ell}$. The matrix with elements $\langle \varphi_k, \varphi_\ell \rangle$ is often termed the mass matrix **M** and for piecewise linear basis functions the elemenwise contributions are

$$\left[\mathbf{M}_{k\ell}\right]_{\Omega_{\alpha}} = \int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \,\varphi_k \,\varphi_\ell = \frac{1}{12} \left(1 + \delta_{k\ell}\right) A_{\alpha}. \tag{3.40}$$

where $\delta_{k\ell}$ is the Kronecker delta. Considering more generally the operation of multiplication by a scalar function $a(\boldsymbol{x})$ we can approximate it by replacing a with a piecewise linear approximation $a = \sum_{m} a_m \varphi_m$. Then the elementwise contributions to the matrix representing this operator turn into

$$\int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \,\varphi_k \, a(\boldsymbol{x}) \varphi_\ell \simeq \int_{\Omega_{\alpha}} \mathrm{d}\boldsymbol{x} \,\varphi_k \left[\sum_m a_m \varphi_m \right] \varphi_\ell = \left[a_k + a_\ell + \sum_m a_m + \delta_{k\ell} \left(2a_\ell + \sum_m a_m \right) \right] A_{\alpha}/60 \,. \quad (3.41)$$

This expression was obtained using the following general result for integrals on the reference element

$$\int_0^1 d\xi \int_0^{1-\xi} d\eta \left[\xi^\alpha \eta^\beta (1-\xi-\eta)^\gamma \right] = \frac{\Gamma(\alpha+1)\Gamma(\beta+1)\Gamma(\gamma+1)}{\Gamma(\alpha+\beta+\gamma+3)}$$
(3.42)

where α , β and γ are arbitrary powers greater than -1 and $\Gamma(x)$ is the Gamma function; notice that $\Gamma(n+1) = n!$ for integer n.

We now proceed to show how the above formulas for the element integrals can be almost directly translated into a program in MATLAB or any other programming language. In order to do this we consider a model problem

$$-\nabla^2 u + a(\boldsymbol{x})u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega$$
(3.43)

where we shall approximate a and f as being piecewise linear. Discretizing with the Galerkin method and performing partial integration for the Laplace operator we obtain a system of linear equations

$$\mathbf{K}\mathbf{u} + \mathbf{M}_a \mathbf{u} = \mathbf{f} + \mathbf{b}.\mathbf{t}. \tag{3.44}$$

Example 3.1

```
\% Assemble matrices K and Ma and right hand side F for the problem
% -nabla<sup>2</sup> u + a(x) u = f(x)
\% using piecewise linear basis functions. Finite element mesh points
\% and triangles are given as the variables p and t. Expansion
\% coefficients for a(x) and f(x) are given as variables a and f.
K = sparse(length(p),length(p));
Ma = sparse(length(p),length(p));
F = zeros(length(p),1);
% loop elements in mesh
for m = 1:length(t)
  % global indices of element corners
  n = t(1:3,m);
   % Jacobian matrix
   J = [p(1,n(2))-p(1,n(1)) p(1,n(3))-p(1,n(1))
        p(2,n(2))-p(2,n(1)) p(2,n(3))-p(2,n(1))];
   detJ = J(1,1)*J(2,2) - J(1,2)*J(2,1);
   % element area
   A = detJ/2;
   % derivatives of basis functions
   phix = [J(2,1)-J(2,2); J(2,2); -J(2,1)]/detJ;
   phiy = [J(1,2)-J(1,1); -J(1,2); J(1,1)]/detJ;
   % loop nodes in element
   for i = 1:3
      for j = 1:3
         K(n(i),n(j)) = K(n(i),n(j)) + A*(phix(i)*phix(j) + phiy(i)*phiy(j));
         Ma(n(i),n(j)) = Ma(n(i),n(j)) + A*(a(n(i)) + a(n(j)) + sum(a(n)))/60;
      end
      Ma(n(i),n(i)) = Ma(n(i),n(i)) + A*(2*a(n(i)) + sum(a(n)))/60;
      F(n(i)) = F(n(i)) + A*(f(n(i)) + sum(f(n)))/12;
   end
end
```

where the boundary term b.t. arises from the partial integration. In the Example 3.1 it is assumed that we have already obtained a finite element mesh covering Ω . The data structure that we use for the mesh is described in detail in Sec. 3.5.1; in short we make use of a 2 by n_p table of mesh vertices or points **p** such that the x and y coordinates of the k'th point x_k are stored in **p**(1,k) and **p**(2,k) respectively. Also we use a 3 by n_t table of mesh triangles **t** such that **t**(1,n) contains the index of the point at first corner of the n'th triangle, and **t**(2,n) and **t**(3,n) the indices of the points in the second and third corners. Finally in the example it is assumed that the values of the functions a(x) and f(x) at the mesh points x_k are given

as two n_p by 1 tables **a** and **f**.

For every element the example first computes the Jacobian matrix and its determinant using Eq. (3.35). Next the element area is given by Eq. (3.36) and the derivatives of the element basis functions are obtained from Eq. (3.34) using $\phi_1 = 1 - \xi - \eta$, $\phi_2 = \xi$ and $\phi_3 = \eta$. The elementwise contributions to **K** and **M**_a are implemented directly from Eqs. (3.38) and (3.41).

The example demonstrates an important feature – in stead of looping over the basis functions φ_k , each corresponding to a row in **K**, and summing over contributions from elements Ω_{α} in the support of each particular φ_k as indicated originally in Eq. (3.22) – we might as well loop over the elements and add for each particular Ω_{α} the contributions to all φ_k that contain this Ω_{α} in their support; but these are exactly those basis functions associated with the corner nodes of Ω_{α} .

Finally we note that before the problem can actually be solved we need to modify the equations to account for boundary conditions. For points \boldsymbol{x}_k on the Neumann part of the boundary there is an extra contribution to \mathbf{f}_k from the boundary integral. For points on the Dirichlet boundary we replace the *k*th row in **K** and \mathbf{M}_a by zeros, put a one at the diagonal entry \mathbf{K}_{kk} and replace \mathbf{f}_k by the value of the Dirichlet condition at \boldsymbol{x}_k .

Then the system is ready for solution e.g. using MATLAB's built-in linear solver as $u = (K+Ma)\F$. The algorithm that hides behind the backslash operator is actually quite efficient in solving problems involving large and sparse matrices as we shall discuss in Sec. 3.5.4.

3.4.2 Gauss-Legendre quadrature

While the explicit formulas for the element integrals discussed in the previous section are simple and easy to implement, in some cases it turns out to be preferable to perform the element integrals numerically rather than analytically. Indeed we only treated the case of linear basis functions as for quadratic or higher order basis functions the expressions become increasingly complicated. Also it may not be appropriate to replace the coefficient and source terms like a and f in Example 3.1 with their piecewise linear aproximations for doing the integrals. And for quadrangle elements or triangle elements with curved sides the coordinate transform from the reference element to the actual elements in the xy-plane is non-linear so that the Jacobi-determinant cannot be moved outside the integral which was a premise for the simple results.

However the finite element method readily lends itself to numerical integration as the computational domain Ω , unwieldy as it may be, is already divided into regular elements. Also we discussed how to map those elements onto a completely regular reference element in the integration coordinate space.

Numerical integration or quadrature generally amount to approximating

the definite integral of some function f(x) over an interval [a, b] by evaluating the integrand at a number N of points x_i in [a, b] and adding up $f(x_i)$ with some specific weights W_i

$$\int_{a}^{b} dx f(x) \simeq \sum_{i=1}^{N} W_{i} f(x_{i}).$$
 (3.45)

A familiar example is the midpoint quadrature rule where f is evaluated at N equispaced points x_i in the interval and the same weight $W = \frac{b-a}{N}$ is used for all points. The error on the result obtained is of the order of $|b-a|^3/N^2$.

It turns out that there are more efficient ways of choosing the quadrature points and weights – and the choice that is obtained by optimizing the ability of the formula to integrate polynomial integrands is called a Gaussian quadrature rule. As there are N degrees of freedom associated with the choice of x_i and other N with the choice of W_i and since the Eq. (3.45) is linear in the integrand we hope to be able to determine correctly the integral of all of $1, x, x^2, \ldots x^{2N-1}$ and generally any polynomial of degree $\leq 2N - 1$. The proof by Gauss that this is indeed possible employs the theory of orthogonal polynomials to show that the N quadrature points should be chosen as the roots in the Nth degree Legendre polynomial $P_N(\xi)$ with corresponding weights $W_i = 2(1 - \xi_i^2)/[(N + 1)P_{N+1}(\xi_i)]^2$ – where we assume now that the arbitrary interval $x \in [a, b]$ has been mapped onto the reference interval $\xi \in [-1, 1]$.

In Tab. 3.2 the positions and weights for the first few N are listed. Similiar results are tabulated in [8] up to N = 512 with 30 digits of accuracy. In order to perform element integrals on the quadrangle reference element that we introduced with $(\xi, \eta) \in [-1, 1] \times [-1, 1]$ we simply apply the line integral rule to both ξ and η .

For the triangle reference element matters are somewhat more complicated. Applying the line integral rule directly to the integral in the form $\int_0^1 d\xi \int_0^{1-\xi} d\eta f(\xi,\eta)$ clusters a waste amount of points close to the corner $(\xi,\eta) = (1,0)$ and because of asymmetry the numerical results obtained using such a formula depend on the numbering of the corner nodes, c.f. Fig. 3.4. A direct approach to optimization of the ability to integrate polynomials up some power by adjusting the quadrature point positions ξ_i , η_i and weights W_i is difficult as the problem is highly nonlinear in ξ_i and η_i . Yet this is the approach taken in [9] where symmetric rules are computed with degree of precision up to 20.

Notice that since we have defined the basis functions to be polynomials within each element the numerical integration employing Gaussian quadrature rules is particularly appropriate. When the integrand depends only on the basis functions, e.g. as for the mass matrix $\mathbf{M}_{k\ell} = \langle \varphi_k, \varphi_\ell \rangle$, we are able to choose the number of integration points as the minimum required to yield the *exact* result. However even for integrands that are not polyno-

#points	Positions	Weights	Precision
N	ξ_i	W_i	
1	$\{0\}$	$\{2\}$	1
2	$\left\{\pm\frac{1}{\sqrt{3}}\right\}$	$\{1\}$	3
3	$\{0,\pm\frac{\sqrt{3}}{\sqrt{5}}\}$	$\{\tfrac{8}{9}, \tfrac{5}{9}\}$	5

Gaussian quadrature rules for line integrals over interval $\xi \in [-1, 1]$.

Gaussian quadrature rules for integrals over triangle with $0 < \xi < 1$ and $0 < \eta < 1 - \xi$.

#point	s Positions		Weights	Precision
	ξ_i	η_i	W_i	
1	$\left\{\frac{1}{3}\right\}$	$\left\{\frac{1}{3}\right\}$	$\left\{\frac{1}{2}\right\}$	1
3	$\{\frac{2}{3}, \frac{1}{6}, \frac{1}{6}\}$	$\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\}$	$\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$	2
4	$\left\{\frac{1}{3}, \frac{3}{5}, \frac{1}{5}, \frac{1}{5}\right\}$	$\{\frac{1}{3}, \frac{1}{5}, \frac{3}{5}, \frac{1}{5}\}$	$\{-\frac{27}{48}, \frac{25}{48}, \frac{25}{48}, \frac{25}{48}, \frac{25}{48}\}$	} 3

Table 3.2: Gaussian quadrature rules for line integrals and area integrals on triangular domain. The precision denotes the highest order polynomial for which the rule yields the exact result [8, 9].

mial Gaussian quadrature generally yields very good results which can be attributed to the fact that most functions can be well approximated on a finite interval by some polynomial.³ And even integrands that are singular or undefined at the integration interval endpoints can be handled since the Gaussian quadrature points always lie in the interior of the interval.⁴

We have implemented a routine gquad for elementwise Gaussian quadrature. We describe it in detail in Sec. 3.5 but the basic syntax is as follows

I = gquad(f, #ng, element, p, t, flag)

The first argument f describes the integrand and should be a character string that can be evaluated to a function of x and y, e.g. (2*pi*x)+y'. The second #ng is a number specifying which rule to use; there is a slight inconsistency in that for line and quadrangle elements #ng specifies the number of quadrature points N which yields a precision of 2N - 1 as discussed above – whereas for triangle elements #ng specifies simply the degree of precision. The third argument *element* should be one of the strings 'line', 'triangle' or 'quadrangle' specifying the element type and the fourth and fifth arguments p and t tables containing the mesh points and elements

 $^{{}^{3}}$ E.g. when the coordinate transform is non-linear then the derivatives of the basis functions as obtained from Eq. (3.34) are rational functions rather than simple polynomials; thus they are not integrated exactly by Gaussian quadrature.

⁴This is not true for all of the rules given in [9] where some quadrature points are located slightly outside the triangular reference element.

Example 3.1a

```
\% Assemble matrices K and Ma and right hand side F for the problem
% -nabla<sup>2</sup> u + a(x) u = f(x)
% using piecewise linear basis functions. Finite element mesh points
\% and triangles are given as the variables p and t.
% strings defining the coefficient and source term
a = 'sin(x).*sin(y)';
f = '1 + exp(-y.^2)';
i = [1 2 3 1 2 3 1 2 3]; % matrix row indices in t
j = [1 1 1 2 2 2 3 3 3]; % matrix column indices in t
K = gquad('',1,'triangle',p,t,'dphi.dphi');
                                              % elementwise quadrature
K = sparse(t(i,:),t(j,:),K,length(p),length(p)); % assemble matrix
Ma = gquad(a,5,'triangle',p,t,'phi.phi');
Ma = sparse(t(i,:),t(j,:),Ma,length(p),length(p));
i = [1 2 3]; % vector row indices in t
F = gquad(f,4,'triangle',p,t,'phi');
F = sparse(t(i,:),1,F,length(p),1);
```

Condensed version of Example 3.1 using the gquad routine. For the stiffness and mass matrices gquad returns a table with nine rows and as many columns as there are triangles in the triangle table t. The nine rows correspond to all combinations of the three corners of the triangle, and the variables i and j shows the order followed by gquad e.g. for the integral $\int_{\Omega_{\alpha}} dx \nabla \varphi_i \cdot \nabla \varphi_j$. That is, first three rows in the result correspond to $i = \{1, 2, 3\}$ and j = 1 and rows four to six to $i = \{1, 2, 3\}$ and j = 2 and so forth.

Since we use the default piecewise linear basis functions the gradients are constant within each element so a single quadrature point #ng = 1 is sufficient when computing **K**. The element integrals for \mathbf{M}_a involve the product of two linear basis functions which requires degree of precision #ng = 2; thus choosing #ng = 5 we effectively approximate $a(\mathbf{x}) = \sin(x)\cos(x)$ with a polynomial of degree three within each triangle. Similarly for $f(\mathbf{x})$ in computing **f**.

Example 3.1a executes approximately an order of magnitude faster than the loop version.

as described in detail in Sec. 3.5. Finally the optional *flag* should be one of the strings ', 'phi', 'phi.phi' or 'dphi.dphi' specifying whether the routine should simply perform an element integral $\int_{\Omega_{\alpha}} dx f(x)$ or rather a projection onto the basis functions as $\int_{\Omega_{\alpha}} dx f(x) \varphi_k$, or compute contributions to a mass or stiffness matrix as $\int_{\Omega_{\alpha}} dx f(x) \varphi_k \varphi_\ell$ or $\int_{\Omega_{\alpha}} dx f(x) \nabla \varphi_k \nabla \varphi_\ell$.

Depending on the flag specified, the result produced by gquad is either a single number for the simple integral, a list of the projections onto each basis function φ_k in the element, or a list of projections onto each pair of φ_k and φ_ℓ for the mass or stiffness matrix integrals.

A key feature in gquad is that it allows *vectorized* operation meaning that it allows element integrals of the same type to be performed for several elements at once. This is an important concept when working with MATLAB since performance generally proves to be rather poor for loop structures but quite efficient for vector and matrix operations. Thus we could think of dropping the outer for loop in Example 3.1 and do the arithmetic operations not on single numbers for a particular element but on lists of numbers for the corresponding quantities in all elements in the triangle table t.

Using the gquad routine the operations performed in Example 3.1 are condensed to the form in Example 3.1a. The example assumes the particular form $a(\mathbf{x}) = \sin(x)\sin(y)$ and $f(\mathbf{x}) = 1 + \exp(-y^2)$ for the coefficient and the source term respectively.

3.5 Implementation

In this section we discuss some specific aspects of our implementation of the finite element method in MATLAB. Particular features of our implementation include support for Lagrange element basis functions in one and two dimensions of any order of accuracy, and simple computation of element integrals using the gquad routine described in Sec. 3.4.2. The gquad routine is also useful for postprocessing of data, e.g. computation of total flow through a part of the boundary or total charge in a section of the domain.

Adaptive solution is supported with the routine gjmp that computes the jump in normal derivative across element boundaries, which is required by the error indicator discussed in Sec. 3.3.2. Whereas this task is almost trivial in the case of triangular elements with linear basis functions since the slope of the solution is then constant over each element, it is more involved for higher order basis functions. The error in the interior of each element is computed with gquad.

We have implemented the multigrid method, which is a fast iterative solution strategy for problems with very fine meshes that would be almost impossible to solve within finite time using direct methods. This feature is not used in any of the applications that we treat in the subsequent chapters, but still the implementation of the method has required quite an effort. However, while the method works fine for the Poisson equation, we have had problems using it for flow problems.

Finally, for nonlinear problems we provide a routine **newt** for solving it with Newtons method. In order to use this routine we require the user to implement a function that computes the residual and Jacobian for the problem. In particular for the problem of electroosmotic flow discussed in Chap. 7 we must admit that the implementation of such a function was cumbersome, even if rather straightforward, as the solution contains six components counting the fluid velocity v_x and v_y , pressure p, electrostatic potential ϕ and the concentration of two charged species in the liquid.

We proceed by first explaining the details of the mesh data structure that we have chosen. Next we introduce the syntax for the Lagrange element basis functions and show how this is used with the Gaussian quadrature routine gquad. Then we discuss different strategies for solving the system of linear equations for the discretized problem – either direct methods relying on Gaussian elimination, or iterative methods including the conjugate gradients method and the multigrid method. Finally we explain how to solve a system of non-linear equations by use of Newtons method.

In Appendix C we have listed the headers of all the routines mentioned in this section. Header comments are displayed as help message when e.g. the command **help gquad** is executed in the MATLAB command prompt; thus they are intended to give a quick overview of what the routine does.

3.5.1 Mesh data structure

We have chosen to adopt the mesh data structure of the MATLAB PDE toolbox [10], which is a module for handling elliptic partial differential equations in two dimensions. The toolbox relies solely on triangular elements and linear basis functions, and further it is not well suited for fluid dynamics since the convective term does not fit with the generic elliptic model used in the toolbox. However it provides easy meshing facilities where the geometry is described as a combination of solid objects such as rectangles and circles, or in terms of a user-defined geometry subfunction. Thus we have chosen to rely on this meshing facility.

The data structure consist of a mesh point matrix \mathbf{p} , a triangle matrix \mathbf{t} and a boundary matrix \mathbf{e} . The point matrix is $2 \times n_p$, where n_p is the total number of points, and $\mathbf{p}(1:2,\mathbf{n})$ contains the x and y coordinate of the *n*th point respectively. The triangle matrix is $4 \times n_t$, where n_t is the total number of triangles in the mesh, and $\mathbf{t}(1:3,\mathbf{n})$ contains the indices of the three mesh points spanning the *n*th triangle. The fourth component $\mathbf{t}(4,\mathbf{n})$ contains a number, the so-called subdomain label, which is used to group elements together – e.g. the inside and the outside of a channel as used in Chap. 7. The boundary table is $7 \times n_e$, where $\mathbf{e}(1:2,\mathbf{n})$ contains the indices of the two points spanning the *n*th line element of the boundary. Further $\mathbf{e}(5,\mathbf{n})$ is the boundary segment label, and the rest of the entries in \mathbf{e} are used to define the parametrization of the boundary.

The PDE toolbox only allows the use of triangle elements; however we spoke to several experienced FEM practitioners that preferred using quadrangular elements, since they yield more accurate results. Therefore we decided to add also quadrangular elements to our implementation. Thus we define a quadrangle matrix \mathbf{q} being $5 \times n_q$ and containing the indices of the four corner points in counterclockwise order and a subdomain label.

Further, in order to allow the use of higher order basis functions it is necessary to have knowledge also about the unique lines in the mesh, connecting the mesh points, in order to be able to place edge type nodes on these lines, c.f. Fig. 3.3. Therefore we also define a side matrix \mathbf{s} , containing the unique element sides in the mesh. The side matrix is $2 \times n_s$ with s(1:2,n) containing the indices of the two mesh points connected by the *n*th side. We have implemented a small routine elm2sd, such that s=elm2sd(e,t,q) determines the unique sides in the mesh provided the boundary, triangle and quadrangle matrices. Further a call of the form [s,es,ts,qs]=elm2sd(e,t,q) returns three matrices, each $1 \times n_e$, $3 \times n_t$ and $4 \times n_q$, relating the elements to their unique sides.

The PDE toolbox provides a mesh refiner **refinemesh** that implements both regular and longest-edge mesh refinement, where regular mesh refinement corresponds dividing every element in the mesh into four smaller elements. Also this routine allows that only a specific subset of the triangles be refined. This is important in order to be able to do adaptive solution.

However, in order to accomodate also quadrangular elements, we have implemented a routine halfmesh that allows mixed triangle and quadrangle meshes to be refined. It only supports regular refinement and we wrote it specifically to allow easy computation of the prolongation operators used in the multigrid method.

3.5.2 Basis functions

In order to do numerical integration it is necessary to evaluate the element basis functions on a number of quadrature points ξ_i , η_i inside each element. For the Lagrange element basis, this is accomplished by a function lagrange with the calling syntax, e.g. phi=lagrange(3,'quadrangle',xi,eta) for cubic basis functions on the quadrangular reference element.

If the element basis function, apart from the vertex nodes, contain also m_s nodes on each side and m_t or m_q on the triangle or quadrangle element interiors, then the global solution will contain $n_p+m_s \times n_s+m_t \times n_t+m_q \times n_q$ data items, which are collected in into a long column vector. Enhanced element tables containing the indices in the large solution vector for all nodes in the element are obtained as [ee,tt,qq]=lagrange(3,'table',e,t,q), again for the case of cubic basis functions.

3.5.3 Quadrature revisited

In Sec. 3.4.2 we explained how to use the gquad routine to perform elementwise integrals for an integrand depending on the position x and y. However gquad also allows the integrand to depend on the value of the solution as well as its gradient and curvature. The most general case handled by gquad is an integral of the form

$$\int_{\Omega_{\alpha}} dx dy f(x, y, u, v, w) \varphi_k \psi_\ell$$
(3.46)

where the quantities u, v and w are expansions of the form $u = \sum_{i} u_i \chi_i$, $v = \sum_{j} v_j \varphi_j$ and $w = \sum_{k} w_k \psi_k$, and χ is the basis functions used for the coordinate transform. The syntax is then

I = gquad(f, #ng, element, p, {t,u}, proj, {phi,tt,v}, {psi,ttt,w}) where f is a string expression of the integrand, e.g. 'sin(u)+x.*w'. #ng defines the precision of the quadrature rule to be applied, element is a string defining the element type, p is the point matrix and t is the element matrix – either e, t or q. proj is a string defining the projection to be performed – either an empty string '' for the simple integral, 'phi' or 'dphi' for projection onto φ_k or $\nabla \varphi_k$, or phi.phi, phi.dphi or dphi.dphi for projection onto $\varphi_k \psi_\ell$, $\varphi_k \nabla \psi_\ell$ or $\nabla \varphi_k \nabla \psi_\ell$ respectively. Finally phi and psi are integers defining the Lagrange element order, tt and ttt are enhanced element tables as returned by lagrange and u, v and w are column vectors containing the expansion coefficients of the solution components.

Further gquad exploits symmetry in the case where φ and ψ are identical since then $\int d\mathbf{x} f \varphi_k \varphi_\ell = \int d\mathbf{x} f \varphi_\ell \varphi_k$.

3.5.4 Solution of linear system of equations

The most straightforward approach to solving the systems of linear equations obtained after discretizing the problem is to apply a direct solver. In MATLAB one calls upon the built-in solver by using the backslash operator, as e.g. $\mathbf{x} = \mathbf{A} \setminus \mathbf{b}$, which invokes a method based on Gaussian elimination for solving the system. Whereas Gaussian elimination for an m by m matrix problem is generally an $\mathcal{O}(m^3)$ operation, for a sparse matrix this can often be reduced a great deal. E.g. for matrices that are banded, that is, for which all elements outside the dth diagonal are zero, the cost is reduced to $\mathcal{O}(d^2m)$. The bandwidth of a matrix depends strongly on the ordering of the points in the mesh; however the algorithm behind the MATLAB backslash operator performs a row and column permutation before initiating the solution process in order to make the solution as fast and memory efficient as possible, and thus the ordering chosen is not important, which is very convenient.

For small to medium sized problems the solution via the direct method is preferred over the iterative methods since it is simple and fast. Only for larger problems are the iterative methods more efficient, but then the question of convergence of the iterative scheme is introduced as an extra concern to the problem.

Conjugate gradients method

The conjugate gradients method is an iterative solution method that applies to symmetric and positive definite matrices. The method solves a problem $\mathbf{A}\mathbf{x} = \mathbf{b}$ by minimizing the function $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x}$ as this has unique minimum when $\mathbf{A}\mathbf{x} = \mathbf{b}$. The method only requires the computation of the gradient $\frac{\partial F}{\partial \mathbf{x}}$, which is equal to the residual $\mathbf{A}\mathbf{x} - \mathbf{b}$, which is inexpensive to compute for sparse matrices. We shall not go into detail with the way the method works, but only mention that it is in a sense an extension to the steepest decent method, in which every step consist of minimizing the function F along the steepest decent direction, i.e. the gradient [5, 11]. This is also how the first step in the conjugate gradients method is done, but in the second step the algorithm takes into account both the old and the new steepest decent direction to obtain a better search direction. It can be shown that in every iteration the conjugate gradients method eliminates the error in the initial guess for \mathbf{x} along one of the eigenvectors, and thus that if no round-off errors are present it terminates within a number of iterations equal to the dimension of **A**. This is not particularly fast – however it turns out that by *preconditioning* the set of equations one can obtain convergence much faster. The preconditioning operation amounts to solving, rather than Ax = b, a system of the form $M^{-1}Ax = M^{-1}b$ where M should be a good approximation to \mathbf{A} , but easier to invert! Indeed using $\mathbf{M} = \mathbf{A}$ the method converges in one step, but of course if A^{-1} is available, the problem is already solved. The simplest choise of preconditioner is to use the diagonal of A for M, but this is not very efficient either. Another method that is often cited to yield good convergence is the use of an incomplete LU-factorisation of **A**. MATLAB provides a function to compute such an incomplete factorization, as well as an implementation of the conjugate gradients method and several other iterative methods of the same class. However, we have generally only had little succes with the use of those methods, but prefer either the direct method discussed above or multigrid.

The multigrid method

The multigrid method is based on solving the system of equations on several levels of coarseness simultaneously. The basic idea is to accelerate the convergence of other iterative methods by applying them on several levels. As many iterative methods work by solving the equations locally rather than globally, they tend to converge slowly for very fine meshes since they miss the global view. But applying the same iterative method on a coarser mesh, its view becomes larger and it is now able to reduce also the long wavelength components of the error.

The classical method used in conjunction with the multigrid method is the Gauss-Seidel iterative scheme, that proceeds by solving the *k*th equation $\langle \varphi_k, \mathcal{L}u - f \rangle$ for the *k*th solution component u_k assuming all other variables are fixed. However, the *k*th equation then had better depend on u_k and not too much on the rest of the variables, which is equivalent to requiring that the system matrix should be diagonally dominant. In the case of the Navier-Stokes equations for incompressible flow, this is *not* the case with the incompressibility constraint $\nabla \cdot \boldsymbol{v} = 0$, which is independent of the pressure. Other iterative methods that do not require the system to be diagonally dominant can be applied, but we don't know of any methods that are quite as effective as the Gauss-Seidel method. One of the key features of the Gauss-Seidel method in relation to multigrid is that it is a *smoother*, that is, is can be shown to be very efficient in reducing short wavelength components of the error.

We now sketch the basic working principle of the multigrid method, following [11]. We consider first two mesh levels, a coarse one of typical element size H, and a fine one typical element size h = H/2 obtained by regular mesh refinement of the coarse mesh. Because the fine mesh is constructed from the coarse one, any function on the coarse mesh can be represented also on the fine mesh, that is, $\mathbb{H}^H \subset \mathbb{H}^h$. In particular, the coarse mesh basis functions can be expressed as a linear combination of the fine mesh basis functions, $\varphi_{\ell}^H(\boldsymbol{x}) = \sum_k \mathcal{P}_{k\ell} \varphi_k^h(\boldsymbol{x})$. The matrix with elements $\mathcal{P}_{k\ell}$ is called the prolongation operator, and it relates the expansion coefficients for any function expressed on the coarse mesh to the expansion coefficients on the fine mesh, since

$$u^{H}(\boldsymbol{x}) = \sum_{\ell} u^{H}_{\ell} \varphi^{H}_{\ell}(\boldsymbol{x}) = \sum_{\ell} u^{H}_{\ell} \sum_{k} \mathcal{P}_{k\ell} \varphi^{h}_{k}(\boldsymbol{x}) = \sum_{k} u^{h}_{k} \varphi^{h}_{k}(\boldsymbol{x}), \quad (3.47)$$

where $u_k^h = \sum_{\ell} \mathcal{P}_{k\ell} u_{\ell}^H$, or in matrix notation $\mathbf{u}^h = \mathbf{P} \mathbf{u}^H$. For Lagrange element basis functions it is easy to construct the prolongation operator, simply using $\mathcal{P}_{k\ell} = \varphi_{\ell}^H(\boldsymbol{x}_k)$ where \boldsymbol{x}_k is the node associated with the fine mesh basis function φ_k^h .

We wish to solve the problem $\mathcal{L}u = f$ on the fine mesh, and we define u^h as the true solution to the discretized equations $\langle \varphi_k, \mathcal{L}u^h - f \rangle = 0$. Also we have some initial guess \hat{u}^h for the solution, and we apply the iterative method, e.g. Gauss-Seidel, to improve on this guess. However, after a number of iteration, the residual r^h defined by

$$r_k^h = \langle \varphi_k^h, \mathcal{L}\hat{u}^h - f \rangle \tag{3.48}$$

and the error $\delta \hat{u}^h = u^h - \hat{u}^h$ have become so smooth that the iterative method cannot appreciably reduce the error any more. Notice that provided the operator is linear, we can write an equation for the error $\delta \hat{u}^h$ as

$$\langle \varphi_k^h, \mathcal{L}\delta\hat{u}^h \rangle = \langle \varphi_k^h, f - \mathcal{L}\hat{u}^h \rangle = -r_k^h.$$
(3.49)

Now we transfer this equation to the coarse mesh, since there $\delta \hat{u}^h$ and r^h are not quite as smooth, so that the iterative method should perform better. Thus we attempt to find a coarse approximation to the error $\delta \hat{u}^H$ by applying the iterative method to the problem

$$\langle \varphi_{\ell}^{H}, \mathcal{L}\delta\hat{u}^{H} \rangle = -r_{\ell}^{H} = \langle \varphi_{\ell}^{H}, \mathcal{L}\hat{u}^{h} - f \rangle = -\sum_{k} \mathcal{P}_{k\ell} r_{k}^{h}.$$
(3.50)

After a number of iterations we transfer the result $\delta \hat{u}^H$ back to the fine mesh to obtain a better guess $\hat{u}^{h,\text{new}} = u^{h,\text{old}} + \mathcal{P}\delta \hat{u}^H$ for the solution.

The multigrid method simply consist of extending this simple idea to a whole sequence of meshes obtained by refinement of an initial coarse one. We have implemented a routine **prolong** that determines an explicit matrix representation of the prolongation operator \mathcal{P} relating a coarse mesh to a fine mesh obtained by regular refinement of the coarse one, when the refinement is done with halfmesh. Further we have implemented a routine mglin that applies the linear multigrid method described above to a problem when provided a set of prolongation operators and the system matrix and right hand side on the finest mesh. The routine applies the Gauss-Seidel method as default. We have tried using other iterative methods as well, such as conjugate gradients method, but none of them matched the Gauss-Seidel; further we are not sure to what extent those methods tried out are actually smoothers.

3.5.5 Solution of nonlinear system of equations

Above we elaborated on the solution of a linear problem; when faced with a non-linear operator \mathcal{L} , one way to proceed is to linearize it and use Newtons method to solve the problem. We define u^* as the true solution to the discretized problem $\langle \varphi_k, \mathcal{L}(u^*) - f \rangle = 0$ and assume further that we have an initial guess u^i for the solution. Also we define the residual as $r_k(u^i) = \langle \varphi_k, \mathcal{L}(u^i) - f \rangle$. Assuming u^i is close to u^* we may linearize r to obtain, in matrix form

$$\mathbf{0} = \mathbf{r}(\mathbf{u}^*) \simeq \mathbf{r}^i + \mathbf{J}^i \, (\mathbf{u}^* - \mathbf{u}^i), \tag{3.51}$$

where $\mathbf{J}^{i} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \end{bmatrix}_{\mathbf{u}=\mathbf{u}_{i}}$ is the Jacobian matrix. Thus we obtain our next guess for the solution \mathbf{u}^{i+1} by solving the problem

$$\mathbf{J}^i \Delta \mathbf{u}^i = -\mathbf{r}^i \tag{3.52}$$

for $\Delta \mathbf{u}^i$ and setting

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \Delta \mathbf{u}^i. \tag{3.53}$$

This way the solution of the non-linear problem is reduced to that of solving a sequence of linear problems. Once the trial solution \mathbf{u}^i is close enough to \mathbf{u}^* and provided \mathbf{J} is not singular at \mathbf{u}^* , Newtons method is known to have quadratic convergence speed. That is, if the error of *i*th trial solution is $\epsilon_i = |\mathbf{u}^i - \mathbf{u}^*|$ then $\epsilon_{i+1} = |\mathbf{u}^{i+1} - \mathbf{u}^*| \simeq C\epsilon_i^2$ for some constant *C*. This is fast, and once within this asymptotic regime, typically only a few iterations are needed to reach the desired accuracy.

However, outside the asymptotic region one may spend a lot of time, and Newtons method is prone to diverence if a bad initial guess is supplied. One way to ensure convergence is to modify the step as $\mathbf{u}^{i+1} = \mathbf{u}^i + \alpha \Delta \mathbf{u}^i$ where α is some parameter in the interval [0, 1] to be picked by performing a so-called line search for α . However, the residual $\mathbf{r}(\mathbf{u}^i + \alpha \Delta \mathbf{u}^i)$ is expensive to evaluate, and therefore one should not spend too much time searching for an optimal value of α . A particularly simple approach is the so-called Armijo-Goldstein in-exact line search, where α is picked as the first number in the sequence $[1, \frac{1}{2}, \frac{1}{4}, \dots, \frac{1}{2^n}]$ for which the residual new residual r^{i+1} is decreased from the old r^i by at least a factor of $(1 - \frac{\alpha}{2})$.

Notice that the elements of the Jacobian matrix $\mathbf{J}_{k\ell}$ can be computed from element integrals of the form

$$\mathbf{J}_{k\ell} = \left\langle \varphi_k, \left[\frac{\partial [\mathcal{L}(u) - f]}{\partial u} \right] \varphi_\ell \right\rangle + \left\langle \varphi_k, \left[\frac{\partial [\mathcal{L}(u) - f]}{\partial \mathbf{\nabla} u} \right] \cdot \mathbf{\nabla} \varphi_\ell \right\rangle.$$
(3.54)

Thus we can move the differentiation from the discretized to the analytical equations, which is convenient.

We provide Newtons method as the routine **newt**, which requires the user implement a subfunction computing the Jacobian matrix and the residual for the problem.

Chapter 4

Application of FEM to hydrodynamics

In this chapter we apply the finite element method to the Navier-Stokes equation for laminar incompressible flow.

It turns out that the numerical scheme obtained from the most straightforward application of the FEM fails, as one can show that the pressure solution is ill-determined when the same basis functions are used to approximate both the velocity field v and the pressure p. The classical way of dealing with this problem is to choose the order of accuracy for the velocity approximation one order higher than for the pressure.

However, alternative methods for stabilizing the pressure is subject to active research [12, 13, 14, 15]. Some are focused on stabilizing the simplest possible approximation pair with piecewise linear velocity and piecewise constant pressure. Other formulations allow equal order approximations to be used for all variables, which is a major convenience in relation to the implementation. In Sec. 4.3 we shall discuss the so-called pressure gradient projection method (PGP), which is of this latter type. We have chosen to work with this scheme because it is conceptually simple and rather easy to implement; at the present stage, though, both the classical method and the PGP can be handled easily with our FEM tool.

As our FEM tool is geared only towards two-dimensional problems so far, we shall only consider two-dimensional flows in this chapter. However, the quasi three-dimensional case of axisymmetric flow can also be handled since it is effectively only a two dimensional problem.

4.1 Discretization of the Navier-Stokes equation

The Navier-Stokes equation for incompressible flow is, c.f. Eq. (2.14)

$$\rho \left[\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla) \boldsymbol{v} \right] = -\nabla p + \mu \nabla^2 \boldsymbol{v} + \boldsymbol{f}, \qquad (4.1)$$

which is supplemented by the incompressibility constraint $\nabla \cdot \boldsymbol{v} = 0$. We choose some basis spaces \mathbb{V}^h and \mathbb{Q}^h for the approximation of \boldsymbol{v} and p, and apply the Galerkin method as discussed in Sec. 3.2.1

$$\rho \Big[\langle \varphi_k, \frac{\partial \boldsymbol{v}}{\partial t} \rangle + \langle \varphi_k, (\boldsymbol{v} \cdot \boldsymbol{\nabla}) \boldsymbol{v} \rangle \Big] + \mu \langle \boldsymbol{\nabla} \varphi_k; \boldsymbol{\nabla} \boldsymbol{v} \rangle - \langle \boldsymbol{\nabla} \varphi_k, p \rangle \\ = \langle \varphi_k, \boldsymbol{f} \rangle - \int_{\partial \Omega} \mathrm{d}s \, \varphi_k [\boldsymbol{n} p - \mu (\boldsymbol{n} \cdot \boldsymbol{\nabla}) \boldsymbol{v}], \quad \text{for all } \varphi_k \in \mathbb{V}^h \quad (4.2)$$

and

$$\langle \psi_m, \boldsymbol{\nabla} \cdot \boldsymbol{v} \rangle = 0, \quad \text{for all } \psi_m \in \mathbb{Q}^h.$$
 (4.3)

This is written compactly in matrix notation as

$$\rho \left[\mathbf{M} \dot{\mathbf{v}}_x + \mathbf{C} \mathbf{v}_x \right] + \mu \mathbf{K} \mathbf{v}_x - \mathbf{Q}_x^T \mathbf{p} = \mathbf{f}_x \tag{4.4a}$$

$$\rho \left[\mathbf{M} \dot{\mathbf{v}}_y + \mathbf{C} \mathbf{v}_y \right] + \mu \mathbf{K} \mathbf{v}_y - \mathbf{Q}_y^T \mathbf{p} = \mathbf{f}_y \tag{4.4b}$$

$$\mathbf{Q}_x \mathbf{v}_x + \mathbf{Q}_y \mathbf{v}_y = \mathbf{0} \tag{4.4c}$$

where the column vectors \mathbf{v}_x , \mathbf{v}_y and \mathbf{p} hold the expansion coefficients $v_{x,\ell}$ etc. The matrices appearing are the mass matrix

$$\mathbf{M}_{k\ell} = \langle \varphi_k, \varphi_\ell \rangle \tag{4.5}$$

and the stiffness matrix

$$\mathbf{K}_{k\ell} = \langle \boldsymbol{\nabla} \varphi_k; \boldsymbol{\nabla} \varphi_\ell \rangle. \tag{4.6}$$

Further the matrix \mathbf{C} represents the action of the convective operator, and is given by

$$\mathbf{C}_{k\ell} = \left\langle \varphi_k; \left[\sum_m \varphi_m v_{x,m} \right] \frac{\partial \varphi_\ell}{\partial x} + \left[\sum_m \varphi_m v_{y,m} \right] \frac{\partial \varphi_\ell}{\partial y} \right\rangle$$
(4.7)

whereas the matrices \mathbf{Q} represent the divergence action

$$\mathbf{Q}_{x,k\ell} = \langle \psi_k, \frac{\partial \varphi_\ell}{\partial x} \rangle \tag{4.8}$$

and

$$\mathbf{Q}_{y,k\ell} = \langle \psi_k, \frac{\partial \varphi_\ell}{\partial y} \rangle. \tag{4.9}$$

Finally the right hand sides \mathbf{f}_x and \mathbf{f}_y include both the body force term and the boundary integral

$$\mathbf{f}_{x,k} = \langle \varphi_k, f_x \rangle - \int_{\partial \Omega} \mathrm{d}s \, \varphi_k[\, n_x \, p - \mu(\boldsymbol{n} \cdot \boldsymbol{\nabla}) v_x], \qquad (4.10)$$

and similarly for $\mathbf{f}_{y,k}$. Notice that the discretized equations do not contain derivatives of the pressure basis functions; therefore they need not be continuous but might well be chosen e.g. piecewise constant.

We shall only be concerned with stationary flow for which $\dot{\mathbf{v}}_x = \dot{\mathbf{v}}_y = 0$. We collect all the solution components in a single column vector $\mathbf{u} = [\mathbf{v}_x^T \ \mathbf{v}_y^T \ \mathbf{p}^T]^T$ and define the residual $\mathbf{r}(\mathbf{u})$ with

$$\mathbf{r}_x = \rho \mathbf{C} \mathbf{v}_x + \mu \mathbf{K} \mathbf{v}_x - \mathbf{Q}_x^T \mathbf{p} - \mathbf{f}_x \tag{4.11a}$$

$$\mathbf{r}_y = \rho \mathbf{C} \mathbf{v}_y + \mu \mathbf{K} \mathbf{v}_y - \mathbf{Q}_y^T \mathbf{p} - \mathbf{f}_y$$
(4.11b)

$$\mathbf{r}_0 = \mathbf{Q}_x \mathbf{v}_x + \mathbf{Q}_y \mathbf{v}_y. \tag{4.11c}$$

Thus stationary flow is equivalent to solving for $\mathbf{r}(\mathbf{u}) = \mathbf{0}$. We apply Newtons method which requires the Jacobian matrix $\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \end{bmatrix}$ to be computed. Only the convective term is non-linear so

$$\mathbf{J} = \begin{bmatrix} \rho \mathbf{C} + \mu \mathbf{K} + \rho \begin{bmatrix} \frac{\partial \mathbf{C}}{\partial \mathbf{v}_x} \mathbf{v}_x \end{bmatrix} & \rho \begin{bmatrix} \frac{\partial \mathbf{C}}{\partial \mathbf{v}_y} \mathbf{v}_x \end{bmatrix} & -\mathbf{Q}_x^T \\ \rho \begin{bmatrix} \frac{\partial \mathbf{C}}{\partial \mathbf{v}_x} \mathbf{v}_y \end{bmatrix} & \rho \mathbf{C} + \mu \mathbf{K} + \rho \begin{bmatrix} \frac{\partial \mathbf{C}}{\partial \mathbf{v}_y} \mathbf{v}_y \end{bmatrix} & -\mathbf{Q}_y^T \\ \mathbf{Q}_x & \mathbf{Q}_y & \mathbf{O} \end{bmatrix}$$
(4.12)

where \mathbf{O} is all zeros and

$$\left[\frac{\partial \mathbf{C}}{\partial \mathbf{v}_x} \mathbf{v}_x\right]_{k\ell} = \left\langle \varphi_k, \varphi_\ell \left[\sum_m \frac{\partial \varphi_m}{\partial x} v_{x,m}\right] \right\rangle \tag{4.13}$$

$$\left[\frac{\partial \mathbf{C}}{\partial \mathbf{v}_{y}}\mathbf{v}_{x}\right]_{k\ell} = \left\langle \varphi_{k}, \varphi_{\ell}\left[\sum_{m} \frac{\partial \varphi_{m}}{\partial y} v_{x,m}\right]\right\rangle \tag{4.14}$$

$$\left[\frac{\partial \mathbf{C}}{\partial \mathbf{v}_{x}}\mathbf{v}_{y}\right]_{k\ell} = \left\langle \varphi_{k}, \varphi_{\ell}\left[\sum_{m} \frac{\partial \varphi_{m}}{\partial x} v_{y,m}\right]\right\rangle \tag{4.15}$$

$$\left[\frac{\partial \mathbf{C}}{\partial \mathbf{v}_{y}}\mathbf{v}_{y}\right]_{k\ell} = \left\langle \varphi_{k}, \varphi_{\ell}\left[\sum_{m} \frac{\partial \varphi_{m}}{\partial y} v_{y,m}\right]\right\rangle.$$
(4.16)

4.1.1 Boundary conditions

In a typical microfluidic problem the boundary of the computational domain is defined by a set of channel walls. Further the domain is connected to the environment by two or more fluidic leads through which the liquid enters and leaves the system; these are typically assumed to be long and straight so that sufficiently far upstream or downstream in the leads, we can assume that the velocity profile is fully developed and parabolic.

On the channel walls we impose the no-slip condition v = 0, whereas the pressure is generally unknown and therefore left unspecified. On the lead boundaries we typically know either the flow rate or the pressure, which corresponds to Dirichlet and Neumann boundary conditions respectively. Of course, if the flow rate is specified in all leads then it had better be chosen such that the net flow into the domain is zero, since otherwise liquid will be accumulating and the incompressibility constraint cannot be satisfied. Even if a consistent choice has been made, numerical errors can challenge this and create wiggles in the solution; thus generally it is better to have at least one boundary on which the pressure level is defined while the flow rate is left to be determined.

Considering the boundary integral in Eq. (4.10) then on the channel walls we never need to compute it since we replace the Navier-Stokes equation with the no-slip Dirichlet condition for all φ_k on the wall. This is also the case on those lead boundaries where the flow rate is specified, only we impose a parabolic profile rather than no-slip.

On the lead boundaries on which the pressure is known, this is used as input for the boundary integral, while we assume that the flow is fully developed at the lead boundary such that $(\boldsymbol{n} \cdot \boldsymbol{\nabla}) \boldsymbol{v} = \mathbf{0}^{1}$. Notice however that this procedure does *not* actually ensure that the pressure at the lead boundary is equal to that specified – generally it is not! Only if the section of the lead channel that we include in the computational domain is very long will the actual solution be fully developed with $(\boldsymbol{n} \cdot \boldsymbol{\nabla}) \boldsymbol{v} = \mathbf{0}$. As we are interested in keeping the computational cost low, we tend to choose the lead section as small as possible – thus in a sense we now have a way of checking whether the lead section included is sufficiently large, simply by checking if the pressure is far from or close to the specified value.²

Finally, on boundaries corresponding to symmetry lines, the velocity component normal to the symmetry line is zero which is imposed as a homogeneous Dirichlet condition $v_{\perp} = 0$. The velocity component parallel to the symmetry line is generally unknown, while the normal derivative should be zero $(\boldsymbol{n} \cdot \boldsymbol{\nabla})v_{\parallel} = 0$ which is used as input for the boundary integral. Also the pressure is unknown, which fits well the fact that it drops out of the boundary integral for v_{\parallel} , as the component of the boundary outward normal \boldsymbol{n} is of course zero in the direction parallel to the boundary. If furthermore the symmetry axis is chosen to be parallel with one of the coordinate axes, then one avoids having to form linear combinations of v_x and v_y in order to satisfy the mixed Dirichlet and Neumann conditions.

Example 4.1

We now proceed with a small numerical example using the simple piecewise linear basis functions to approximate both the pressure and the velocity. As mentioned above such a straightforward approach should fail, and the example shows that indeed it does.

We consider a channel with a bend as shown in Fig. 4.1, where a wide rectangular channel of width W and depth d goes through a 90° bend. We assume that the channel width is much larger than its depth such that

 $^{^1\}mathrm{This}$ assumes that the boundary is perpendicular to the lead axis, though.

²As long as the basis functions used for the pressure approximation are continuous it is not strictly necessary to do partial integration on the pressure term. If this is left out, only the $(n \cdot \nabla)v$ term is present in the boundary integral. The pressure is then specified as a Dirichlet condition, thereby neglecting the incompressibility constraint at the lead boundary. Thus in this case it is possible to specify $(n \cdot \nabla)v = 0$ and $p = p_0$ separately – but if the lead section is too short such a condition is unphysical.



Figure 4.1: Channel with a bend. A rectangular channel of width W and depth d bends down. It is assumed that the aspect ratio $\alpha = d/W$ is very small such that the flow in the central part of the channel is effectively independent of z.

in the central part of the channel the flow is essentially two-dimensional $\boldsymbol{v} = v_x(x, y)\boldsymbol{e}_x + v_y(x, y)\boldsymbol{e}_y$.

We impose Neumann type boundary conditions on the inflow and outflow boundaries with $\boldsymbol{n}p - \mu(\boldsymbol{n} \cdot \boldsymbol{\nabla})\boldsymbol{v} = \boldsymbol{n}p_0$ at the inflow boundary and $\boldsymbol{n}p - \mu(\boldsymbol{n} \cdot \boldsymbol{\nabla})\boldsymbol{v} = \boldsymbol{0}$ at the outflow, corresponding to a pressure drop of $\Delta p = p_0$ across the channel.



Figure 4.2: Flow in channel with a 90° bend; straightforward FEM approach using the same (linear) basis functions for both velocity and pressure. a) Streamlines for the flow with the finite element mesh used for the computation shown below. b) Surface plot of pressure distribution in the channel. The general trend of the solution is a linear pressure drop from p_0 at the inflow to zero at the outflow which is what we would expect; however there is also a rapid and unphysical oscillation in the solution. The Reynolds number computed from the maximal velocity of the solution is $Re = v_{max} d\rho/\mu = 1.95$.

The results are shown in Fig. 4.2 showing the streamlines for the flow and the pressure variation in the channel. The streamlines appear as we would expect, and indeed the velocity solution is rather well behaved. For



Figure 4.3: One-dimensional analogy to pressure instability. Top row shows the dodgy mode q whereas second row shows a representative one dimensional chapeau function ϕ_k . Third row shows the value of the product $q \cdot \frac{\partial \varphi_k}{\partial x}$ and it is evident that $\int_{\Omega} dx \, q \cdot \frac{\partial \varphi_k}{\partial x} = \langle \frac{\partial \varphi_k}{\partial x}, q \rangle = 0$ for all interior nodes x_k . The mode q is not orthogonal to the basis functions at the boundary nodes \circ – but on all wall boundaries the Navier-Stokes equation is not solved but replaced by the Dirichlet boundary condition.

the pressure solution the general trend is a linear pressure drop from p_0 at the inflow to zero at the outflow which is also as we would expect. However there seems to be some kind of noise in the solution as it oscillates rapidly, jumping up and down from one mesh node to the next, which is obviously unphysical. Still the solution seems to get the overall trend right, and we might think of just filtering out the noisy oscillatory part. However, the relatively good behaviour turns out to be caused by the rather irregular mesh used, which tends to dampen the oscillations. If a more regular mesh is used, then the oscillations grow several orders of magnitude larger than p_0 and the influence on the velocity field is not neglible.

4.2 The Babuška-Brezzi inf-sup condition

The cause of the instability observed in Example 4.1 turns out to be that there exist a rapidly oscillatory pressure mode which is orthogonal to $\nabla \varphi_k$ for all φ_k , that is, there exist $q \in \mathbb{Q}^h$ such that $\langle \nabla \varphi_k, q \rangle = 0$ for all $\varphi_k \in \mathbb{V}^h$. Then if p is the physical solution to the equations Eqs. (4.11), so is p + q, and the pressure solution is therefore ill-determined.

This feature is easily understood from a simple one-dimensional analogy as shown in Fig. 4.3. The figure shows that it is possible for a mode q to obey homogeneous Dirichlet boundary conditions and be orthogonal to the basis functions at all interior nodes. The mode q is *not* orthogonal to the basis functions at the boundary – but the projection of the Navier-Stokes equation onto the basis functions at all Dirichlet type boundary nodes is neglected and replaced by the Dirichlet boundary condition.

The problem turns out to be general when the same basis space is used for both the velocity and pressure approximation. The pathology can be formulated as the fact that the quantity

$$\beta \equiv \inf_{q \in \mathbb{Q}^h} \sup_{\boldsymbol{v} \in \mathbb{V}^h} \frac{|\langle \boldsymbol{\nabla} \cdot \boldsymbol{v}, q \rangle|}{\|\boldsymbol{v}\| \|q\|}$$
(4.17)

is zero. To avoid the spurious pressure oscillations it is necessary that β be positive – known as the Babuška-Brezzi (BB) inf-sup condition [6].

The cure for the problem and the way to satisfy the BB condition is to make the space \mathbb{V}^h richer than \mathbb{Q}^h – the idea is quite obvious as we wish to avoid the existence of q such that $\langle \nabla \phi_k, q \rangle = 0$; we try to obtain this by adding more ϕ_k to \mathbb{V}^h so as to make it more difficult for q. In order to stabilize the pressure solution when \mathbb{Q}^h is the space of piecewise linear functions it is sufficient choose \mathbb{V}^h as the space of piecewise quadratics.

Another candidate $\mathbb{V}^h/\mathbb{Q}^h$ pair that we might think of could be piecewise linear velocity and piecewise constant pressure; however this combination does *not* satisfy the BB condition. Yet for quadrangular elements the use of bilinear velocity and piecewise constant pressure is found to yield acceptable results [16]. With quadratic velocity and piecewise constant pressure the BB condition is satisfied – but the convergence is only as $\mathcal{O}(h)$ whereas we obtain $\mathcal{O}(h^2)$ with quadratic velocity and linear pressure.

Example 4.2

We now repeat the Example 4.1 using piecewise quadratic basis functions for the velocity and piecewise linear pressure; otherwise the geometry and boundary conditions are identical to Example 4.1. The results are shown in Fig. 4.4. Comparing with Example 4.1 we find that the rapid pressure



Figure 4.4: Flow in channel with a 90° bend; pressure instability cured by using quadratic velocity and linear pressure basis functions which satisfies the BB condition Eq. (4.17). *a*) Streamlines with finite element mesh shown below. *b*) Surface plot of pressure distribution in the channel. The spurious pressure oscillations observed in Example 4.1 are now absent. The Reynolds number is $Re = v_{\text{max}} d\rho/\mu = 1.96$.

oscillation is now absent. The velocity solutions of Examples 4.1 and 4.2

differ by less than 1% indicating that the accuracy was not severely degraded by spurious pressure field. \blacksquare

4.3 The pressure gradient projection method (PGP)

The pressure gradient projection method (PGP) is an alternative method proposed recently by Cordina *et al.* [12, 17] to stabilize the pressure instability discussed above. The method allows the same basis functions to be used for all variables.

The key point is the observation that the unstable pressure modes with $\langle \nabla \varphi_k, q \rangle$ for all $\varphi_k \in \mathbb{V}$ are highly oscillatory. Notice also that this means that the pressure gradient has large jumps between adjacent elements. The idea of the PGP method is to add a term to the equations that will act like a sort of artificial diffusion, so as to smear out the pressure. Of course as pressure diffusion is not present in the physical system such an approach will introduce error in the solution. Therefore the additional term is multiplied by a small factor to ensure that the error introduced is of the order of the truncation error of the FEM solution in order not to upset accuracy. Further we shall see that the additional term vanishes whenever the gradient of the pressure solution can be expressed as a continuous function within the basis space; only when the pressure gradient is discontinuous does the additional term show up.

We introduce a new discrete variable $\boldsymbol{\xi}^h = \sum_{\ell} \boldsymbol{\xi}_{\ell} \varphi_{\ell}$ that should be a continuous approximation to the gradient of the pressure p. This is obtained by requiring $\langle \varphi_k, \boldsymbol{\xi} \rangle = \langle \varphi_k, \boldsymbol{\nabla} p \rangle$ for all $\varphi_k \in \mathbb{H}^h$, or in matrix form

$$\mathbf{M}\boldsymbol{\xi}_x = \mathbf{Q}_x \mathbf{p} \quad \text{and} \quad \mathbf{M}\boldsymbol{\xi}_y = \mathbf{Q}_y \mathbf{p}, \tag{4.18}$$

where the matrices \mathbf{M} and \mathbf{Q} were introduced in Sec. 4.1. Then we subtract the divergence of the difference $\tau [\nabla p - \boldsymbol{\xi}]$ from the incompressibility constraint, where τ is some small algorithmic parameter to be defined below. After partial integration of the term $\langle \phi_k, \nabla \cdot \tau [\nabla p - \boldsymbol{\xi}] \rangle$ we obtain a modified incompressibility constraint

$$\langle \varphi_k, \boldsymbol{\nabla} \cdot \boldsymbol{v} \rangle + \langle \boldsymbol{\nabla} \varphi_k, \tau [\boldsymbol{\nabla} p - \boldsymbol{\xi}] \rangle = 0, \quad \text{for all } \varphi_k \in \mathbb{H}^h.$$
 (4.19)

In matrix notation the full system of equations reads, c.f. Eqs. (4.4)

$$\rho \mathbf{C} \mathbf{v}_x + \mu \mathbf{K} \mathbf{v}_x - \mathbf{Q}_x^T \mathbf{p} = \mathbf{f}_x \tag{4.20a}$$

$$\rho \mathbf{C} \mathbf{v}_y + \mu \mathbf{K} \mathbf{v}_y - \mathbf{Q}_y^T \mathbf{p} = \mathbf{f}_y \tag{4.20b}$$

$$\mathbf{Q}_x \mathbf{v}_x + \mathbf{Q}_y \mathbf{v}_y + \mathbf{K}_\tau \mathbf{p} - \mathbf{Q}_{x,\tau}^T \boldsymbol{\xi}_x - \mathbf{Q}_{y,\tau}^T \boldsymbol{\xi}_y = \mathbf{0}$$
(4.20c)

$$\mathbf{M}\boldsymbol{\xi}_x - \mathbf{Q}_x \mathbf{p} = \mathbf{0} \tag{4.20d}$$

 $\mathbf{M}\boldsymbol{\xi}_{y} - \mathbf{Q}_{y}\mathbf{p} = \mathbf{0} \tag{4.20e}$
where $\mathbf{K}_{\tau,k\ell} = \langle \nabla \varphi_k; \tau \nabla \varphi_\ell \rangle$ and similarly for $\mathbf{Q}_{x,\tau}$ and $\mathbf{Q}_{y,\tau}$. The parameter τ is defined within each element Ω_{α} as the following constant

$$\tau_{\alpha} = \left[c_1 \frac{\mu}{h^2} + c_2 \frac{\rho v_{\alpha}}{h}\right]^{-1},\tag{4.21}$$

where v_{α} is the maximal value of the velocity field within Ω_{α} and h is a measure of the size of the element which we define as the length of the longest side. Finally the two constants c_1 and c_2 are taken as either $c_1 = 2$ and $c_2 = 4$ when linear basis functions are used or $c_1 = 4$ and $c_2 = 16$ for quadratic basis functions [17].

While in practice we always solve the equations in the form of Eqs. (4.20), it is instructive to eliminate the variable $\boldsymbol{\xi}$ from the system by formally inverting the mass matrix **M**. Thus we take $\boldsymbol{\xi}_x = \mathbf{M}^{-1}\mathbf{Q}_x\mathbf{p}$ and similarly for $\boldsymbol{\xi}_y$ to obtain a system of the form

$$\rho \mathbf{C} \mathbf{v}_x + \mu \mathbf{K} \mathbf{v}_x - \mathbf{Q}_x^T \mathbf{p} = \mathbf{f}_x \tag{4.22}$$

$$\rho \mathbf{C} \mathbf{v}_y + \mu \mathbf{K} \mathbf{v}_y - \mathbf{Q}_y^T \mathbf{p} = \mathbf{f}_y \tag{4.23}$$

$$\mathbf{Q}_x \mathbf{v}_x + \mathbf{Q}_y \mathbf{v}_y + \left[\mathbf{K}_\tau - \mathbf{K}_\tau \right] \mathbf{p} = \mathbf{0}$$
(4.24)

where $\widetilde{\mathbf{K}}_{\tau} = \mathbf{Q}_{x,\tau}^T \mathbf{M}^{-1} \mathbf{Q}_x + \mathbf{Q}_{y,\tau}^T \mathbf{M}^{-1} \mathbf{Q}_y$. The matrix $[\mathbf{K}_{\tau} - \widetilde{\mathbf{K}}_{\tau}]$ is positive semi-definite, meaning that $\mathbf{q}^T [\mathbf{K}_{\tau} - \widetilde{\mathbf{K}}_{\tau}] \mathbf{q} \ge 0$ for all \mathbf{q} . This is easily verified by considering

$$0 \le \langle \boldsymbol{\xi} - \boldsymbol{\nabla} p; \boldsymbol{\xi} - \boldsymbol{\nabla} p \rangle = \mathbf{p}^T \big[\mathbf{K}_{\tau} - \widetilde{\mathbf{K}}_{\tau} \big] \mathbf{p} , \qquad (4.25)$$

where we use that $\boldsymbol{\xi}_x = \mathbf{M}^{-1}\mathbf{Q}_x\mathbf{p}$. The notion *semi*-definite corresponds to the fact that $\mathbf{q}^T [\mathbf{K}_{\tau} - \widetilde{\mathbf{K}}_{\tau}] \mathbf{q} = 0$ does not imply $\mathbf{q} = \mathbf{0}$; indeed for all p that have a continuous gradient such that the equality $\boldsymbol{\xi} = \boldsymbol{\nabla} p$ holds, we obtain $\mathbf{p}^T [\mathbf{K}_{\tau} - \widetilde{\mathbf{K}}_{\tau}] \mathbf{p} = 0$. In a sense the additional term introduced adds to the coercivity of the system, c.f. Sec. 3.2.2.

Example 4.3

We now consider once again the flow in a channel with a 90° bend from Example 4.1, and apply the pressure gradient projection method to the problem using piecewise linear basis functions for all variables. The result is shown in Fig. 4.5. Comparing with Example 4.1 we find that the rapid pressure oscillation is again absent. Comparing with the results obtained with mixed interpolation in Example 4.2 we find that both the velocity and pressure solutions agree to within 0.5%. Using instead piecewise quadratic basis functions for all variables, we find that the PGP method and the mixed method agree to within 5×10^{-4} .

As a final remark we compare the convergence of the PGP method with the classical mixed interpolation formulation. In [12] it is proven that for



Figure 4.5: Flow in channel with a 90° bend; pressure instability cured by the pressure gradient projection method. a) Streamlines with finite element mesh shown below. b) Surface plot of pressure distribution in the channel. The spurious pressure oscillations observed in Example 4.1 are again absent. The Reynolds number is $Re = v_{\text{max}} d\rho/\mu = 1.95$.

small Reynolds numbers and provided the true solution is smooth enough, the solution obtained with the PGP method should converge as

$$\mu \|\boldsymbol{\nabla}\boldsymbol{v} - \boldsymbol{\nabla}\boldsymbol{v}_h\|^2 + \tau \|\boldsymbol{\nabla}p - \boldsymbol{\nabla}p_h\|^2 \le Ch^{2k}$$
(4.26)

where the constant C is independent of the mesh but possibly dependent on μ , and k is the accuracy of the basis functions.³

In order to check the convergence numerically we focus on a problem for which the analytic solution is known. It is easily verified that the velocity and pressure fields

$$v_x(x,y) = 1 - e^{\lambda x} \cos(2\pi y) \tag{4.27}$$

$$v_y(x,y) = \frac{\lambda}{2\pi} e^{\lambda x} \sin(2\pi y) \tag{4.28}$$

$$p(x,y) = -\frac{1}{2}e^{2\lambda x} \tag{4.29}$$

satisfies both the Navier-Stokes equation and the incompressibility constraint when $\rho = 1$ and $\mu = 1/Re$ provided the parameter λ is chosen as $\lambda = Re/2 - (Re^2/4 + 4\pi^2)^{1/2}$. This problem was introduced by Kovasznay [18], modelling laminar flow behind a two-dimensional grid. We chose Re =10 and solved the problem on the rectangular domain $(x, y) \in [0, 1] \times [0, 2]$, imposing Dirichlet boundary conditions on all the boundary with the value of the analytical solution. We solved the problem with the PGP method for both piecewise linear and piecewise quadratic basis functions and with the classical mixed method using quadratic basis for the velocity and linear basis for the pressure.

³In Chap. 3 we used the symbol p for accuracy which is the standard FEM terminology; here however we prefer to reserve p for the pressure.



Figure 4.6: Convergence in the H^1 norm of PGP method and mixed method for Kovasznay flow as function of mesh size h. \circ : PGP method with linear basis functions. \triangle : PGP method with quadratic basis functions. \Box : Mixed method with quadratic velocity and linear pressure basis functions.

In Fig. 4.6 the results are shown as the mesh size h is varied between 1/2and 1/32. It is seen that for the PGP method with linear basis functions the error in the H^1 norm decays as $\mathcal{O}(h^1)$ for both velocity and pressure, while with quadratic basis functions the error decays as $\mathcal{O}(h^2)$ for the velocity and as $\mathcal{O}(h^{1.5})$ for the pressure. For the mixed method the error decays as $\mathcal{O}(h^2)$ for the velocity and $\mathcal{O}(h^1)$ for the pressure. Notice that in Eq. (4.26) the parameter τ enters, which is dependent on h. Therefore the convergence of the pressure in the PGP method is not assured to be as fast as for the velocity, and this is what we observe for quadratic interpolation.

Chapter 5

Flow over backward-facing step

In this chapter we analyse the flow over a backward-facing step using the finite element tools that we developed during this project. The backstep problem is a classical one and is often used as test bench for computational fluid dynamics (CFD) programs; this is part of our motivation for considering this geometry.

In Chap. 6 we shall discuss an experiment where the detailed velocity profile of blood flowing in a thin glass capillary has been measured using micro particle-image velocimetry (μ PIV). It was also planned to perform such measurements on blood flow in more complicated geometries such as a channel with a backstep, since it is known that in such geometries certain non-Newtonian liquids display remarkable flow patterns with vortices appearing that would not be expected in a Newtonian flow. It was the idea to investigate if blood would display such features in flows on the microscale. Thus our second motivation for considering the backstep geometry was to be able to compare the experimental results with simulations of the flow. However it turned out to be not exactly straightforward to do the μ PIV experiments and we never made it to the more complex geometries experimentally.

The geometry that we consider is that of a long straight rectangular channel of width W and depth h that expands suddenly to a depth of H = 2h as shown in Fig. 5.1. The channel is assumed to have a very small aspect ratio $\alpha = H/W \ll 1$, such that the flow in the central part can be considered effectively a two dimensional problem with no variation in the direction along the step.

We shall compare our simulation results with an experiment performed by Armaly *et al.* [19]. They investigated gas flow over a backward-facing step for Reynolds numbers in the range 70 < Re < 8000, defining the Reynolds number as $Re = 2h\bar{v}_0/\nu$ where \bar{v}_0 is the average velocity at the inflow



Figure 5.1: The backstep geometry. A rectangular channel of width W and depth h suddenly expands to a depth H = 2h. The channel is assumed to have a very small aspect ratio $\alpha = H/W \ll 1$ such that the flow in the central part of the channel is effectively independent of z.

boundary and $\nu = \mu/\rho$ is the kinematic viscosity. This range covers both the laminar (Re < 1200), transitional (1200 < Re < 6600) and turbulent (Re > 6600) regimes of the flow. However, even though the aspect ratio of the channel in the experiment was $\alpha = 1/36$, the authors report that the flow lost its two dimensional nature already for Re > 400.

We have applied the PGP method discussed in Sec. 4.3 for this problem using piecewise quadratic basis functions for all variables. We initially focus on the case Re = 100, choosing channel dimensions H = 2h, l = 4h and L = 12h, c.f. Fig. 5.1.

5.1 Regular mesh refinement

We first solved the problem on a regular and very coarse mesh as shown in Fig. 5.2 using an all zero initial guess – this is quite far from the true solution and the Newton method required 19 iterations to converge. Only the last four iterations were taken within the asymptotic regime of quadratic convergence of the Newton method. The relative error tolerance was chosen as $\varepsilon_r = 10^{-8}$ and the absolute tolerance $\varepsilon_a = 10^{-10}$; thus the solution was accepted only when the last Newton step $\Delta \mathbf{u}$ satisfied $\|\Delta \mathbf{u}\| \leq \varepsilon_r \|\mathbf{u}\| + \varepsilon_a$.

Subsequently the mesh was refined with regular mesh refinement dividing each element in the mesh into four smaller elements. The solution from the coarse mesh was then used as initial guess on the fine mesh and the solution met the tolerance after only 6 Newton iterations. This process was repeated three times such that in the final mesh every element in the coarse mesh was divided into $4^3 = 64$ smaller elements and the total number of elements was 5312.

On the finest mesh the solution time was dominated by the time required to solve the linear system in each Newton step which took up 60 seconds on our computer,¹ however only 5 iterations were needed to converge. This should be compared with the total time of 30 seconds required to do all 19

¹IBM 933 Mz Pentium 4 with 500 MB RAM.



Figure 5.2: Solution for Re = 100 after three regular mesh refinements.

a): Streamlines. Notice the recirculating region behind the step; the velocity in the recirculation region is much smaller than in the main flow, therefore the streamlines there are drawn almost 10 times as dense as in the main flow. The triangle marks the *reattachment point* where the streamline leaving the sharp corner of the step reattaches to the channel wall. b): Pressure isobars. The pressure has a local minimum just before the reattachment point. c): Initial coarse mesh. The final solution was computed on a fine mesh obtained after dividing each of these elements into 64 smaller elements.

iterations at the coarsest mesh; thus it pays to have a good initial guess when the finest mesh is reached.

The solution on the finest mesh is shown in Fig. 5.2 showing streamlines, pressure isobars and the initial coarse mesh for the solution. Notice the recirculating region behind the step. The interface between the recirculation region and the main flow is defined by a streamline leaving from the sharp corner of the step and reattaching to the channel wall at a certain point, marked with a triangle in the figure. The location of this *reattachment point* was determined by evaluating the derivative $\frac{\partial v_x}{\partial y}$ for a dense number of points on the channel wall and detecting a sign change in this quantity. The point is located a distance of $x_1 = 2.93 \times h$ from the step.

The accuracy of the final solution can be estimated by inspecting the maximal change in the solution upon each refinement as shown in Tab. 5.1. It is seen that the maximal relative change upon the third mesh refinement is of the order of 4%, indicating that the final solution is still not particularly accurate.

5.2 Adaptive mesh refinement

As was discussed above the final refinement level reached in the regular mesh refinement sequence is not sufficiently fine to resolve the solution accurately.

$ \Delta v_x _{\max}$	$ \Delta v_y _{\max}$	$ \Delta p _{\rm max}$	x_1/h	# refinements	CPU time [s]
			2.95	0	33.47
0.064	0.203	0.135	2.98	1	14.66
0.028	0.067	0.041	2.96	2	51.67
0.007	0.037	0.039	2.93	3	360.03

Table 5.1: Convergence upon regular mesh refinement at Re = 100: maximal relative change in the numerical solution upon successive mesh refinements. Also the location of the reattachment points is indicated.

However an additional regular mesh refinement would increase the number of mesh points to the limit of what can be handled using the MATLAB direct linear solver that we have employed.

Instead we shall now focus on the use of adaptive mesh refinement to obtain an accurate solution with much fewer elements. We use an elementwise error indicator of the form

$$e_{\alpha} = \left[C_{1} \left| \Omega_{\alpha} \right| \left\| \rho(\boldsymbol{v} \cdot \boldsymbol{\nabla}) \boldsymbol{v} + \mu \boldsymbol{\nabla}^{2} \boldsymbol{v} - \boldsymbol{\nabla} p \right\|_{\Omega_{\alpha}}^{2} + C_{2} \sum_{\Gamma \in \partial \Omega_{\alpha}} \left| \Gamma \right| \left\| \left[\frac{\partial \boldsymbol{v}}{\partial n} \right] \right\|_{\Gamma}^{2} + C_{3} \left\| \boldsymbol{\nabla} \cdot \boldsymbol{v} \right\|_{\Omega_{\alpha}} \right]^{1/2}$$
(5.1)

which is a simple generalization of the error indicator discussed in Sec. 3.3.2 [7]. The constants C_1, C_2 and C_3 are scaling factors, all of which were set to one in the present problem. We use the error indicator to refine elements where either the solution does not satisfy the Navier-Stokes equation or the incompressibility constraint² or it has jumps in the normal derivative of the velocity across the element boundaries. We select those elements for refinement where the error is larger than 25% of the maximal error found in the mesh.

Fig. 5.3 shows the solution after two successive adaptive mesh refinements. It is seen that even at this rather coarse mesh level, the solution agrees well with the result obtained using regular mesh refinement.

Fig. 5.4 shows the result after six successive mesh refinements. Evidently the solution is rather difficult to resolve close to the sharp corner of the step. The accuracy of the final solution can be estimated from Tab. 5.2 showing the maximal relative change upon successive adaptive mesh refinements. The same accuracy as for regular mesh refinement is obtained with much smaller computational cost.

Now one may ask how relevant it is to resolve the solution at the sharp corner in such fine detail. Inspecting the solution one finds that the pressure appears to have singularity at the sharp corner, which will be impossible to

²In the strong, not weak, sense, that is.



Figure 5.3: Solution for Re = 100 after two adaptive refinements. a): Streamlines. b): Pressure isobars. c): Adapted mesh. Notice that the streamlines and pressure contours are not particularly smooth – this is because of the rather coarse mesh used. Still the solution agrees rather well with the final solution obtained using regular mesh refinement.





The solution is almost indistinguishable from the one obtained with regular mesh refinement. Close to the corner we detect a second sign change in the y-derivative of the x-velocity; we believe this is an artefact.

$ \Delta v_x _{\max}$	$ \Delta v_y _{\max}$	$ \Delta p _{\rm max}$	x_1/h	# refinements	CPU time [s]
			2.99	0	36.76
0.021	0.096	0.154	2.99	1	8.11
0.049	0.097	0.025	2.99	2	6.37
0.021	0.062	0.016	2.99	3	10.58
0.012	0.040	0.025	2.97	4	16.90
0.010	0.049	0.024	2.97	5	36.41
0.008	0.027	0.018	2.97	6	73.02

Table 5.2: Convergence upon adaptive mesh refinement at Re = 100: maximal relative change in the numerical solution upon successive mesh refinements. Also the location of the reattachment points is indicated.

resolve at the singular point nomatter how fine the mesh is made. Of course, experimentally the corner will always be rounded rather than perfectly sharp below a certain length scale. This could be modelled by 'cutting off' a small snippet of the corner, which would effectively remove the singularity.

Still, physically we would not expect the very fine details of the geometry at the corner to influence the solution at some distance out into the main flow. Thus it is not necessary to resolve the singularity in order to obtain good overall accuracy of the solution, and we could choose a lower hard limit for the element sizes without compromising the overall accuracy.

5.3 Bifurcation

Now we consider the case Re = 500 and increase the channel dimensions to H = 2h, l = 8h and L = 40h, c.f. Fig. 5.1. Fig. 5.5 shows the solution after ten adaptive mesh refinements; only part of the rather long computational domain is shown. Notice the emergence of a second recirculation region in the flow.

Fig. 5.6 shows a scan over Reynolds numbers up to Re = 1250 showing the locations of the detachment and reattachment points. The data were produced using an older version of the code employing piecewise linear elements and a regular mesh. The figure should be compared to the low Re part of Fig. 5.7 showing the experimental results from Armaly *et al.* [19]. The qualitative agreement is good, in particular for the first reconnection point on the lower channel wall. The second recirculation region of the numerical results seems to be consistently located to far from the step. It should be kept in mind, though, that for Re > 400 the experimental system was no longer truly two-dimensional.



Figure 5.5: Solution for Re = 500 after ten adaptive refinements.

a): Streamlines. b): Pressure isobars. c): Adapted mesh.

Notice the emergence of a second recirculation region. The pressure has a deep local minimum just outside the step.



Figure 5.6: Simulated results for location of detachment and reattachment of the flow as a function of Reynolds number. •: Reattachment point on lower channel wall x_1 . \triangle and \blacktriangle : Detachment and reattachment points on upper channel wall x_4 and x_5 , c.f. Fig. 5.7.

The qualitative agreement is good, though x_4 and x_5 seems to be consistently too large; further x_5 fails to bend over at about Re = 800.



Figure 5.7: Experimental results for location of detachment and reattachment of the flow as a function of Reynolds number. From: B. F. Armaly, F. Durst, J. C. F. Pereira, B. Schönung, Experimental and theoretical investigation of backward-facing step flow, *J. Fluid Mech.* **127**, 473-496 (1983) [19]

Chapter 6

Non-Newtonian flow

The concept of non-Newtonian liquids is a broad one covering all cases where the deviatoric stress tensor τ is not simply proportional to the shear rate $\dot{\gamma}$. An important example of a non-Newtonian liquid is blood; moreover in flows on the microscale, blood reveals its inherent granular structure as a suspension of blood cells in a plasma. Both from a fundamental physiological point of view, but certainly also in relation to medico applications for blood analysis in lab-on-a-chip systems, it is important to gain a thorough understanding of how blood behaves on the microscale.

The blood flow in microchannels was the research topic for Lennart Bitsch during his Master thesis [20] and his subsequent work at MIC, where he has performed micro particle-image velocimetry (μ PIV) measurements on blood flow in thin glass capillaries.

In relation to this work I have been involved with simulations so as to predict accurately the flow patterns to expect from a description of blood as a continuous medium but with a shear rate dependent viscosity to model the non-Newtonian character of the blood. Further I have been involved in parts of the data analysis and in calculations concerning the optics in the experimental setup. The work has resulted in two conference proceedings [1, 2] and a paper submitted to *Experiments in Fluids* [3]. I have included the paper in the present thesis in Appendix D and discuss the experimental results in Sec. 6.3

6.1 Non-Newtonian liquids

Simple liquids composed from small simple molecules are typically Newtonian and examples include e.g. water, oil, and syrup. Non-Newtonian behaviour is typically observed for more complicated liquids such as polymeric solutions and suspensions of particles, e.g. paint, blood, ketchup, and cough medicine.

There are several ways in which liquids can deviate from Newtonian

behaviour. The most simple case is that of a liquid for which the viscosity is not constant but depends on the instantaneous shear rate. Such a liquid is termed a generalized Newtonian liquid and it obeys a deviatoric stress vs. shear rate relation of the form

$$\boldsymbol{\tau} = 2\mu(\dot{\boldsymbol{\gamma}})\dot{\boldsymbol{\gamma}},\tag{6.1}$$

where $\dot{\gamma}$ is the magnitude of the shear rate as defined in Eq. (2.15). Further, the liquids are classified as either shear thinning or shear thickening depending on whether $\mu(\dot{\gamma})$ decreases or increases with $\dot{\gamma}$.

In other cases it is found that below a certain yield stress τ_{yield} the substance ceases to flow and behaves as a solid rather than a liquid. This makes it more difficult to model the flow of such a liquid since there is not a one-toone correspondance between τ and $\dot{\gamma}$. Notice that in a direct measurement of the viscosity dependence on shear rate, the apparent viscosity defined as $\mu(\dot{\gamma}) \equiv \tau/2\dot{\gamma}$ will diverge for $\dot{\gamma} \to 0$ as τ remains finite in this limit.

Finally, so-called viscoelastic liquids exhibit both elastic and plastic properties at the same time. Such behaviour is typically observed for polymeric solutions in which the long and branched molecules tend to be entangled. When a fluid element is suddenly deformed, the entangled molecules interact strongly and exert a restoring force against the deformation; only in time as the stretched molecules deentangle and relax the restoring force disappears. This behaviour introduces a time dependence or a memory of deformation history to the liquid which makes it complex to model. Viscoelastic liquids can display rather extraordinary flow patterns, e.g. for flow through a narrow neck or into a sudden contraction of a channel. Whereas the streamlines of a Newtonian liquid tend to go straight into the contraction, the viscoelastic liquid can develop a vortex outside the contraction [21].

6.1.1 Blood

The properties of blood in shear flow are discussed in the textbook by Y. C. Fung [22]. Often blood can be adequately described as a generalized Newtonian liquid, although some models operate also with a yield stress for blood. Also it is known that blood can develop a thin lubrication layer next to solid walls that is free of the red blood cells otherwise present in the blood.

Red blood cells are disc-like but very flexible objects with a diameter of about 7 μ m. The volume fraction of red blood cells to the total volume of liquid is reported as the hematocrit value, and for humans it is typically in the range of 35% to 50% depending on sex, age, and physical condition. The remaining part of the blood is the blood plasma which consist of water with some proteins and salts dissolved in it. The blood used in the μ PIV experi-

ments was human blood obtained from the blood bank at Rigshospitalet; it was used undiluted having a hematocrit of approximately 60%.

At low shear rates the red blood cells tend to aggregate or stack. These large agglomerates of blood cells cause disturbances and long range interactions in the laminar flow profile of the plasma and therefore friction. At higher shear rates the agglomerates are torn apart, and the individual cells do not disturb the flow to the same extent as did the larger chunks, hence friction is lowered. At even higher shear rates the blood cells are observed to align parallel with the flow which decreases the viscosity further – this behaviour is particular of the blood cells and is not observed for e.g. rigid discs. To that end we also mention that a suspension of rigid spheres or discs is typically observed to stop flowing when the volume fraction of the suspended objects exceeds about 50%, whereas blood can maintain flow until hematocrits of 98%.

There is a range of models available for the viscosity of blood, but they are generally only valid within a limited range of shear rates and hematocrit values. Moreover they do not all agree particularly well. Often the models do not have a direct physical background but contain several fitting parameters; they are used clinically to make a diagnosis for a patient based on an examination of the patients blood.

Several models operate with a yield stress and of these the simple Casson model

$$\sqrt{\tau} = \sqrt{\tau_{\text{yield}}} + \sqrt{2\mu\dot{\gamma}} \tag{6.2}$$

is often found to agree rather well with experiments at low shear rates; here μ is the apparent viscosity at high shear rates and τ_{yield} is the yield stress which is typically found to be of the order of a few mPa. Measurements of blood viscosity at low shear rates is a complicated matter due to the small corresponding streses. Furthermore blood has a tendency to form a sticky skin when exposed to air which may disturb measurements, and there is some disagreement of what happens in this range. At high shear rates beyond $\dot{\gamma} \simeq 150 \text{ s}^{-1}$ it is generally found that the viscosity does not drop any further and thus that blood behaves Newtonian from this point.

For the blood used in the μ PIV experiments the apparent viscosity has been measured and the results are reproduced in Fig. 6.1. The measurements were performed with a concentric cylinders rheometer where the blood is confined between two concentric cylinders. A constant torque is applied to the inner cylinder and the angular velocity is measured. The velocity profile between the two cylinder surfaces is known as a Couette flow and if the gap between the cylinders is small compared to their radii the profile is approximately linear corresponding to constant shear rate.

In the range 0.95 s⁻¹ $< \dot{\gamma} < 140$ s⁻¹ where data are most dense and well behaved it has been found that the deviatoric stress vs. shear rate relation



Figure 6.1: Direct measurement of the apparent viscosity of blood as a function of shear rate [20]. The solid line corresponds to a least squares spline approximation within the data range; open circles are spline nodes. Outside the data range the viscosity is approximated as being constant. At high shear rates this is justified since the viscosity of blood is generally observed to stay constant beyond a shear rate of $\dot{\gamma} \gtrsim 150 \text{ s}^{-1}$. At low shear rates it is rather a matter of convenience since the fit is used as viscosity model in our finite element computations; infinite viscosity at zero shear rate would tend to confuse the numerical solution.

is well described by a power law model of the form

$$\tau = 2m\dot{\gamma}^n. \tag{6.3}$$

with prefactor 2m = 0.057 and exponent n = 0.623 [20].

At low shear rates the data are more sparse. Still the measurements from data set #1 appears to fall on a straight line with a slope corresponding to a power law with exponent n = 0.25.

6.2 Blood flow in straight channel

The μ PIV experiment was performed on blood flow in a long and straight glass capillary. Thus the flow is described as a Poiseuille flow as discussed Sec. 2.2; the channel is chosen to be aligned with the z-axis so that the velocity takes the form $\boldsymbol{v} = v_z(x, y)\boldsymbol{e}_z$. Then the Navier-Stokes equation reduces to a force balance between the viscous force and a constant pressure gradient in the channel from $p = p_0 - Gz$. Because the viscosity $\mu(\dot{\gamma})$ is not constant it cannot be taken outside the divergence of the deviatoric stress tensor as in Eq. (2.13) and the equivalent of Eq. (2.19) becomes

$$G + \left[\boldsymbol{\nabla} \cdot \boldsymbol{\tau}\right]_{z} = G + \left[\boldsymbol{\nabla} \cdot \left(2\mu(\dot{\gamma})\dot{\boldsymbol{\gamma}}\right)\right]_{z} = 0.$$
(6.4)

The only non-zero components of the shear rate tensor are $\dot{\gamma}_{xz} = \dot{\gamma}_{zx} = \frac{1}{2} \frac{\partial v_z}{\partial x}$ and $\dot{\gamma}_{yz} = \dot{\gamma}_{zy} = \frac{1}{2} \frac{\partial v_z}{\partial y}$ and the magnitude is $\dot{\gamma} = \frac{1}{2} \left[\left(\frac{\partial v_z}{\partial x} \right)^2 + \left(\frac{\partial v_z}{\partial y} \right)^2 \right]^{1/2}$ by Eqs. (2.11) and (2.15).

The problem is simplified further by considering that the glass capillaries used in the experiment were approximately of rectangular shape and of small aspect ratio $\alpha = H/W \simeq 1/10$. As was found in Sec. 2.2.2 the flow in the central part of the channel is then essentially one dimensional with $\boldsymbol{v} = v_z(y)\boldsymbol{e}_z$.

If we assume that a single power law model $\mu(\dot{\gamma}) = m\dot{\gamma}^{n-1}$ is valid for the blood viscosity at all shear rates then it is easily verified that the solution for the velocity profile also takes the form of a power law

$$v_z(y) = v_0 \left[1 - \left(\frac{2y}{H}\right)^{1+\frac{1}{n}} \right]$$
 (6.5)

with $v_0 = \frac{Hn}{n+1} \left(\frac{HG}{4m}\right)^{1/n}$. In the Newtonian case for n = 1 we can confirm that this reduces to the parabolic form of Eq. (2.25). This solution is plotted in Fig. 6.2 for the cases n = 0.25 and n = 0.623, corresponding to the low and intermediate shear rate regimes of Fig. 6.1, as well as for the Newtonian case n = 1 which is reached in the high shear rate regime. The figure shows that for n = 0.623 the velocity profile does not deviate much from a parabolic one. However for n = 0.25 corresponding to the low shear rate regime the profile is rather flat or blunted and it should certainly be possible to distinguish this one from a parabolic profile experimentally.

Fig. 6.2 also shows four profiles that are numerical solutions obtained with the FEM using the fit to the measured viscosity data as viscosity model. The pressure gradient was adjusted in the four cases so as to obtain solutions of maximal shear rates $\dot{\gamma}_{max}$ of 1000 s⁻¹, 100 s⁻¹, 10 s⁻¹ and 1 s⁻¹ respectively. In particular we notice that the profile for $\dot{\gamma}_{max} = 1000 \text{ s}^{-1}$ is almost parabolic; only in a narrow range close to the center is it more flat than the parabolic one, which corresponds to the range where the shear rate drops into the non-Newtonian regime. Also we notice that the cutoff introduced in the viscosity fit at very low shear rates does not influence any of the solutions shown significantly; only for $\dot{\gamma}_{max} \leq 0.05 \text{ s}^{-1}$ does the cutoff show up.

6.3 The μ PIV experiment

In short, the particle-image velocimetry technique (PIV) amounts to deriving the instantaneous velocity of small particles or beads suspended in the fluid by comparing two images of the flow recorded shortly after each other; see our paper in Appendix D for more experimental details. The actual image analysis is performed with dedicated statistical software to determine



Figure 6.2: Theoretical velocity profiles for blood flow in a flat capillary of thickness H. Solid lines: Velocity profiles from Eq. (6.5) for power law exponents $n = \{1.0, 0.623, 0.25\}$ corresponding to high, intermediate and low shear rate regimes in Fig. 6.1. All profiles are normalized to the same maximal velocity to allow for comparison of the shape of the profiles. Dotted lines: FEM solutions for the velocity profile using the spline fit to the meaured blood viscosity shown in Fig. 6.1. In four cases the pressure has been adjusted such that the maximal shear rates $\dot{\gamma}_{max}$ in the solutions are 1000 s⁻¹, 100 s⁻¹, 10 s⁻¹ and 1 s⁻¹ with the corresponding profiles ranging from an almost parabolic one to a rather flat or blunted shape.

peaks in the spatial cross-correlation function between the two images for the pixel gray value.

There was a significant level of noise in the measurements; however for steady flow it was found that by averaging the cross-correlation function for a number of image pairs, it was possible to obtain a good resolution for the velocity field in the focal plane.

In μ PIV the observations are made through a microscope connected to a CCD camera; thus the resolution in the direction normal to focal plane is defined by the focal depth of the microscope lens system. Ideally the setup should record only those particles exactly in the focal plane such that by moving the microscope stage up and down it would be possible to obtain a complete three dimensional map of the velcity field. However because the microscope has a finite depth of focus, also particles slightly out of focus will show up in the images recorded.

The first set of μ PIV measurements were performed on a suspension of latex beads in water and this experiment was meant to serve as a Newtonian reference with which to compare the results of the blood experiment. Fig. 6.3 shows a sketch of the cross section of the glass capillary used in the experiments. The images were recorded in a top view of the central part of



Figure 6.3: Sketch of cross section of glass capillary used in the μ PIV experiments. The channel width is approximately 360 μ m and the depth varies from 28 μ m in the central part of the channel to 32.5 μ m at the sides. Further, the cross sectional area is 1.09×10^{-8} m², the perimeter 7.6×10^{-4} m and the thickness of the glass wall approximately 25 μ m.

the channel and the velocity profile $v_z(y)$ was determined by scanning the focal plane along the y-axis. Because of the small aspect ratio we would expect the flow to be essentially a parabolic profile of the form Eq. (2.25). However it was found that the focal depth of the microscope was rather large compared to the thickness of the channel. Thus at any position of the focal plane y_{focal} inside the channel, particles from the entire depth of the channel showed up in the images, and the result obtained was effectively a channel average and therefore independent of y. Even up to 30 μ m outside the channel, particles were still visible though considerably blurred.

The problem was in a sense that the imaging software was performing too well in recognizing the motion of the blurred out-of-focus particles. This has been delt with by a filtering strategy where all gray values in the images below a certain base-clip or cutoff are deleted, leaving only the darkest and sharpest particles in the images. The profile thus obtained agrees well with the theoretical profile, as seen in Fig. 5 of the paper in Appendix D. The theoretical profile was obtained by matching a FEM solution for the Poiseuille flow in a channel of geometry as that in Fig. 6.3 with a flow rate of $Q = 50 \times 10^{-12}$ m³/s which was fixed in the experiment using a syringe pump.

The second set of μ PIV measurements were performed on blood, imaging the motion of the individual blood cells. The profiles obtained turn out to be almost completely flat with a constant level of $v_0 = 17$ mm/s, as shown in Fig. 6 of the paper in Appendix D.

The experiments were carried out at a flow rate of $Q = 167 \times 10^{-12} \text{ m}^3/\text{s}$ which corresponds to an average velocity in the channel of $\bar{v}_z = 15.3 \text{ mm/s}$; for a Newtonian liquid the maximal shear rate near the walls would reach 1500 s^{-1} at this flow rate. This means that the experiment is far into the high shear rate regime and we cannot attribute the flat profile observed to the bulk non-Newtonian character of blood with its shear rate dependent viscosity, c.f. Sec. 6.2.

We suggest that the observations rather indicate the formation of a lu-

brication layer next to the channel walls. By the cell-free nature of the lubrication layer we cannot expect to observe it directly in μ PIV using the blood cells as seeding. As a simplified model we assume that the flow can be described as an almost solid plug of blood cells surrounded by a low viscosity cell-free lubrication layer. Thus we assume that the velocity is constant within the plug at the level v_0 , while in the lubrication layer the velocity drops linearly from v_0 at the plug interface to zero at the walls.

Within this simple model we can estimate the thickness d of the cell-free layer from the nominal flow rate Q and the measured value for the velocity of the plug v_0 . The cross-sectional area of the channel was measured to be $A = 1.09 \times 10^{-8} \text{ m}^2$ and the length of the perimeter $\ell = 7.57 \times 10^{-4} \text{ m}$ so that the area of the lubrication layer is approximately $\ell \times d$. Since the average velocity in the lubrication layer is simply $v_0/2$, we can calculate the total volume flow rate in the capillary as $Q = v_0(A - \ell d) + v_0\ell d/2$. Solving this for the lubrication layer thickness we find

$$d = \frac{2}{\ell} \left(A - \frac{Q}{v_0} \right) \tag{6.6}$$

from which we calculate $d = 3 \ \mu \text{m}$.

A simple way to validate the result Eq. (6.6) of the simplified two-phase model could be obtained with the following energy consideration. The power P fed into the capillary is given simply in terms of the pressure drop Δp across the capillary times the flow rate as $P = \Delta p \times Q$; this energy is lost to friction in the liquid. In the simple model we assume that the shear rate and therefore also the friction is zero inside the plug. All the power is lost to the lubrication layer and we can calculate this power loss from the following formula

$$P_{friction} = \int_{\Omega} \mathrm{d}\boldsymbol{x} \, 2\mu |\dot{\gamma}|^2 = 2\mu L\ell d \times \left(\frac{v_0}{2d}\right)^2 \tag{6.7}$$

where L is the full length of the capillary and we used that inside the lubrication layer $\dot{\gamma} = v_0/2d$. Also μ is the viscosity in the lubrication layer which we can take to be equal to that of water, since this is what the blood plasma mainly consist of. Notice that even if the plug is not assumed to be entirely solid but only much more viscous than the lubrication layer, then by analogy with a small and a high resistance in parallel we would still find that most of the power is lost to the lubrication layer. Thus we infer that

$$\Delta p \times Q = 2\mu L\ell d \times \left(\frac{v_0}{2d}\right)^2 = \frac{\mu L\ell v_0^2}{2d} \tag{6.8}$$

which provides a means to validate the result for d. While there was actually a pressure sensor installed in the experimental setup, it turned out to be unreliable and showed considerable drift in otherwise steady state experimental circumstances. Therefore we have not been able to verify our two-phase model using the above expression.

Finally we would like to mention that an obvious way to deal with the problem of the finite depth of focus would be to take a side view rather than a top view of the capillary of Fig. 6.3. Then provided the capillary could be aligned accurately with the optical axis, the large depth of focus would pose no problem since the profile does not vary along the x-axis in the central part of the capillary.

This strategy was tried out, but the measurements turned out to contain considerably more noise than for the top view setup. Further we found it difficult to figure out how to treat the data such as to compensate for the refraction in the glass walls of the channel. Taking a simple ray optics approach we were not even able to conclude that a sharp image would actually be formed in the microscope with this setup. The standard way to deal with the problem of refractions is the use of an oil immersion lens, where the space between the lens and the object is filled with an oil with a refractive index close to that of the glass capillary and lens. However we did not pursue this strategy any further in the present experiment.

6.4 Summary

We have reviewed a few basic concepts in the theory of non-Newtonian liquids with particular focus on the properties of blood. In fairness I would like to stress that the actual experimental work that has been discussed in this chapter was carried out solely by Lennart Bitsch, while my contribution has been limited to simulations of the blood flow in one and two dimensions to be compared with the experimental results. Further I have been involved in numerous discussions with Lennart concerning the interpretation of the experimental data and have contributed to the data analysis by doing calculations on the optical properties of the experimental setup; in particular in calculating the profile that we would expect to observe experimentally given that the results are obtained using an optical system with finite focal depth.

The main conclusion that can be drawn from the experimental results is that blood flow in the microchannel cannot be adequately described as a continuous liquid. Rather a two-phase model should be employed describing the formation of a lubrication layer next to the channel walls, and we believe that a truely succesful model will have to take into account in some way the actual granular structure of the liquid at the length scales involved in the problem.

Chapter 7

Electroosmotic flow

In this chapter we discuss electroosmotic flow (EOF) in micro- and nanochannels. The basis for EOF is the formation of an electric double layer, the so-called Debye layer, at the channel walls. Chemical reactions at the wallliquid interface result in a transfer of charge from the liquid to the wall. The surface becomes charged leaving a surplus of ions in the liquid of opposite charge. These ions, however, tend to collect near the surface in a thin charged layer that screens the surface charge and leaves the bulk of the liquid charge neutral. If an external electrical field is applied along the channel, it will pull the charged screening layer ahead. The motion of the screening layer will induce a drag; this is the basic electroosmotic pumping, and the motion of the bulk fluid is called electroosmotic flow (EOF) [23].

The thickness of the screening layer is given by the Debye length λ_D that we shall define below, and typically it is of the order of 1-10 nm. If the depth of the channel is much larger than the Debye length, then the description of the flow in thin Debye layer can effectively be decoupled from the flow in the bulk of the channel. It is found that the EOF velocity of the bulk fluid is proportional to the magnitude of the electric field applied parallel to the wall with the constant of proportionality termed the electroosmotic mobility μ_{eo} . Thus the flow in the bulk of the channel can be modelled by replacing the standard no-slip boundary condition at the walls with a slip condition, where the velocity parallel to the walls is given as the electric field times the mobility $v_{\parallel} = \mu_{eo} E_{\parallel}$. This is the standard way in which electroosmotic flow is handled in e.g. COVENTOR and CFD-ACE+. In our FEM tool we could as well enforce such a condition by using the components **nx** and ny of boundary outward normal vector available when doing the boundary integrals with gquad - the velocity normal to the surface should vanish, $\boldsymbol{n} \cdot \boldsymbol{v} = 0$, whereas parallel to the surface we should have $\boldsymbol{t} \cdot \boldsymbol{v} = \mu_{eo} \boldsymbol{t} \cdot \boldsymbol{E}$, defining a unit vector parallel to the boundary as $t = [-n_u, n_x]$.

However, our focus in this chapter is rather on the case where the channel depth is comparable to the Debye length. Thus this has relevance to electroosmosis in porous structures with pore sizes below 100 nm. Such pore sizes are found in so-called frits which are used presently at MIC in electroosmotic pump design [24].¹

We have chosen to focus on the small scale problem because we find it challenging and also because we would like to demonstrate that or finite element tool is general enough to accommodate relatively easy for more complex problems appearing in microfluidics.

However the application of the tool turned out to be not as straightforward as we had hoped. In particular we encountered problems when trying to enforce some of the boundary conditions that we wished to impose on the problem considered. Whether this present shortcomings in the way that we have implemented the tool or rather correspond to inappropriate choises of boundary conditions is not quite clear to us. In any case, the results that we present, and which we must characterize as preliminary, represents just how far we got with EOF at the time of writing.

7.1 The Debye layer

We consider now a charged species dissolved in a liquid with concentration c_i and wish to determine the variation of c_i when moving from the bulk of the liquid to the channel walls. In thermodynamic equilibrium the chemical potential $\mu_{\text{chem},i}$ for the species must be constant throughout the system. For a dilute electrolyte the chemical potential is given as [25]

$$\mu_{\text{chem},i} = \mu_{\text{chem},i}^0 + k_B T \ln \frac{c_i(\boldsymbol{x})}{c_i^0} + z_i e \phi(\boldsymbol{x}), \qquad (7.1)$$

where $\mu_{\text{chem},i}^0$ is the reference chemical potential and c_i^0 is the reference concentration of the species; for convenience we choose the reference point to be somewhere out in the bulk of the liquid. Further z_i is the valence number of the ions and $\phi(\boldsymbol{x})$ is the electrostatic potential which we assume is zero at the reference point. Now in order for $\mu_{\text{chem},i}$ to be constant we must have

$$c_i = c_i^0 \exp\left[-\frac{z_i e\phi}{k_B T}\right],\tag{7.2}$$

¹The reason for considering porous materials in electroosmotic pumping is that they allow large pressures to be produced. For a given pressure drop across the pump, there will be a flow back through the pumping section. If we think of the porous structure as a bundle of circular capillaries then the volume flow rate through each capillary scales as R^4 as given in Eq. (2.23) whereas the number of capillaries grows as R^{-2} for a given pump cross section area and such that the total volume flow rate is proportional to R^2 . Thus if pumping section is filled with a porous material of small pore size, the backflow will be small. On the other hand, the electroosmotic mobility and thus the velocity at the capillary walls is independent of the pore size and therefore the electroosmotic flow rate depends only on the total cross-sectional area of the capillaries. This assumes, though, that the cross-section area taken up by the Debye layer in the individual capillaries is neglible.

that is, the reference concentration times a Boltzmann factor. The electrostatic potential is coupled to the total charge density via the Poisson equation

$$\nabla^2 \phi = -\frac{1}{\epsilon} \rho_e = -\frac{e}{\epsilon} \sum_i z_i c_i, \qquad (7.3)$$

where ϵ is the dielectric constant that we assume constant throughout the liquid for simplicity. Eqs. (7.2) and (7.3) result in a nonlinear differential equation for the potential ϕ . For now we assume that the electrostatic energy of the ions is much smaller than the thermal energy scale, that is $\frac{ez_i\phi}{k_BT} \ll 1$, which is known as the Debye-Hückel approximation. Then we can take $\exp(-\frac{ez_i\phi}{k_BT}) \simeq 1 - \frac{ez_i\phi}{k_BT}$ and Eq. (7.3) reduces to

$$\boldsymbol{\nabla}^2 \phi = -\frac{e}{\epsilon} \sum z_i c_i^0 \left(1 - \frac{e z_i \phi}{k_B T} \right) = \left[\frac{e^2}{\epsilon k_B T} \sum c_i^0 z_i^2 \right] \phi = \frac{1}{\lambda_D^2} \phi , \qquad (7.4)$$

where the second equality follows since we assume that the bulk liquid is charge neutral $\sum c_i^0 z_i = 0$ while the last equality identifies the characteristic length scale for variation of ϕ which we define as the Debye length

$$\lambda_D = \sqrt{\frac{\epsilon k_B T}{e^2 \sum c_i^0 z_i^2}} . \tag{7.5}$$

We notice that the Debye length increases with temperature as the diffusion of the ions increases and the screening becomes less efficient, whereas it decreases with the bulk charge carrier concentration. If we consider a specific example of a 0.001 mol/L solution of a simple salt with $z_i = \pm 1$ in water with $\epsilon = 78.3\epsilon_0$ at room temperature, we obtain $\lambda_D = 9.61$ nm. For comparision we mention that the average distance between like ions in a 0.001 mol/L solution is approximately 12 nm; thus it appears that the continuum model that we employ for the description of the Debye layer is not particularly well justified.

The potential difference from the bulk liquid to the wall is called the zeta potential, ζ , and is typically of the order of 1 - 200 mV. It is strongly dependent on the pH-value in the solution. Considering the common case of a siliconoxide (glass) channel in contact with an aqueous solution, the silanol SiOH groups on the surface react with the water to form either SiOH₂⁺ or SiO⁻ according to

$$SiOH + H_3O^+ \rightarrow SiOH_2^+ + H_2O$$
 (acid solution) (7.6)

$$SiOH + OH^- \rightarrow SiO^- + H_2O$$
 (basic solution). (7.7)

Thus in an acid solution the surface is negatively charge and the Debye layer positive yielding a positive ζ potential – and vice versa in a basic solution.

7.2 EOF in porous structure

We now consider the out-of-equilibrium case. The motion of the ions in the liquid has contributions from electromigration, where the ions migrate due to the imposed electrical field, diffusion and convection. The ion flux J_i is expressed in the Nernst-Planck equation [25]

$$\boldsymbol{J}_i = \mu_i z_i c_i \boldsymbol{E} - D_{\text{mass},i} \boldsymbol{\nabla} c_i + c_i \boldsymbol{v}, \qquad (7.8)$$

where \boldsymbol{E} is the electric field, μ_i is the mobility of species i, $D_{\text{mass},i}$ is the mass diffusion coefficient and \boldsymbol{v} is the velocity of the liquid. The mobility can be expressed as $\mu_i = D_{\text{mass},i}e/k_BT$, known as the Einstein-Smoluchowski relation. The flux enters the continuity equation for the *i*'th species

$$\frac{\partial c_i}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{J}_i = 0, \tag{7.9}$$

and in steady state we have simply $\nabla \cdot J_i = 0$. Further, the electrical force on the ions present in the liquid enters the Navier-Stokes equations as a body force term $\rho_e E$. We shall consider for simplicity the case of a salt with only two ions of concentrations c_+ and c_- and valence $z_+ = -z_- = 1$. Further we shall assume that the diffusion coefficients for the two species are equal $D_{\text{mass},+} = D_{\text{mass},-} = D$. The full system of governing equations that we wish to solve are the Navier-Stokes equation with the incompressibility constraint, the Poisson equation for the electrostatic potential and the continuity equations for the two species

$$\rho(\boldsymbol{v}\cdot\boldsymbol{\nabla})\boldsymbol{v} + \boldsymbol{\nabla}p - \mu\boldsymbol{\nabla}^2\boldsymbol{v} + e[c_+ - c_-]\boldsymbol{\nabla}\phi = 0$$
(7.10)

$$\boldsymbol{\nabla} \cdot \boldsymbol{v} = 0 \tag{7.11}$$

$$-\epsilon \nabla^2 \phi - e[c_+ - c_-] = 0 \tag{7.12}$$

$$-\frac{De}{k_BT}\boldsymbol{\nabla}\cdot(c_+\boldsymbol{\nabla}\phi) - D\boldsymbol{\nabla}^2 c_+ + (\boldsymbol{v}\cdot\boldsymbol{\nabla})c_+ = 0$$
(7.13)

$$\frac{De}{k_B T} \nabla \cdot (c_- \nabla \phi) - D \nabla^2 c_- + (\boldsymbol{v} \cdot \nabla) c_- = 0$$
(7.14)

where we used $\boldsymbol{E} = -\boldsymbol{\nabla}\phi$ and $\rho_e = e[c_+ - c_-]$ and further eliminated $\boldsymbol{\nabla} \cdot \boldsymbol{v}$ in the convective term in the last two equations. The equations are discretized using the Galerkin method of Sec. 3.2.1 to obtain

$$\rho \mathbf{C} \mathbf{v}_x - \mathbf{Q}_x^T \mathbf{p} + \mu \mathbf{K} \mathbf{v}_x + e \mathbf{Q}_{c,x} \boldsymbol{\phi} = \int_{\partial \Omega} ds \, \boldsymbol{\varphi} [n_x p - \mu (\boldsymbol{n} \cdot \boldsymbol{\nabla}) v_x] \qquad (7.15a)$$

$$\rho \mathbf{C} \mathbf{v}_y - \mathbf{Q}_y^T \mathbf{p} + \mu \mathbf{K} \mathbf{v}_y + e \mathbf{Q}_{c,y} \boldsymbol{\phi} = \int_{\partial \Omega} \mathrm{d}s \, \boldsymbol{\varphi}[n_y p - \mu(\boldsymbol{n} \cdot \boldsymbol{\nabla}) v_y] \qquad (7.15\mathrm{b})$$

$$\mathbf{Q}_x \mathbf{v}_x + \mathbf{Q}_y \mathbf{v}_y = 0 \tag{7.15c}$$

$$\epsilon \mathbf{K} \boldsymbol{\phi} - e \mathbf{M} \mathbf{c}_{+} + e \mathbf{M} \mathbf{c}_{-} = \int_{\partial \Omega} \mathrm{d}s \, \boldsymbol{\varphi}(\boldsymbol{n} \cdot \boldsymbol{\nabla}) \boldsymbol{\phi}$$
(7.15d)

$$-\frac{De}{k_BT}\mathbf{K}_{+}\boldsymbol{\phi} - D\mathbf{K}\mathbf{c}_{+} + \mathbf{C}\mathbf{c}_{+} = \int_{\partial\Omega} \mathrm{d}s \,\boldsymbol{\varphi}\boldsymbol{n} \cdot \left[\frac{De}{k_BT}c_{+}\boldsymbol{\nabla}\phi + D\boldsymbol{\nabla}c_{+}\right] \quad (7.15e)$$

$$\frac{De}{k_BT}\mathbf{K}_{-}\phi - D\mathbf{K}\mathbf{c}_{-} + \mathbf{C}\mathbf{c}_{-} = \int_{\partial\Omega} ds \,\boldsymbol{\varphi}\boldsymbol{n} \cdot \left[-\frac{De}{k_BT}c_{-}\boldsymbol{\nabla}\phi + D\boldsymbol{\nabla}c_{-}\right], \quad (7.15f)$$

where the column vector φ in the boundary integrals contains all the basis functions φ_k . Most of the matrices appearing are the same as in Chap. 4, but a few more have been introduced. They are

$$[\mathbf{Q}_{c,x}]_{k\ell} = \langle \varphi_k, [c_+ - c_-] \frac{\partial \varphi_\ell}{\partial x} \rangle$$
(7.16)

and similarly for $\mathbf{Q}_{c,y}$, whereas

$$[\mathbf{K}_{+}]_{k\ell} = \langle \boldsymbol{\nabla}\varphi_{k}; c_{+}\boldsymbol{\nabla}\varphi_{\ell} \rangle \tag{7.17}$$

and similarly for \mathbf{K}_{-} . In order to solve the nonlinear problem we employ Newtons method, which requires computation of the Jacobian matrix for the system, in which several other submatrices appear. The procedure is straightforward, though, and we shall not discuss it here.

The boundary conditions for the Navier-Stokes equations were discussed is Chap. 4. For the electrostatic potential we may either impose Dirichlet conditions, fixing the potential at electrodes, or Neumann boundary conditions fixing the gradient of ϕ , that is, the electric field, normal to the boundary. Often the potential on the channel walls is defined in terms of a fixed ζ potential, which is assumed to be obtainable experimentally by adjusting the pH-value of the solution. We shall rather choose to fix the surface charge on the walls and leave the potential to be defined. A realistic model should describe the interaction between the solution and the ions on the surface and thus let the surface charge as a function of the local chemical potential. However, we choose the surface charge to be constant for simplicity.

For the concentrations c_i there are several possible boundary conditions to be considered. On solid walls the normal flux $\mathbf{n} \cdot \mathbf{J}_i$ should vasish, and since the fluid velocity is zero on the walls we find that the boundary integral in Eqs. (7.15e) and (7.15f) vanishes. On boundaries far into the bulk liquid we may choose a Dirichlet condition fixing the concentrations at the reference values.

On high symmetry boundaries the normal derivatives of both ϕ and c_i will typically vanish. In other cases we may wish the concentration gradients to vanish, but still impose a finite electrical field, which should also be possible. However one should be careful when specifying Neumann conditions on *all* boundaries. For the potential such a condition corresponds fixing the electric field, and in particular by choosing the field such that all

field lines that go in, also must come out, it is possible to enforce global charge neutrality in the system. Of course, then the level of the potential is left undetermined, but this should not influence the physical properties of the system since the potential only enters the equations by its gradient and curvature. Still, it will pose a problem for the accuracy of the solution if the numerical solution ends up with an offset of 10^{24} or so, and to help this it may be wise to include a Dirichlet boundary condition for at least just a single point on the wall.

For the concentrations the problems with all Neumann conditions are more complicated since the solution *does* depend on the absolute level of the concentration. Also, if all Neumann conditions are specified for a domain deep inside a long pore, the system becomes effectively decoupled from the bulk fluid. One way to help this would be to use the thermodynamic equilibrium value of Eq. (7.2) on some points or parts of the boundary. However we believe that the most healthy way to define the problem that we have in mind, regarding the electroosmotic flow in a porous structure, is to couple it rigorously to the bulk liquid outside the pore.

7.2.1 Pore geometry

The geometry that we have chosen is shown Fig. 7.1; we assume a large array of parallel pores which by symmetry allows us to model only the small cell defined by the thick lines in Fig. 7.1. The pores are chosen not as circular channels but rectangular, extending far out of the paper, which makes the problem two-dimensional.² The dimensions of the channel are defined relative to the Debye length λ_D so that the thickness of the pore is $h = 3\lambda$ and the length $w = 6\lambda_D$, whereas the repeat distance of adjacent pores is $H = 7\lambda_D$ and the distance from the pore to the point where we define the bulk liquid is $L = 8\lambda_D$.

We consider an aqueous solution of concentration 0.001 mol/L at room temperature, yielding a Debye length of $\lambda_D = 9.61$ nm. The surface charge density at the walls is taken as $\sigma_e = 0.01 \epsilon_{\text{water}} / \lambda_D = 7.22 \times 10^{-4} \text{ C/m}^2$, which corresponds to the surface charge density of an infinite planar liquid glass interface with ζ potential $\zeta_0 = 10$ mV in the Debye-Hückel approximation.³ Finally the diffusion constant D is chosen as $D = 5.3 \times 10^{-9} \text{ m}^2/\text{s}$ corresponding to the value for OH⁻ in water.

The boundary conditions on the symmetry lines are homogeneous Neumann conditions for all variables except the *y*-velocity component which is

²Another obvious choise would be to consider parallel circular pores in a close-packed array. The symmetry is rather high, but the problem remains three dimensional unless one approximates it as being axisymmetric.

³The thermal energy scale $k_BT = 25.7$ meV is larger than $\zeta_0 = 10$ mV so we should expect the Debye-H2 uckel approximation to hold approximately. Solving the full nonlinear differential equation for the potential one obtains a potential difference of $\zeta_0 = 10.063$ mV from the bulk solution to the glass wall at an infinite planar liquid glass interface.



Figure 7.1: Geometry of the porous structure considered. The white area is open and filled with liquid whereas the gray area is the glass defining the channel. We assume a large array of parallel pores and model only the domain mared with thick lines. The grid corresponds units of the length scale $\lambda_D = 9.61$ nm. In the liquid domain all variables are defined whereas inside the glass we only compute the electrostatic potential.

zero there. On the glass-liquid interface we impose no-slip for the velocity and homogeneous Neumann for the concentrations, whereas the potential normal gradient is given a jump determined by the fixed surface charge as

$$\boldsymbol{n} \cdot (\epsilon_{\text{water}} \boldsymbol{\nabla} \phi_{\text{water}} - \epsilon_{\text{glass}} \boldsymbol{\nabla} \phi_{\text{glass}}) = \sigma_e \tag{7.18}$$

where the normal vector \boldsymbol{n} points from the liquid into the glass, and the dielectric constant of glass is taken as $\epsilon_{\text{glass}} = 1.5\epsilon_0$. On the bulk liquid boundary we fix the pressure at zero and the concentrations at the reference value 0.001 mol/L. Also we shall fix the potential at the bulk liquid boundary.

Fig. 7.2 shows the solution at zero applied potential. The maximum value of the potential is 10.5 mV, which is obtained on the wall at the channel center. Notice that this is 0.5 mV higher than the value ζ_0 used to define the surface charge σ_e . The charge concentration, defined as $c_e = c_+ - c_-$, follows the potential closely – as it should according to Eq. (7.2).

Fig. 7.3 shows the solution when a potential of 5.3 mV is applied on the left bulk fluid boundary while the right is kept at 0 mV, corresponding to an average electric field of $E_0 = 2.5 \times 10^4$ V/m. The potential is distorted, whereas the change in the charge distribution is hardly visible when comparing with Fig. 7.2 – the maximal charge displacement is of the order of 0.5%. The potential displacement $\delta\phi$ denotes the change in potential from equilibrium and Fig. 7.3 shows that the applied potential drop is somewhat concentrated onto the pore. Yet, when considering that the cross-sectional area of the conducting electrolyte inside the pore is approximately half that outside the pore, it should be no surprise that the field is twice as strong inside the pore.



Figure 7.2: Solution at zero applied potential. The maximum electrostatic potential of 10.5 mV is obtained on the wall at the channel center. Also there the charge concentration is minimal with -8.4×10^{-4} mol/L whereas in goes rapidly to zero in the bulk liquid.

The maximal velocity is 0.157 mm/s which is 80% of the electroosmotic velocity defined as $v_{\rm eo} = \mu_{\rm eo} E_0 = 0.195$ mm/s where $\mu_{\rm eo} = \epsilon_{\rm water} \zeta_0/\mu$ is the electroosmotic mobility [25].



Figure 7.3: Solution with potential drop across system of 5.3 mV corresponding to 2.5×10^4 V/m. The potential- and charge displacements $\delta\phi$ and δc_e displays the difference change from the equilibrium value. The charge displacement is negative to the left of the pore (close to the positive electrode) and positive to the right (close to ground) with a maximum of $\delta c_e = 1.4 \times 10^{-6}$ mol/L.

Chapter 8

Conclusion

Computational fluid dynamics is an important tool in microfluidics, providing detailed information when theoretical models and experimental results are unavaiblable or difficult to obtain. We have developed a simulation tool in MATLAB based on the finite element method (FEM) allowing complex problems in microfluidics to be approached.

We have analysed elementary flow in channels of different cross sections, comparing the flow rate in a rectangular channel to that in channels of triangular and Gaussian shaped cross section. Such channel geometries appear typically in microfluidic networks fabricated in polymer substrates using the laser ablation technique.

The FEM is a powerful and popular simulation method in many branches of physics and engineering. We have given a short overview of important concepts in the FEM with special focus on aspects central to our implementation. Also we have discussed specific features available with our tool, including support of basis functions of arbitrary order in one and two dimensions, adaptive solution strategy to resolve well the difficult parts of a problem, and fast iterative solution using the multigrid method. However we have had problems applying the multigrid method to fluid dynamics.

Two different schemes for the problem of incompressible flow have been considered, representing different ways of dealing with the spurious oscillatory pressure mode that occurs in the most straightforward aplication of the FEM. The implementation has been tested on the classical problem of flow over a backward-facing step, and the qualitative agreement with experimental results from the litterature was found to be good.

The concept of non-Newtonian liquids has been introduced, and in particular we have discussed bloow flow in a capillary, modelling blood as a generalized Newtonian liquid with shear rate dependent viscosity. The theoretical predictions for the velocity profile turned out not to agree with experimental results for blood flow in a thin glass capillary, which could be attributed to the granular nature of blood at the microscopic level. In relation to the experimental work I have also been involved in parts of the data analysis, in articular in predicting the profiles to observe experimentally given that the observations were made with a microscope of finite focal depth. This work has resulted in two conference proceedings [1, 2] and a paper submitted to *Experiments in fluids* [3].

Finally electroosmotic flow in a porous structure has been considered. While there are many interesting details to study in such a system, we did not have the time to get very far with this problem. Still we have demonstrated that our FEM tool can accomodate rather complex problems appearing microfluidics.

LAURITS HØJGAARD OLESEN, c971820

Appendix A

Computation of curvature

In order to evaluate the plane integrals arising when the Galerkin method is used to discretize a FEM problem we split the integrals into elementwise contributions and subsequently map the individual elements from the xyplane to a simple reference element in the integration coordinate $\xi\eta$ -plane, as discussed in detail in Sec. 3.4.

We refer to the coordinate transform $\boldsymbol{\xi}(\boldsymbol{x})$ mapping from a particular element Ω_{α} in the *xy*-plane to the reference element in the $\boldsymbol{\xi}\eta$ -plane as the *inverse transform* since it is actually the transform $\boldsymbol{x}(\boldsymbol{\xi})$ that we know explicitly. The element basis function $\varphi_k(\boldsymbol{x})$ on Ω_{α} is related to $\phi_k(\boldsymbol{\xi})$ on the reference element as

$$\varphi_k(\boldsymbol{x}) = \phi_k(\boldsymbol{\xi}(\boldsymbol{x})). \tag{A.1}$$

In Sec. 3.4 we found the gradient

$$\begin{bmatrix} \frac{\partial \varphi}{\partial x} \\ \frac{\partial \varphi}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix} = \frac{1}{|\boldsymbol{J}|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}, \quad (A.2)$$

where |J| is the determinant of the Jacobian matrix for the coordinate transform $J = \begin{bmatrix} \frac{\partial x}{\partial \xi} \end{bmatrix}$.

We now wish to express also the curvature of the element basis function $\nabla^2 \varphi_k(\mathbf{x})$ in terms of the reference element basis function and coordinate

transform. Application of the chain rule is straight-forward leading to

$$\frac{\partial^{2} \varphi_{k}}{\partial x^{2}} = \frac{\partial^{2} \phi_{k}}{\partial \xi^{2}} \left(\frac{\partial \xi}{\partial x} \right)^{2} + \frac{\partial^{2} \phi_{k}}{\partial \eta^{2}} \left(\frac{\partial \eta}{\partial x} \right)^{2} + 2 \frac{\partial^{2} \phi_{k}}{\partial \xi \partial \eta} \frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial x} + \frac{\partial \phi_{k}}{\partial \xi} \frac{\partial^{2} \xi}{\partial x^{2}} + \frac{\partial \phi_{k}}{\partial \eta} \frac{\partial^{2} \eta}{\partial x^{2}} \tag{A.3a}$$

$$\frac{\partial^{2} \varphi_{k}}{\partial y^{2}} = \frac{\partial^{2} \phi_{k}}{\partial \xi^{2}} \left(\frac{\partial \xi}{\partial y} \right)^{2} + \frac{\partial^{2} \phi_{k}}{\partial \eta^{2}} \left(\frac{\partial \eta}{\partial y} \right)^{2} + 2 \frac{\partial^{2} \phi_{k}}{\partial \xi \partial \eta} \frac{\partial \xi}{\partial y} \frac{\partial \eta}{\partial y} + \frac{\partial \phi_{k}}{\partial \xi} \frac{\partial^{2} \xi}{\partial y^{2}} + \frac{\partial \phi_{k}}{\partial \eta} \frac{\partial^{2} \eta}{\partial y^{2}} \tag{A.3b}$$

$$\frac{\partial^{2} \varphi_{k}}{\partial x \partial y} = \frac{\partial^{2} \phi_{k}}{\partial \xi^{2}} \frac{\partial \xi}{\partial x} \frac{\partial \xi}{\partial y} + \frac{\partial^{2} \phi_{k}}{\partial \eta^{2}} \frac{\partial \eta}{\partial x} \frac{\partial \eta}{\partial y} + \frac{\partial^{2} \phi_{k}}{\partial \xi \partial \eta} \left(\frac{\partial \xi}{\partial x} \frac{\partial \eta}{\partial y} + \frac{\partial \xi}{\partial y} \frac{\partial \eta}{\partial x} \right) + \frac{\partial \phi_{k}}{\partial \xi} \frac{\partial^{2} \xi}{\partial x \partial y} + \frac{\partial \phi_{k}}{\partial \eta} \frac{\partial^{2} \eta}{\partial x \partial y} \cdot \tag{A.3c}$$

In these expression most of the terms are readily available – the second derivatives of ϕ_k with respect to ξ and η can be computed since the polynomial coefficients for ϕ_k are known explicitly. Also the first derivatives of the inverse transform $\left[\frac{\partial \xi}{\partial x}\right]$ can be computed from Jacobian $\left[\frac{\partial x}{\partial \xi}\right]$ as in Eq. (A.2). However the second derivatives of the inverse transform remains a problem.

In case of triangular elements with a linear coordinate transform the second derivatives simply vanish – otherwise we obtain for the last two terms in each of the above expressions

$$\frac{\partial \phi_k}{\partial \xi} \frac{\partial^2 \xi}{\partial x^2} + \frac{\partial \phi_k}{\partial \eta} \frac{\partial^2 \eta}{\partial x^2} = \frac{1}{|\mathbf{J}|} \left\{ \frac{\partial \phi_k}{\partial \xi} \left[\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \eta} \right) \right] - \frac{\partial \phi_k}{\partial \eta} \left[\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \xi} \right) \right] - \frac{\partial \varphi_k}{\partial x} \frac{\partial |\mathbf{J}|}{\partial x} \right\} \quad (A.4a)$$

$$\frac{\partial \phi_k}{\partial \xi} \frac{\partial^2 \xi}{\partial y^2} + \frac{\partial \phi_k}{\partial \eta} \frac{\partial^2 \eta}{\partial y^2} = \frac{1}{|\mathbf{J}|} \left\{ \frac{\partial \phi_k}{\partial \eta} \left[\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \xi} \right) \right] - \frac{\partial \phi_k}{\partial \xi} \left[\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \eta} \right) \right] - \frac{\partial \varphi_k}{\partial y} \frac{\partial |\mathbf{J}|}{\partial y} \right\} \quad (A.4b)$$

$$\frac{\partial \phi_k}{\partial \xi} \frac{\partial^2 \xi}{\partial x \partial y} + \frac{\partial \phi_k}{\partial \eta} \frac{\partial^2 \eta}{\partial x \partial y} = \frac{1}{|\mathbf{J}|} \left\{ \frac{\partial \phi_k}{\partial \eta} \left[\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \xi} \right) \right] - \frac{\partial \phi_k}{\partial \xi} \left[\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \eta} \right) \right] - \frac{\partial \varphi_k}{\partial y} \frac{\partial |\mathbf{J}|}{\partial x} \right\}, \quad (A.4c)$$
where the following expressions are needed

$$\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \xi} \right) = \frac{\partial \xi}{\partial x} \frac{\partial^2 x}{\partial \xi^2} + \frac{\partial \eta}{\partial x} \frac{\partial^2 x}{\partial \xi \partial \eta}$$
(A.5a)

$$\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \eta} \right) = \frac{\partial \xi}{\partial x} \frac{\partial^2 x}{\partial \xi \partial \eta} + \frac{\partial \eta}{\partial x} \frac{\partial^2 x}{\partial \eta^2}$$
(A.5b)
$$\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \eta} \right) = \frac{\partial \xi}{\partial x} \frac{\partial^2 y}{\partial \xi \partial \eta} + \frac{\partial \eta}{\partial x} \frac{\partial^2 y}{\partial \eta^2}$$
(A.5b)

$$\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \xi} \right) = \frac{\partial \zeta}{\partial x} \frac{\partial}{\partial \xi^2} + \frac{\partial \eta}{\partial x} \frac{\partial}{\partial \xi \partial \eta}$$
(A.5c)

$$\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \eta} \right) = \frac{\partial \xi}{\partial x} \frac{\partial^2 y}{\partial \xi \partial \eta} + \frac{\partial \eta}{\partial x} \frac{\partial^2 y}{\partial \eta^2}$$
(A.5d)

$$\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \xi} \right) = \frac{\partial \eta}{\partial y} \frac{\partial^2 x}{\partial \xi \partial \eta} + \frac{\partial \xi}{\partial y} \frac{\partial^2 x}{\partial \xi^2}$$
(A.5e)

$$\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \eta} \right) = \frac{\partial \eta}{\partial y} \frac{\partial^2 x}{\partial \eta^2} + \frac{\partial \xi}{\partial y} \frac{\partial^2 x}{\partial \xi \partial \eta}$$
(A.5f)
$$\frac{\partial}{\partial y} \left(\frac{\partial y}{\partial \xi} \right) = \frac{\partial \eta}{\partial y} \frac{\partial^2 y}{\partial \xi \partial \eta} + \frac{\partial \xi}{\partial y} \frac{\partial^2 y}{\partial \xi^2}$$
(A.5g)

$$\frac{\partial}{\partial y} \left(\frac{\partial y}{\partial \eta} \right) = \frac{\partial \eta}{\partial y} \frac{\partial^2 y}{\partial \eta^2} + \frac{\partial \xi}{\partial y} \frac{\partial^2 y}{\partial \xi \partial \eta}$$
(A.5h)

and finally

$$\frac{\partial |\mathbf{J}|}{\partial x} = \left[\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \xi} \right) \right] \frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \xi} \left[\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \eta} \right) \right]
- \left[\frac{\partial}{\partial x} \left(\frac{\partial x}{\partial \eta} \right) \right] \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \eta} \left[\frac{\partial}{\partial x} \left(\frac{\partial y}{\partial \xi} \right) \right]$$
(A.6a)
$$\frac{\partial |\mathbf{J}|}{\partial y} = \left[\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \xi} \right) \right] \frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \xi} \left[\frac{\partial}{\partial y} \left(\frac{\partial y}{\partial \eta} \right) \right]
- \left[\frac{\partial}{\partial y} \left(\frac{\partial x}{\partial \eta} \right) \right] \frac{\partial y}{\partial \xi} - \frac{\partial x}{\partial \eta} \left[\frac{\partial}{\partial y} \left(\frac{\partial y}{\partial \xi} \right) \right].$$
(A.6b)

Notice that x and y appears asymmetric in Eq. (A.4c). Of course the result should be independent of the order in which we choose to differentiate with respect to x and y. However we do not see an obvious way of making the expression in Eq. (A.4c) symmetric without doubling the number of terms and thus we rely on the result to evaluate correctly even if it is not aestetically pleasing.

Appendix B

Computation of streamlines

In this appendix we discuss the visualization of a flow pattern with streamlines and explain how to compute streamlines for two-dimensional flows using the stream function.

A streamline is defined as a curve that is everywhere tangent to the velocity field and it can be thought of as the path that a small fluid particle follows as it rides along with the flow[26]. This makes a plot of streamlines usefull to identify separation regions in the flow as done in Chap. 5, or e.g. to understand diffusion when two chemical solutions are streaming together.

For unsteady flows one distinguishes between streamlines, streaklines and particle orbits. A particle orbit displays the actual path in space traversed by a particular particle, whereas a streakline is the trace observed when smoke or dye is injected through a small tube into the flow in e.g. a wind tunnel. A streamline is neither of these but rather depict the velocity field at a single instant of time. For steady flows though they are all equivalent.

The streamline is parametrized as $\boldsymbol{x}(s)$ where in steady flow the parameter s is equivalent to time spent by the fluid particle. Being defined as everywhere tangent to the velocity field the streamline obeys

$$\frac{d\boldsymbol{x}(s)}{ds} = \boldsymbol{v}\big(\boldsymbol{x}(s), t\big) \tag{B.1}$$

where \boldsymbol{v} is the velocity field and t is the instant of time being depicted. This is an ordinary differential equation and picking some point as $\boldsymbol{x}_0 = \boldsymbol{x}(s_0)$ it is readily solved as an initial value problem. MATLAB provides several functions for computing a set of streamlines given a velocity field and a set of streamline starting points; however they all require the velocity solution to be given on a square or rectangular type grid. Thus the FEM solution has to be evaluated on a such a grid before the MATLAB functions can be used – this is inconvenient since the solution will be degraded unless a grid that is finer than the finite element mesh is employed, in particular if adaptive mesh refinement has been employed to resolve critical regions of the solution better. Moreover for flow over a backwards facing step we found that in the recirculation region behind the step the streamlines produced by MATLAB were slowly spiralling rather that forming closed curves, indicating that a velocity sink was present inside the region. We believe that this artefact was caused by numerical errors in the algorithm due to the fact that the FEM velocity solution is not strictly divergence free and from numerical noise introduced when transfering the solution from the finite element mesh to the square grid.

Streamlines from the stream function

Streamlines never cross, and never begin nor end except at inflow and outflow boundaries and stagnation points. These properties would be ensured in a streamline plot if we could draw the streamlines as the contours of some function. This function turns out to be the stream function ψ and it exists only for divergence free flows.

For a given two-dimensional velocity field we define $\psi(\boldsymbol{x},t)$ by

$$\frac{\partial \psi}{\partial x} = -v_y, \quad \text{and} \quad \frac{\partial \psi}{\partial y} = v_x.$$
 (B.2)

By construction the gradient of ψ is everywhere normal to the velocity field $\boldsymbol{v} \cdot \boldsymbol{\nabla} \psi = 0$, such that a curve of constant ψ corresponds exactly to a streamline. Further the local density of the streamlines when drawn at equispaced levels of ψ is proportional to the magnitude of the velocity since $|\boldsymbol{\nabla} \psi| = |\boldsymbol{v}|$.

Since Eq. (B.2) only defines ψ up to an additive constant we must pick some reference point \boldsymbol{x}_0 and choose $\psi(\boldsymbol{x}_0) = \psi_0$. Then ψ may be determined for any \boldsymbol{x} by a line integral

$$\psi(\boldsymbol{x}) = \psi_0 + \int_{\boldsymbol{x}_0}^{\boldsymbol{x}} \mathrm{d}\boldsymbol{\ell} \cdot \boldsymbol{\nabla}\psi = \psi_0 + \int_{\boldsymbol{x}_0}^{\boldsymbol{x}} \mathrm{d}\boldsymbol{\ell} \cdot \hat{\boldsymbol{v}}, \tag{B.3}$$

where $\hat{\boldsymbol{v}} = [-v_y, v_x]$. Now in order for ψ to be single-valued the integral along any closed loop must vanish. This is satisfied only if \boldsymbol{v} is divergence free

$$0 = \oint d\boldsymbol{\ell} \cdot \hat{\boldsymbol{v}} = \int_{S} d\boldsymbol{\mathcal{S}} \, \boldsymbol{\nabla} \times \hat{\boldsymbol{v}} = \int_{S} d\boldsymbol{\mathcal{S}} \, \boldsymbol{\nabla} \cdot \boldsymbol{v}, \tag{B.4}$$

where S is the area enclosed by the integration path, and the second equality is Stokes theorem while the third follows from the two-dimensionality of the flow.

Our first approach to the computation of the stream function was based on Eq. (B.3). Once a solution for the velocity field was available, the stream function was computed by line integrals along element edges starting from some reference node and recursively propagating ψ to the entire mesh. However the FEM velocity field solution is not strictly divergence free; it is only the projection onto the basis functions $\langle \varphi_k, \nabla \cdot v \rangle$ that is zero. Thus the integral around the edges of a single element is not necessarily zero, which means that the stream function as determined by line integration along the element edges depends on which integration path is taken.

The results obtained with the line integration routine were actually not that bad, reflecting the fact that the FEM velocity field if not strictly divergence free is still close to be so. However, following [16] we have taken another approach, determining the stream function by solving a FEM Poisson equation. We take the divergence of Eq. (B.2) and obtain

$$\nabla^2 \psi = \frac{\partial v_x}{\partial y} - \frac{\partial v_y}{\partial x}.$$
 (B.5)

which is a Poisson equation for ψ ; and it is easily verified that the solution to Eq. (B.5) obeys $\nabla \psi = \hat{v}$ provided such a ψ exist.

The choices for boundary conditions is between Dirichlet and Neumann types. From Eq. (B.3) we deduce that ψ is constant along solid walls since $\boldsymbol{v} = 0$ there; thus a constant Dirichlet condition can be applied to solid walls. The wall levels can be connected since across an inflow or outflow boundary Γ_q , where the velocity profile is specified such that the volumetric discharge $Q = \int_{\Gamma_q} ds \, \boldsymbol{n} \cdot \boldsymbol{v}$ is known, the jump in ψ between the two walls on either side of Γ_q is simply $\Delta \psi = Q$.

However we have obtained the best results applying Neumann conditions to the entire boundary of the type

$$(\boldsymbol{n}\cdot\boldsymbol{\nabla})\psi = \boldsymbol{n}\cdot\hat{\boldsymbol{v}} = n_y v_x - n_x v_y$$
 (B.6)

computed explicitly from the known solution for \boldsymbol{v} and the boundary outward normal. All Neumann boundary conditions only determine the value of ψ up to an additive constant which leaves the linear system obtained by discretizing Eqs. (B.5) and (B.6) weakly singular. To resolve this, either the equation stemming from the projection of Eq. B.5 onto φ_0 can be deleted and replaced with $\psi(\boldsymbol{x}_0) = \psi_0$ before solving the system; or the problem can simply be ignored in which case the solution for ψ comes out with an arbitrary off-set. The off-set can subsequently be removed, fixing $\psi(\boldsymbol{x}_0) = \psi_0$ at the reference point. The former choise has the inconvenience that in a sense the correctness of the solution is sacrified at the reference point; with the latter choise there is the risk that the linear solver settles on a very large off-set so that the number of significant digits in the solution is reduced.

We have implemented the stream function computation as a simple routine strmfunc, see Appendix C, that determines a piecewise linear approximation to the stream function for given a velocity field. Streamlines are then drawn as a contour plot of ψ with gcontour.

Appendix C

Matlab function headers

In this appendix we include the headers for the MATLAB routines discussed in the thesis. The header comment is displayed as help message when e.g. the command help elm2sd is executed in the MATLAB command prompt – thus the headers are intended to give a quick overview of what the routine does. We hope they will be instructive to read.

We have not included all files though, and some of the headers therefore refer to routines not present in this appendix.

C.1 gaussgeom

GAUSSGEOM geometry M-file for half gaussian profile
The half gaussian profile is defined as the region
 (x,y) | 0 < x < 2*w & 0 < y < h*[exp(-x^2/(2*w^2))-exp(-2)]/[1-exp(-2)]
where w and h is the halfwidth and height of the gaussian.
Segment no.
 (1) [x(s),y(s)] = [w*s,0]
 (2) [x(s),y(s)] = [w*(1-s),h*[exp(-w*(1-s)^2 ...]]
 (3) [x(s),y(s)] = [0,h*(1-s)]
w and h are are set persistently [for the session] by a four argument
 call to gaussgeom([],[],w,h); e.g. gaussgeom([],[],2.0,4.0)
See also PDEGEOM for required syntax of geometry M-files</pre>

C.2 elm2sd

ELM2SD lookup of unique sides between elements

[s,es,ts,qs] = ELM2SD(e,t,q) where e is the edge table and t and q are the triangle and quadrangle tables of the mesh, returns the unique line segments connechting the mesh nodes together with their locations on the edge and as element sides. That is qs is a table of the indices in s of the four line segments making up q. Further qs is negative where the ordering of the line segment in s is reversed with respect to counterclockwise ordering in q whereas it is positive when the line segment is aligned.

See also INITMESH

C.3 halfmesh

HALFMESH Regular refinement of a mesh

[p1,e1,t1,q1] = HALFMESH(geom,p,e,t,q) refines the mesh in [p,e,t,q] by cutting every side in two. Edge points follow the boundary described by the Geometry M-file geom.

[p1,e1,t1,q1] = HALFMESH(geom,s,p,e,es,t,ts,q,qs) uses an existing set of unique sides in stead of generating a new with ELM2SD.

[s1,p1,e1,es1,t1,ts1,q1,qs1] = HALFMESH(geom,s,p,e,es,t,ts,q,qs) returns also a set of refined level unique sides although NOT those that are obtainable with ELM2SD(e1,t1,q1).

See also REFINEMESH, ELM2SD, PROLONG

C.4 prolong

PROLONG prolongation and restriction operators

[P,R] = PROLONG('linear',s,p,t,q,p1,t1,q1) builds the prolongation and restriction operators for transfer in linear interpolation between the coarse mesh [p,e,t,q] and the fine mesh [p1,e1,t1,q1] when this is obtained from the coarse using HALFMESH.

[P,R] = PROLONG('constant',s,p,t,q) builds operators for elementwise constant interpolation.

[P,R] = PROLONG('quadratic',s,p,{t,tt},{q,qq},p1,{t1,tt1},{q1,qq1}) builds operators for quadratic interpolation where tt and qq are the quadratic element tables, see QUADRATIC.

Notice that one is typically adviced to choose the restriction operator such that the inner products (v, R*w1) = (P*v, w1) for all v and w1 on the coarse and fine meshes respectively. This is equivalent to the matrix equation M*R = P'*M1 where M and M1 are the coarse and fine mesh mass matrices. However while P is a sparse matrix $R = M \setminus P'*M1$ is generally dense. To cure this, lumped mass matrices are introduced which are simply the diagonal parts of the full matrices, and

thus the sparsity pattern of P is recovered in R. A different approach is to use simply unit mass matrices. This works. More

on that topic later. See also HALFMESH

C.5 lagrange

LAGRANGE element interpolation

```
phi = LAGRANGE(n, 'line', xi) evaluates the n'th order Lagrange element
interpolation function and its slope on the interval [-1,1] for the
specified values of xi.
phi = LAGRANGE(n, 'triangle', xi, eta) evaluates the n'th order Lagrangle
element interpolation function and its slope on the interval
[ 0 < xi < 1 & 0 < eta < 1-xi ].
phi = LAGRANGE(n, 'quadrangle', xi, eta) evaluates the n'th order Lagrange
element interpolation function and its slope on the interval
[-1,1]x[-1,1].
[ee,tt,qq] = LAGRANGE(n, 'table', e, t, q) builds the n'th order edge and
element tables from the linear ones.
[ee,tt,qq] = LAGRANGE('table', e, t, q, s, es, ts, qs) does the same provided
```

a set of unique element sides in stead of generating one with ELM2SD.

See also ELEMENTS, ELM2SD

C.6 mglin

MGLIN linear multigrid

MGLIN(P,K,f,A,a) applies full linear multigrid to find the solution u of a linear(ized) FEM problem discretized with the Galerkin method to form a set of linear equations K*u = f where K is the stiffness matrix and f is the load vector.

Neumann boundary conditions should be included in K and f, whereas Dirichlet boundary conditions should be projected onto the basis to form a Dirichlet boundary mass matrix A and constraint vector a A*u = a

The Dirichlet b.c. is enforced by replacing all rows in K and f where the diagonal element in A is non-zero with the corresponding rows in A and a.

Finally a cell array of prolongation operators P connecting finest mesh, on which K, f, A and a are discretized, with a sequence of coarser meshes should be provided. E.g. use PROLONG to obtain the prolongation operator connecting a refined mesh from HALFMESH with the original one.

MGLIN(P,K,f,A,a,u) uses an initial guess u for the solution on the finest mesh in stead of employing full multigrid to ramp up a starting guess based on the exact solution on the coarsest mesh.

See also PROLONG, HALFMESH

C.7 gquad

GQUAD elementwise Gauss quadrature

```
I = GQUAD(f,n,element,p,t) performs n'th order Gauss quadrature over the
elements of type element, defined by the node table p and element table
t, of the function f where f is a vectorized string expression of
 coordinate(s) - e.g. on 2D elements it should depend on 'x' and 'y'.
Elements supported are:
                p:[1xN] t:[3xM] coordinate: 's'
   'line'
   'triangle' p:[2xN] t:[4xM] coordinates: 'x' and 'y'
'quadrangle' p:[2xN] t:[5xM] coordinates: 'x' and 'y'
The last row in the element tables is the subdomain label which is not
 actually used by GQUAD but must be present. See INITMESH.
For boundary integrals in two dimensions the element 'boundary' is
 available. Then the node table p should contain the x and y coordinates
 of the boundary nodes and the element table t should be replaced by the
boundary table e, see INITMESH. The integrand f is allowed to depend on
 the boundary parametrization 's' as well as 'x' and 'y' and also the
 components 'nx' and 'ny' of the boundary unit outward normal.
With n'th order quadrature the error is of
  O[h^(n+1)] for triangular elements
  O[h^(2n)] for line and quadrangular elements
I = GQUAD(f,n,element,p,{t,u}) and
I = GQUAD(f,n,element,p,{t,u1,u2,..}) evaluates the fields u1 and u2 and
their slopes u1x, u1y, u2x and, u2y on the quadrature points using the
 linear or bilinear element basis function in GEVAL. Thus f is now allowed
to depend on those fields.
I = GQUAD(f,n,element,p,t,proj) employs GPROJ to performs projection onto
the element basis functions where proj may be [] or any of the strings
                  int[ f*phi(i) ]dx
   'phi'
   'dphi'
                  int[ f*nabla(phi(i)) ]dx
 with i ranging over element nodes - e.g. for triangular elements the
result is [3xM] or two [3xM] arrays for 'dphi'.
More projetions:
                  int[ f*phi(i)*phi(j) ]dx
   'phi.phi'
                                                           [mass matrix]
   'dphi.dphi'
                  int[ f*nabla(phi(i)).nabla(phi(j)) ]dx [stiffness matrix]
   'phi.dphi'
                  int[ f*phi(i)*nabla(phi(j)) ]dx
 with first i and then j ranging over element nodes - e.g. for triangular
 elements i = [1 2 3 1 2 3 1 2 3] and j = [1 1 1 2 2 2 3 3 3].
I = GQUAD(f,n,element,p,t,proj,phi) projects onto the basis function phi
in stead of the default linear or bilinear. See ELEMENTS.
I = GQUAD(f,n,element,p,t,proj,{phi,tphi,v1,v2,..}) evaluates the fields
v1 and v2 on the integration points using phi in GEVAL.
```

```
I = GQUAD(f,n,element,p,t,proj,phi,psi) allows for hybrid interpolation.
That is the projections are changed to
   'phi.phi'
                 int[ f*phi(i)*psi(j) ]dx
   'phi.dphi'
                  int[ f*phi(i)*nabla(psi(j)) ]dx
   'dphi.dphi'
                  int[ f*nabla(phi(i)).nabla(psi(j)) ]dx
I = GQUAD(f,n,element,p,{t,u},proj,{phi,tphi,v},{psi,tpsi,w}) evaluates
all of the fields u, v, and w with the various basis functions.
I = GQUAD(f,n,element,p,t,proj,phi,psi,{gamma,tgamma,g}) evaluates the
field(s) g and both its slope and curvature gx, gy, gxx, gxy, and gyy in
GEVAL. While generally the curvature should not enter the integrand it
does so when the local error estimate is computed.
Examples:
Triangle element areas
  A = gquad('',1,'triangle',p,t)
Volume below x<sup>2</sup>+y<sup>2</sup> across the mesh [using 2nd order quadrature]
  V = gquad('x.^2+y.^2',2,'quadrangle',p,q)
Volume below sin(y) across the mesh
                                       [using 5th order quadrature]
  V = gquad('sin(y)',5,'triangle',p,t)
Projection of sin(x+y) onto element functions
  f = gquad('sin(x+y)',5,'triangle',p,t,'phi')
Next step is to assemble the contributions from different nodes using
  F = full( sparse(t,ones(size(t)),f,length(p),1) )
Projection of sin(u) onto element basis functions
  f = gquad('sin(u)',5,'triangle',p,{t,u},'phi')
where u is a column vector of expansion coefficients of the field u on
the default linear basis.
Continuity matrices int[ phi(i)*nabla(phi(j)) ]dxdy
   [Qx,Qy] = gquad('',1,'triangle',p,t,'phi.dphi')
Convection matrix int[ phi(i)*[a*d/dx + b*d/dy]*phi(j) ]dxdy
where a and b are expansion coefficients of velocity field data
   [Cx,Cy] = gquad('{u1,u2}',3,'quadrangle',p,{q,a,b},'phi.dphi')
  C = Cx + Cy
Stiffness matrix int[ nabla(phi(i)).nabla(phi(j)) ]dxdy with quadratic
basis functions
  K = gquad('',2,'triangle',p,t,'dphi.dphi','quadratic');
See also INITMESH, ELEMENTS, GPROJ, GEVAL
```

C.8 gjmp

GJMP square of jump in normal derivative across element boundaries

- GJMP(n,element,p,t,u) integrates the square of the jump in normal derivative of u across element boundaries using n'th order quadrature [where n=1 is sufficient for the default linear or bilinear element interpolation]. The result is multiplied by the length of the element boundary segment as required for use with error estimation.
- GJMP(n,element,p,t,{phi,tphi,u}) uses phi for the interpolation of u in stead of the default linear or bilinear one, see ELEMENTS.
- [d2u,ds] = GJMP(n,element,p,t,u) is a split representation where the integrated square jump d2u and the length of the sides ds are returned separately. This could be used for element refinement that adapts the new element

flexibility to the direction where the solution varies most rapidly?

See also GQUAD, GEVAL, ELEMENTS

C.9 geval

GEVAL evaluate field on quadrature points

GEVAL(u,element,t,phi) evaluates u on the elements of type element in the table t using the interpolation phi which should already be evaluated on the desired points in (xi,eta) space - see ELEMENTS.

[u,ux,uy] = GEVAL(u,element,t,phi,j) also evaluates the gradient of u using the coordinate transform jacobian - see GJAC.

[u,ux,uy,uxx,uxy,uyy] = GEVAL(u,element,t,phi,j,chi,p,tchi) also evaluates the curvature of u where the element coordinate transform chi and mesh nodes p are required only when chi is not linear and j thus not constant.

See also ELEMENTS

C.10 newt

NEWT solve non-linear system of equations by Newton iteration

```
[u,r] = NEWT(func,u0) looks for a solution u to the problem r(u) = 0 using
uO as initial guess when func is the name of a function that computes the
residual vector r(u) and Jacobian matrix J = d(r)/du. Thus NEWT calls
    [r,J] = feval(func,u)
to evaluate r and J.
Newtons method is based on linearization of r(u) close to u0
   r(u0+du) = r(u0) + J(u0)*du + ...
 where we solve r(u0+du) = 0 for du
   du = - J(u0) \setminus r(u0).
Provided u0 is close enough to the solution the iteration of this scheme
converges fast towards u. However if the solution is not close to u the
 iterations often diverge. Therefore the scheme is modified to read
    du = -J r
   u(new) = u(old) + alpha*du
 and a step length 0 < alpha < 1 is searched out to minimize the square
residual F(u) = |r(u)|^2. This procedure is called a line search with
du as the search direction.
An exact line search to track up a true minimum of F is quite expensive
 and not worth the effort. In stead NEWT implements an Armijo-Goldstein
 in-exact line search where alpha is chosen as the first number in the
 sequence [1 1/2 1/4 1/8 ...] with F(u+alpha*du) <= (1-alpha/2)*F(u).</pre>
For large problems the solution of the linear system du = -J r may
become quite expensive. Iterative solution of the system using MGLIN is
done by calling NEWT with a function func that returns a the Dirichlet
b.c. problem separately as required by MGLIN such that
    [r,J,rd,Jd] = feval(func,u)
where r(u) represents the problem in the interior of the domain and
rd(u) the Dirichlet b.c. problem with J = d(r)/du and Jd / d(rd)/du.
A number of options can be set as parameter/value pairs:
                                                             Default
    Name
                   Description
    '#newton'
                    maximal number of Newton iterations
                                                              8
    '#alpha'
                    maximal number of line search steps
                                                              8
    'reltol'
                    relative error tolerance for du
                                                              0
                                                              0
    'abstol'
                     absolute error tolerance for du
                                                              0
    'restol'
                     absolute error tolerance for norm(r)
    'parameters'
                     cell array of parameters passed to func {}
    'multigrid'
                     cell array of prolongation operators
                                                              {}
    'verbose'
                     flag to signal convergence info
                                                              'off'
Examples:
 Solve r(u) = 0 when r and J are defined by the function myfunc(u,a,b,c).
The solution is accepted when sum(r.^2) < 1e-10 allowing 20 iterations
 to search for it and printing convergence info during iteration
```

[u,r] = newt('myfunc',u0,'verbose','on','parameters',{a,b,c}, ... '#newton',20,'restol',1e-10)

C.11 flow2dpgp

```
FLOW2DPGP incompressible 2D [planar] flow
   equal order interpolation for all variables; stabilized with PGP
   [R,J] = FLOW2DPGP(u,dirichlet,neumann,rho,mu,p,e,t,q,phi,ee,tt,qq)
   returns the residual R and the Jacobian J = dR/du where
       R(u) = [ rho*(v.nabla)v + nabla(p) - mu*(nabla.nabla)v
                nabla.v + tau*((nabla.nabla)p - nabla.pgp)
                nabla(p) - pgp ]
    The trial solution u is stored in a 5 column table u = [vx vy p pgpx pgpy]
    where p is the pressure [not to be confused with node table p..]
   Dirichlet and Neumann type boundary conditions are specified with cell
   arrays of strings that evaluate to functions of the coordinates (x,y)
    and the boundary parametrization s.
   Example:
   dirichlet = {[1:2 5],'0','0',''
                                         % zero vx and vy on boundaries 1:3 and 5
                 [4], '', '0', ''
                                         % zero vy on boundary 4
                 [6],'s.*(2-s)','0',''}; % parabolic vx on boundary 6
    neumann = {[3],'0*nx','0*ny',''};
                                        % zero pressure on boundary 3
   [R,J,Rd,Jd] = FLOW2DPGP(u,dirichlet,neumann,rho,mu,p,e,t,q,phi,ee,tt,qq)
    does not eliminate the Dirichlet boundary conditions but returns separate
   Dirichlet boundary mass matrix and residual as required by MGLIN.
  Stabilization with pressure gradient projection from
   Int. J. Numer. Meth. Fluids 37, 419 (2001)
```

```
See also ELEMENTS, NEWT
```

C.12 gcontour

GCONTOUR Contour plot of function on element mesh

GCONTOUR(u,p,t,q) is a contour plot of the function u defined on the element mesh [p,t,q].

GCONTOUR(u,p,t,q,n) when n is an integer specifies the number of contours and when n is a vector the level of the contours.

Contours are colored from the current colormap; thus black contours are obtained with a precall to colormap(white-1). Otherwise the contours may be postprocessed with the graphics handle:

h = GCONTOUR(...) returns a vector of handles to the plotted contours.

[h,c] = GCONTOUR(...) returns also a contour matrix for use with CLABEL. Not yet though...

See also GPLOT, CONTOUR, CLABEL, COLORMAP, COLORBAR

C.13 strmfunc

STRMFUNC stream function for two dimensional incompressible velocity field

psi = STRMFUNC(u,p,e,t,q) computes the stream function psi from a two dimensional velocity field u = [vx vy] so that

nabla(psi) = [-vy vx]

Thus the gradient of psi is always orthogonal to u. This implies that contour lines of psi are the streamlines of the velocity field.

Notice that the stream function is only well defined when the velocity field is divergence free, nabla.u = 0. Otherwise some streamlines should or at least could end somewhere in the middle of the flow whereas a contour line certainly cannot.

```
STRMFUNC actually determines psi by solving a Poisson problem (nabla.nabla)psi = d(vx)/dy - d(vy)/dx
```

with Neumann boundary conditions n.nabla(psi) = ny*vx-nx*vy on all boundaries. If the velocity field is truly divergence free the solution can be shown to satisfy nabla(psi) = [-vy vx]. All homogeneous Neumann boundary conditions can be obtained by providing an empty edge table e.

```
[K,f] = STRMFUNC(u,p,e,t,q) returns the stiffness matrix K and source
term f for the poisson problem such that psi can be computed as
psi = K \ f
Notice that because we impose Neumann b.c. on all boundaries, psi is
only well defined up to an additive constant. This may render K weakly
singular, a problem that may be resolved by clamping first element in
psi to zero
psi = zeros(length(p),1);
psi(2:end) = K(2:end,2:end) \ f(2:end);
```

psi = STRMFUNC({u,e2,t2,q2},p,e,t,q) computes psi using (bi)quadratic interpolation for the velocity field.

See also GCONTOUR

Appendix D

Paper submitted to Experiments in Fluids

Title

Micro particle-image velocimetry of bead suspensions and blood flows

Authors

Lennart Bitsch, Laurits H. Olesen, Carsten H. Westergaard, Henrik Bruus, Henning Klank, and Jörg P. Kutter

Abstract

We present and discuss velocity profiles of microflows obtained by micro particle-image velocimetry (μ PIV) in a transmission setup. We measured on suspensions of beads in water and on human blood, using the red blood cells as a natural particle seeding. We analyze the limitations imposed by our optical system on the spatial resolution normal to the focal plane, the so-called focal depth. The first direct observations of the influence of the focal depth on the observed velocity profiles are presented. Good agreement is obtained between observations and calculated profiles modified by the finite focal depth through a visibility function.

Submitted

Submitted to Experiments in Fluids on 8 July 2003

Micro particle-image velocimetry of bead suspensions and blood flows

L. Bitsch¹, L. H. Olesen¹, C. H. Westergaard², H. Bruus¹, H. Klank¹, and J. P. Kutter¹ *

¹ Mikroelektronik Centret (MIC), Technical University of Denmark (DTU), DK-2800 Kongens Lyngby, Denmark

² Dantec Dynamics A/S, Tonsbakken 16-18, DK-2740 Skovlunde, Denmark

Submitted 8 July 2003

Abstract We present and discuss velocity profiles of microflows obtained by micro particle-image velocimetry (μ PIV) in a transmission setup. We measured on suspensions of beads in water and on human blood, using the red blood cells as a natural particle seeding. We analyze the limitations imposed by our optical system on the spatial resolution normal to the focal plane, the so-called focal depth. The first direct observations of the influence of the focal depth on the observed velocity profiles are presented. Good agreement is obtained between observations and calculated profiles modified by the finite focal depth through a visibility function.

1 Introduction

The efforts to develop lab-on-a-chip devices have increased substantially during the past few years[1]. Along with this development the techniques to characterize the performance of microfluidic systems are being improved drastically. Volume illuminated micro particle-image velocimetry (μ PIV) has shown to be a promising technique for characterizing detailed velocity profiles in these structures.

An important sub-field in microfluidics concerns the behavior of blood flowing in microchannels, and it has therefore become of high interest to obtain detailed information about the properties of such flows. In particular the non-linear flow properties of blood are interesting. To date, the studies of blood flow in microsystems presented in the literature often focus on measurements and models of the relation between flow rates and pressure drops[2], without utilizing the support of PIV. One exception is Sugii et al. who presented an in vivo PIV experiment of blood flow [3]. However, they did not investigate the influence of velocity gradients normal to the focal plane, and such an investigation is generally needed. Furthermore, the measurements were not compared to theoretical profiles.

In this work, using μ PIV in a transmission setup, we measure velocity profiles of bead suspensions and blood flows in a flat glass capillary with a roughly rectangular cross-section of size 28 μ m by 360 μ m. Restricted to observations in a given focal plane μ PIV has previously proven successful in both transmission and epifluorescent mode [4,5]. We extend these results by taking into account the finite spatial resolution normal to the focal plane, the so-called focal depth, imposed by our optical system, and we present the first direct observation of the influence of the focal depth on the obtained experimental velocity profiles. Our analysis is based on the theoretical expression for the visibility of particles slightly out-of-focus that has recently been derived by Olsen and Adrian^[6]. For fluids with a high density of particles, such as blood, it can be difficult to determine the position of boundaries parallel to the focal plane. We show that these boundaries are related to a steep increase in the size of the errorbars of the measurements. The measurements on blood strongly indicate that blood in these dimensions flows as a plug flow and that it should be modelled as a two-phase flow. Moreover, we find indications of the presence of a 3 μ m wide cell free boundary layer.

The paper is organized as follows. In Sec. II we describe the bead suspensions and blood samples used in our experiments, with special attention to the non-linear flow properties of the latter. In Sec. III we present the details of our μ PIV setup. The central concept of focal depth is explained in Sec. IV. Sec. V contains a description of the μ PIV data analysis. In Sec. VI, based on the finite focal depth, we derive the theoretical expression for the convolved velocity field. Our results are presented and discussed in Sec. VII.

^{*} The authors thank Detlef Snakenborg from MIC for providing the electronic circuitry to help in controlling the LED, and the Polymer Center at DTU for making it possible to perform the viscosity measurements. This work was supported by the Danish Technical Research Council, μ TAS Frame Program Grant No. 26-00-0220.



Fig. 1 The viscosity of blood as a function of shear rate obtained in a concentric cylinder geometry. With its shear rate dependent viscosity, blood is clearly a non-Newtonian fluid. The straight lines indicate the different power laws at high and low shear rates.

2 Preparation of Samples

For the measurements we used human blood delivered from the hospital blood bank at Rigshospitalet, Copenhagen[7]. More precisely, the blood was a suspension of red blood cells in an aqueous solution of adenine, mannitol, sodium chloride, and glucose. The role of the suspending medium is to serve as an energy source for the biochemical processes in the red blood cells and to prevent coagulation and hemolysis. However, since in this work we are mostly interested in the non-linear flow properties of blood, and since a major cause to this nonlinearity is the deformation and flow alignment of the red blood cells[8], we expect a good resemblance between our blood samples and anticoagulated whole human blood on this matter. The blood was stored at 4°C before use, but all experiments were conducted at room temperature, 20° C.

The non-linear flow properties of blood are clearly seen in its viscosity. It is more pronounced at high concentrations of red blood cells, and for that reason we chose to work with the undiluted suspension with a hematocrit at about 60%, where a normal human hematocrit is about 50% depending on gender and physical condition. In Fig. 1 the viscosity η of a blood sample is shown as a function of the shear rate $\dot{\gamma}$. The viscosity measurements were conducted with a rotational rheometer from TA instruments (AR 2000 Advanced Rheometer). The viscosity shows a strong dependency on the shear rate in agreement with other studies of blood[8]. Instead of a constant η characteristic for a Newtonian liquid, we see that blood for low shear rates follows a power law $\eta \propto \dot{\gamma}^{-0.75}$, which then tapers off at higher shear rates, $\eta \propto \dot{\gamma}^{-0.2}.$

As the blood sample had a tendency to develop a sticky skin when exposed to air, we chose to use a concentric cylinder geometry, where a relatively large sample volume minimizes the influence of a skin. The distance between the two cylinders was 1 mm, and the measured viscosity is therefore a bulk property. The viscosity will vary between individual blood donors. However, since we use the measured blood viscosity for a qualitative estimate of the suitability of a bulk property model, we are not concerned with such variations, but instead only focus on the qualitative behavior.

As a reference fluid to blood we chose to work with a suspension of spherical beads in pure water. The bead seeding was about 1% by volume, and each bead had a diameter of 1.02 μ m. The measured viscosity of the bead suspension was 10% larger than the viscosity of pure water, and no shear rate dependency was observed.

3 The measurement setup for μPIV

The measurement setup for μ PIV is sketched in Fig. 2(a). It is centered around an optical microscope (DMLB, Leica) using a CCD camera to record images of the flow of blood or bead suspensions through the capillary. The capillary is placed horizontally in the microscope and illuminated from below by a high intensity light-emitting diode (LED) with a maximum intensity at a wavelength of 450 nm. The transmitted light is focused onto a CCDcamera by a dry infinity-corrected Pl fluotar objective lens with a magnification of 63 and a numerical aperture of 0.7. Camera and LED were synchronized and controlled by a FlowMap^(TM) System Hub from Dantec Dynamics.

The CCD-camera is a PCO SensiCam with a 12 bit cooled imaging system containing a matrix of 1280×1024 pixels. By calibration in air this matrix was found to correspond to a field of view of $129 \times 103 \ \mu m^2$.

A cross section of the capillary is seen in Fig. 2(b). During experiments it was filled with the fluid from a syringe pump through approximately 20 cm long PVC tubes. After passing the capillary, the fluid ended up in a waste beaker at atmospheric pressure. The flow rates were 50 nL s⁻¹ for the bead suspension and 167 nL s⁻¹ for blood.

An experiment consisted of a series of PIV measurements, where the horizontal focal plane was moved vertically down through the channel, as sketched in Fig. 2(b). In that way the velocity gradient could be resolved in the direction normal to the focal plane.

A PIV measurement consisted of 25 pairs of images recorded with intervals of 1 s. The time lapse between the two images in a pair was 500 μ s, and the exposure time for each single image was given by the 100 μ s LED pulses. The exposure time and time between pulses were chosen in such a way that a reasonable signal to noise ratio was obtained on the measured velocities. To minMicro particle-image velocimetry of bead suspensions and blood flows



Fig. 2 (a) A sketch of the experimental setup containing the microfluidic system, the interrogation volume, the LED, the optical system, and the CCD camera. (b) The actual channel cross section in the yz plane obtained by optical inspection. The dimensions are $H = 32.5 \ \mu\text{m}$, $h = 28 \ \mu\text{m}$ and $w = 360 \ \mu\text{m}$. The z-axis is normal to the focal plane. The velocity profiles were scanned by moving the focal plane (dotted line) between z_{\min} and z_{\max} , 35 μ m above and 25 μ m below the channel, respectively.

imize the uncertainty on the travelled distance, the ratio between the exposure time and time between pulses should be as small as possible. We have not optimized this procedure in a systematic way, but rather used a rough guideline that requires a movement of 10 pixels between subsequent exposures and a ratio of time between pulses and exposure time of 5.

From each pulse pair an instantaneous velocity field of the particle images could be calculated. As an example the velocity field for a blood flow measurement is seen in Fig 3. Since we were studying steady state flow the velocity field was constant, and 25 sets of images were recorded and used to present an averaged result with improved statistics.

4 Focal depth and focal plane

The spatial resolution normal to the focal plane, the socalled focal depth $D_{\rm f}$, was estimated experimentally to be 4 μ m. Here $D_{\rm f}$ is defined as the full-width at halfmaximum in a plot of the recorded intensities of a given bead at the bottom of a water-filled channel as a function of the vertical position of the horizontal focal plane, as shown in Fig. 4. The intensity signal is seen to be superimposed on a flat background without any refraction patterns. This background is indicated by the lower dotted line.

To minimize effects from out of focus particles we use a so-called base-clipping technique. A specific threshold intensity level is chosen (the upper dotted line in Fig. 4) and intensities below this level are discarded. In the case of a particle inside the channel, base-clipping discards all patterns with an intensity corresponding to a distance further than 2 μ m away from the focal plane. For particles on a glass slide in air the focal depth was estimated to be at least 20 μ m, which indicates a dependency on the illumination, i.e., the focal depth is not the same when the focal plane is inside and outside the channel.



Fig. 3 An example of a velocity field obtained for a flow measurement on blood. The map is 92 μ m wide and 88 μ m high. The blood cells are observed as dark rings, and the average velocity of some interrogation areas are indicated with an arrow.

Fig. 4 The recorded intensity (open circles) in arb. units of a given bead in a water-filled capillary as a function of the vertical distance z from the horizontal focal plane. The full line is a fit to a Lorentzian intensity profile. The full-width at half-maximum is 4 μ m, which defines the focal depth $D_{\rm f}$. The upper and lower dotted horizontal lines are the base-clipping level and background level, respectively.

The optical effects of changing the position of the focal plane need special attention. The change $\Delta z_{\rm fp}$ in vertical position of the focal plane relative to the capillary is obtained by moving the objective table with the capillary a distance $\Delta z_{\rm cap}$ in steps of 1 μ m using the built-in scale on the microscope. In air we simply have $\Delta z_{\rm fp} = \Delta z_{\rm cap}$, but in the fluid the light rays are bent away from the vertical direction by refraction. Using the theory of paraxial rays we find the linear scaling

$$\Delta z_{\rm cap} = \frac{n_{\rm water}}{n_{\rm air}} \Delta z_{\rm fp}, \qquad (1)$$

where $n_{\text{water}} = 1.33$ and $n_{\text{air}} = 1$ are the refractive indices of water and air, respectively.

$5 \ \mu PIV$ data analysis

Data were analyzed using standard PIV-analysis[9]. Typically in PIV an illuminated volume is projected onto a plane, i.e, a particle image is produced on a CCD-chip. In order to obtain a velocity field the image is divided into smaller areas called interrogation areas. The density of particles or, more precisely, the density of refraction patterns determines the minimum resolution within the image plane. The resolution normal to the plane is dependent on the focal depth $D_{\rm f}$ after base-clipping. The interrogation area together with $D_{\rm f}$ defines the interrogation volume.

A rule of thumb requires approximately 10 particles per interrogation volume to enable so-called ordinary cross-correlation analysis. A particle suspension of 1% by volume, and an interrogation volume given by $6.4 \times 6.4 \times 2 \ \mu m^3$ corresponds roughly to 1.2 spherical particles. Hence, our particle seeding was a little too low, but we can compensate for this problem by making an average correlation, as explained further down. In order to improve the spatial resolution we used a 25% overlap between the interrogation areas.

The recorded light intensity I in a given interrogation area can, without going into details, be written as a function of the pixel position vector \mathbf{s} as $I_1(\mathbf{s})$ and $I_2(\mathbf{s})$ for the two light pulses at time t_1 and t_2 , respectively. The cross-correlation function $R(\Delta \mathbf{x})$ is defined as the average over all pixel coordinates in the interrogation area i as

$$R_i(\Delta \mathbf{x}) \equiv \langle I_1(\mathbf{s}) I_2(\mathbf{s} + \Delta \mathbf{x}) \rangle_i \tag{2}$$

The value $\Delta \mathbf{x}_i$ of $\Delta \mathbf{x}$ that maximizes $R_i(\Delta \mathbf{x})$ is a statistical measure of the overall displacement of the fluid represented by the interrogation area. Thus the average flow velocity of that region is given by

$$\mathbf{v}_i = \frac{\Delta \mathbf{x}_i}{t_2 - t_1}.\tag{3}$$

From each set of particle images we can determine the velocity field. However, in the case of a low seeding it is advantageous to make an average over all 25 image sets in a measurement series, and then make a cross-correlation. This procedure is called average crosscorrelation.

6 Theoretical velocity fields

The measured velocity fields from the μ PIV analysis need to be compared with theoretical velocity fields. These are found by solving the Navier-Stokes equation with a constant negative pressure gradient along the flow direction and with the no-slip boundary conditions for the velocity at the channel walls.

For the bead suspension, being a Newtonian liquid, the calculated Navier-Stokes velocity field $v_{\rm NS}$ can be thought of as the well-known paraboloid Poiseuille profile being distorted at the boundary to fit the dumbbell shape depicted in Fig. 2b rather than a circular shape.

The non-Newtonian character of blood, as seen in Fig. 1, leads to a further distortion of the Poiseuille flow in the form of a more blunt velocity profile close to the center of the capillary. Due to the small dimension of the channel compared to the size of the red blood cells, another option is to model the fluid as a two-phase system.

However, in our μ PIV experiments we expect to observe neither of these two simple Navier-Stokes velocity fields. Even though, we have performed a truncation in intensity levels, the remaining optical depth of our system will cause a pick-up of out-of-focus signals. Recently, Olsen and Adrian derived a theoretical expression for the



Micro particle-image velocimetry of bead suspensions and blood flows



Fig. 5 The velocity of the bead suspension versus z. (a) The z-axis has been centered around the middle of the channel. Legends: $(-\circ)$ first experiment v_{PIV} , (-*) second experiment v_{PIV} , and (-) Navier-Stokes theory $v_{\text{NS}}(z)$. Experimental uncertainties are indicated with error bars. (b) The weighted theoretical profile $v_{\text{conv}}(z)$ obtained from a convolution of the Navier Stokes profile $v_{\text{NS}}(z)$ with the Lorentzian visibility function I(z) of Eq. (4) and Fig. 4. Legends: (-) theory $v_{\text{conv}}(z)$, $(-\cdot)$ theory $v_{\text{NS}}(z)$, and $(-\circ)$ first experiment v_{PIV} . Theoretical uncertainties are indicated with error bars. A fair qualitative agreement is achieved between experiment and theory.



Fig. 6 Measurements on blood flows. (a) The velocity (upper curves) and the uncertainty (lower curves) as a function of z. The z-axis has been centered around the middle of the channel. The channel boundaries (top and bottom) are indicated by vertical lines at $z = \pm 14 \ \mu$ m. Legends: (- \circ) first experiment, (-*) second experiment. The uncertainty increases strongly outside the boundaries. (b) A comparison between the suggested two-phase model and the experimental results. Legends: (-) two-phase model, (- \circ) first experiment, (-*) second experiment.

visibility of particles slightly out-of-focus [6]. A particularly simple limiting case of their result is a Lorentzian visibility function I(z) given by

$$I(z) = \left(1 + \frac{4z^2}{D_{\rm f}^2}\right)^{-1},\tag{4}$$

where I is the intensity, $D_{\rm f}$ is the focal depth, and z is the vertical distance from the focal plane.

Using the simplified visibility function I(z) we can model the distorted velocity field $v_{\rm conv}(z)$ as a convolution integral of the simple theoretical velocity field $v_{\rm NS}(z)$ with the visibility function:

$$v_{\rm conv}(z) = \frac{\int_{-h/2}^{h/2} v_{\rm NS}(z') I(z'-z) dz'}{\int_{-h/2}^{h/2} I(z'-z) dz'},$$
 (5)

where h is the height of the channel.

ι

A theoretical estimate of the error bars $\sigma_v(z)$ can be obtained as being inversely proportional to the square root of the amount of statistical data,

$$\sigma_v(z) \propto \left[\int_{-h/2}^{h/2} I(z'-z) \, dz' \right]^{-1/2}.$$
 (6)

It is important to notice that we cannot expect total agreement between $v_{\rm conv}(z)$ and the measured velocity $v_{\rm PIV}(z)$. $v_{\rm conv}(z)$ is the weighted average of a theoretical profile, where the weighting function I(z) is given by Eq. 4, and no base-clipping is considered. $v_{\rm PIV}$ is based on measured data after base-clipping, which reduces the signal from out-of-focus particles. Despite the base-clipping there still is some influence from the finite optical resolution, and we estimate it qualitatively by means of a convolution.

7 Results and discussion

As a first result, Fig. 5(a) contains a comparison of two experimental velocity profiles $v_{\rm PIV}(z)$ with the simple theoretical profile $v_{\rm NS}(z)$ for the suspension of beads in water. The zero values of the theoretical profile indicate the channel boundaries. There is a fine agreement at the center, but some discrepancy when the boundaries are approached. Beyond the boundaries, an out-of-focus signal is picked up, a feature which is not captured at all by the simple theoretical profile. However, the effect is real: it is nicely reproduced in the two different experiments.

Using instead the convolved theoretical profile $v_{\rm conv}(z)$ from Eq. (5) we obtain Fig. 5(b), where it is seen that the convolution with the visibility function yields a qualitatively correct description of the non-zero values at the channel boundaries. It furthermore explains both the increase in the observed velocities outside the channel, and the rapid increase in uncertainty.

Our measurements on the beads demonstrate that it is possible to resolve velocities normal to the focal plane with a focal depth of 4 μ m using volume illuminated μ PIV, averaged cross-correlation and base-clipping.

We have seen that moving the focal plane outside the channel leads to an increase in observed velocities and to a steep increase in the associated uncertainties. This can be explained qualitatively as follows. When the focal plane and hence the maximum of the visibility function lies outside the channel, the observed average velocity increases because the high-velocity particles in the center of the channel are seen with the same (low) intensity as the low-velocity particles at the boundaries. Moreover, only very few particles are observed at all thus resulting in large $1/\sqrt{N}$ -fluctuations and increased uncertainties.

For the blood measurements, the velocity profile is seen in Fig. 6(a) with the corresponding uncertainties. The uncertainty of a velocity measurement depends on the density of the moving refraction patterns (the image density) in the interrogation volume. For positions of the focal plane inside the blood-filled channel, the particle seeding is relatively high, and the uncertainties become correspondingly small. Even outside the channel the uncertainties are surprisingly small, but as in the case of the bead suspension they increase significantly as the focal plane is moved outside the channel. The first channel wall was positioned where the steepest increase in uncertainty was observed (bottom), and the second wall (top) was determined by the channel height.

Blood is known to develop cell free layers next to solid boundaries to lubricate the transport of a semi-solid plug consisting of cells[8], i.e., a two-phase system. The nearly flat velocity profile in Fig. 6(a) strongly suggests that such a two-phase model indeed is a good description of the system, whereas a single-phase model employing the viscosity data from Fig. 1 does not fit the data. A twophase model is seen in Fig. 6(b), where the velocity of the plug is 17 mm s⁻¹, and the cell free layer is a Newtonian liquid approximated with the viscosity of water. We can obtain an estimate of the width d of the cell-free layer by matching the nominal flow rate Q with the model. We find

$$d = 2\left(A - \frac{Q}{v_0}\right) \frac{1}{l},\tag{7}$$

where $A = 1.09 \times 10^{-8} \text{m}^2$ is the area of the channel, $v_0 = 17 \times 10^{-3} \text{ m s}^{-1}$ is the plug velocity, and $l = 7.57 \times 10^{-4}$ m is the perimeter of the channel. The cell free layer was calculated to be $d = 3 \mu \text{m}$, which is 21% of the channel. In the literature it is reported that cell free layers in microtubes for diluted blood samples took up to about 10% of the volume[8]. Considering the increase in uncertainties and the fine agreement with the twophase model, it is reasonable to assume that we have been able to resolve the velocities along the z-axis, and that we have observed a plug flow.

We have successfully measured steady state velocity profiles on suspensions of beads and human blood in a microchannel. The microflows were imaged through an optical system using stroboscopic back illumination. Due to the focal depth, the optical system limits the resolution vertical to the focal plane, which results in an averaging of velocities in adjacent planes. In the bead suspension, we have qualitatively estimated the influence of the focal depth by means of a Lorentzian intensity function Eq. (4). Using this intensity function, we show that the optical system has a focal depth of 4 μ m. Furthermore, the intensity function illustrates why the measured velocity seems to increase, when the focal plane is moved outside the fluidic channel. Additionally, it provides a prediction of the trend in uncertainties, which can be a help in determining the position of walls parallel to the focal plane.

In the blood suspension, we measured a velocity profile, which together with a two-phase model gives a strong Micro particle-image velocimetry of bead suspensions and blood flows

indication of a plug flow. The fitting of a two-phase model resulted in a cell free layer of 3 $\mu m.$

References

- Sanders GHW; Manz A (2000) Chip-based microsystems for genomic and proteomic analysis. Trends Anal. Chem 19, 364-378
- 2. Trebotich D; Chang W; Liepmann D (2001) Modeling of blood flow in simple microchannels. Proc. Modeling and Simulation of Microsystems 2001, 218-222
- 3. Sugii Y; Nishio S; Okamoto K (2002) In vivo PIV measurements of red blood cell velocity field in microvessels considering mesentery motion. Physiol. Meas. 23: 403-416
- Koutsiaris AG; Mathioulakis KS; Tsangaris S (1999) Microsope PIV for velocity-field measurement of particle suspensions flowing inside glass capillaries. Meas. Sci. Technol. 10: 1037-1046
- Meinhart CD; Wereley ST; Santiago JG (1999) PIV measurements of a microchannel flow. Exp. Fluids 27: 414-419
- Olsen MG; Adrian RJ (2000) Out-of-focus effects on particle image visibility and correlation in microsopic particle image velocimetry. Exp. Fluids [Suppl.], S166-s174
- Rigshospitalet, H:S Blodbank (2003) Blegdamsvej 9, DK-2100 Copenhagen, Denmark
- 8. Fung YC, Biomechanics (2nd ed., Springer-Verlag 1993)
- FlowManager. For the PIV data-analysis we used the commercial software package FlowManager version 3.70.06, which is developed by Dantec Dynamics A/S

Bibliography

- [1] Lennart Bitsch, Laurits H. Olesen, Carsten H. Westergaard, Henrik Bruus, Henning Klank, and Jörg P. Kutter, Transmission micro PIV resolving velocities normal to the focal plane Accepted for microTAS 2003, Lake Taho, USA, October 2003
- [2] Lennart Bitsch, Laurits H. Olesen, Carsten H. Westergaard, Henrik Bruus, Henning Klank, and Jörg P. Kutter, Micro PIV on blood flow in a microchannel, Accepted for microTAS 2003, Lake Taho, USA, October 2003
- [3] Lennart Bitsch, Laurits H. Olesen, Carsten H. Westergaard, Henrik Bruus, Henning Klank, and Jörg P. Kutter, Micro particle-image velocimetry of bead suspensions and blood flows *Exp. Fluids*, (submitted 10 July 2003)
- [4] Flemming Rytter Hansen, Dispersion in electrokinetically and pressure driven microflows, Master thesis, Mikroelektronik Centret, DTU (2002)
- [5] Arieh Iserles, A first course in the numerical analysis of differential equations, Cambridge University Press (1996)
- [6] Susanne C. Brenner, L. Ridgway Scott, The mathematical theory of finite element methods, Springer-Verlag New York, Inc. (1994)
- [7] C. A. Brebbia, M. H. Aliabadi, [editors], Adaptive finite and boundary element methods, Computational Mechanics Publications Southampton Boston (1993) [Elsevier Science Publishers Ltd London New York]
- [8] A. H. Stroud, Don Secrest, Gaussian quadrature formulas, Prentice Hall (1966)
- [9] D. A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, *Int. J. Numer. Meth. Eng.* **21**, 1129-1148 (1985)
- [10] MATLAB www.mathworks.com

- [11] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical recipes in C, the art of scientific computing, second edition, Cambridge University Press (1992)
- [12] Ramon Codina, Jordi Blasco, A finite element formulation for Stokes problem allowing equal velocity-pressure interpolation, *Comput. Meth*ods Appl. Mech. Engrg. 143, 373-391 (1997)
- [13] O. C. Zienkiewicz, P. Nithirasu, R. Codina, M. Vazquez, P. Ortiz, The Characteristic-Based Split procedure: an efficient and accurate algorithm for fluid problems, *Int. J. Numer. Meht. Fluids* **31**, 359-392 (1999)
- [14] L. Franca, T. J. R. Hughes, A. F. D. Loula, and I. Miranda, A new family of stable elements for nearly incompressible elasticity based on a mixed Petrov-Galerkin finite element formulation, *Numerische Mathematik* 53, 123-141 (1988)
- [15] Mark A. Christon, Dealing with pressure: FEM solution strategies for the pressure in the time-dependent Navier-Stokes equations, Int. J. Numer. Meth. Fluids 38, 1177-1198 (2002)
- [16] O. C. Zienkiewicz, R. L. Taylor, The finite element method, fifth edition, volume 3: fluid dynamics, Butterworth-Heinemann (2000)
- [17] R. Codina, J. Blasco, G. C. Buscaglia, A. Huerta, Implementation of a finite element formulation for the incompressible Navier-Stokes equations based on a pressure gradient projection, *Int. J. Numer. Meth. Fluids* 37, 419-444 (2001)
- [18] I. G. Kovasznay, Laminar flow behind a two-dimensional grid, Proceedings of the Cambridge Philosophical Society 44, 58-63 (1948)
- [19] B. F. Armaly, F. Durst, J. C. F. Pereira, B. Schönung, Experimental and theoretical investigation of backwars-facing step flow, J. Fluid Mech. 127, 473-496 (1983)
- [20] Lennart Bitsch, Blood flow in microchannels, Master thesis, Mikroelektronik Centret, DTU (2002)
- [21] P. Szabo, J. M. Rallison, E. J. Hinch, Start-up flow of a FENE-fluid through a 4:1:4 constriction in a tube, J. Non-Newtonian Fluid Mech. 72, 73-86 (1997)
- [22] Y. C. Fung, Biomechanics, mechanical properties of living tissues, Springer Verlag (1984)
- [23] Anders Brask, Principles of electroosmotic pumps, Master thesis, Mikroelektronik Centret, DTU (2002)

- [24] Anders Brask, Henrik Bruus, and Jörg P. Kutter, Electroosmotic pump with ion exchange membranes and a current feedback for flow control, Accepted for microTAS 2003, Lake Taho, USA, October 2003
- [25] R. F. Probstein, Physicochemical hydrodynamics, an introduction, second edition, John Wiley and Sons, Massachusetts Institute of Technology (1994)
- [26] Benny Lautrup, Physics of continuous matter; exotic and everyday phenomena in the macroscopic world, draft 7.2, Niels Bohr Institute (2001)