LINEAR STABILITY OF A BERMAN FLOW IN A CHANNEL 
PARTIALLY FILLED WITH A POROUS MEDIUM

by

CHUNTAO DENG

B.Eng., South China University of Technology, 1998 
M.Sc., Asian Institute of Technology, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF 
THE REQUIREMENTS FOR THE DEGREE OF 
DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES 
DEPARTMENT OF CHEMICAL AND BIOLOGICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

November 2004

© Chuntao Deng 2004
ABSTRACT

In this thesis, the coupled flow of a Newtonian fluid both above and through a porous medium is considered. This work was motivated from a pulp and paper application, namely twin-wire forming. In the fluid-only region, the two-dimensional flow field is governed by the Navier-Stokes equation. In the porous region, the flow field is governed by the Brinkman-extended Darcy law relationship. Inertial terms are retained in both regions and the interface conditions between the two domains are those as outlined by Ochoa-Tapia and Whitaker (Int. J. Heat Mass Transfer 38, 2635 1995).

The model equations were solved using two independent methods. In the first method we develop a similarity transform and reduce the governing equations to two, coupled, non-linear ordinary differential equations to form a three-point boundary value problem. This was solved numerically and validated analytically by examining asymptotic cases. Three characteristic solutions were identified and the stability of each was examined by the method of normal modes. In the second numerical approach, the governing equations were re-posed as a one-domain problem, using the procedure outlined by Basu and Khalili (Phys. Fluids 11, 1031 1999), so that the conditions at the interface need not be considered. The resulting equation was solved directly, in primitive variable form, using a finite volume formulation.

Finally, an experimental device was constructed to compare to the numerical predictions. Eight test cases were performed, using two different porous media, in which the velocity profile of the fluid was measured using pulsed ultrasound doppler anemometry (PUDA). Good agreement was found between the similarity numerical predications and the experimental measurements.
# Contents

ABSTRACT ii

Table of Contents iii

List of Figures vi

List of Tables xi

Nomenclature xii

Acknowledgment xvii

1 INTRODUCTION 1

2 LITERATURE REVIEW 6
   2.1 Rheology of Fibre Suspensions ................................. 6
   2.2 Papermaking .................................................. 13
   2.3 Objectives of the Research ................................. 15

3 MODEL EQUATIONS 17
   3.1 Introduction ............................................... 17
   3.2 Model Equations ......................................... 21

4 SIMILARITY SOLUTIONS 23
   4.1 Similarity Formulation .................................. 24
   4.2 Steady-State Solutions ................................. 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1</td>
<td>Asymptotic Solutions</td>
<td>26</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Series Solutions</td>
<td>30</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Numerical Solutions</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Stability of the Similarity Solutions</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>ONE DOMAIN PROBLEM</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>One-Domain Formulation</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>EXPERIMENTAL WORK</td>
<td>54</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>54</td>
</tr>
<tr>
<td>6.2</td>
<td>Experimental Apparatus</td>
<td>54</td>
</tr>
<tr>
<td>6.3</td>
<td>Pulsed Ultrasound Doppler Anemometry</td>
<td>58</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Measuring Principle</td>
<td>58</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Adjustment of Parameters</td>
<td>59</td>
</tr>
<tr>
<td>6.4</td>
<td>Experimental Procedure</td>
<td>62</td>
</tr>
<tr>
<td>6.5</td>
<td>Data Evaluation</td>
<td>63</td>
</tr>
<tr>
<td>6.6</td>
<td>Results and Discussion</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>SUMMARY AND CONCLUSIONS</td>
<td>76</td>
</tr>
<tr>
<td>8</td>
<td>RECOMMENDATIONS</td>
<td>78</td>
</tr>
<tr>
<td>A</td>
<td>DERIVATION OF THE PRESSURE BOUNDARY CONDITION</td>
<td>91</td>
</tr>
<tr>
<td>A.1</td>
<td>Normal Stress</td>
<td>91</td>
</tr>
<tr>
<td>A.1.1</td>
<td>Fluid Region</td>
<td>92</td>
</tr>
<tr>
<td>A.1.2</td>
<td>Porous Region</td>
<td>93</td>
</tr>
<tr>
<td>A.2</td>
<td>Pressure Drop</td>
<td>93</td>
</tr>
<tr>
<td>B</td>
<td>THE ASYMPTOTIC SOLUTIONS</td>
<td>95</td>
</tr>
</tbody>
</table>

iv
# List of Figures

1.1 The upper figure is a sketch of the first industrial paper machine, the Fourdrinier former, and the lower figure is a schematic drawing of a modern twin-wire hybrid former [1]. .................................................. 3

1.2 A twin-wire former of a pilot paper machine (Paprican) .................. 4

1.3 The channel flow considered. .................................................. 4

2.1 Yield stress versus crowding number from the data given by Bennington et al. [2]. The line shown is a regression using Eq. (2.6). Two different pulp types are shown: the SBK pulp is shown as a circle (o) and the TMP as a square (□). Bennington et al. report experimental uncertainties as large as 100%. .................................................. 10

2.2 A schematic of the geometry in twin-wire forming [1]. .................... 13

2.3 A schematic of the growth of the web during one-sided dewatering [1] .... 13

3.1 The channel flow considered. .................................................. 18

4.1 (i): The x-component of the velocity profiles \( f_y \) and \( g_y \) (normalized by the averaged velocity \( \bar{V} - x \) at position \( x \)) estimated in the limit of \( Re \to 0 \). Four curves are shown: (a) \( Da/\varepsilon \to 0 \), (b) \( Da/\varepsilon = 5 \times 10^{-4} \), (c) \( Da/\varepsilon = 1 \times 10^{-2} \), and (d) \( Da/\varepsilon \to \infty \). (ii): An estimate of the boundary layer thickness in the porous medium. The boundary layer thickness has been normalized to the thickness of the porous medium. These estimates were made with \( \beta = 0.2 \), \( \chi = 0.8 \) and \( \varepsilon = 0.9 \). .................................................. 28
4.2 The effect of \( Re \) for the case \( Da = 1 \times 10^{-3} \) on the the \( x \) component velocity profiles, \( f_y \) and \( g_y \), here \( \beta = 0.2, \chi = 0.8, \epsilon = 0.9. \)

4.3 The 6th order series solution (-) compared to the similarity numerical solution (\( \cdot \)) with \( Da = 1 \times 10^{-3}, Re = 5, \beta = 0.2, \chi = 0.9. \)

4.4 Example solutions: The plot represents the three different families of solutions obtained, here (a): \( Re = 10.93 \), (b): \( Re = 16.68 \) and (c): \( Re = 32.21 \). These simulations were conducted at \( Da = 0.05, \beta = 0.2, \chi = 0.8 \) and \( \epsilon = 0.9 \). The shaded area represents the porous region.

4.5 Example solutions: The figure represents the shear stress at the interface \( -f_{yy}(\chi) \) (here the circles indicate the series solutions given in §4.2.2). These simulations were conducted at \( Da = 0.05, \beta = 0.2, \chi = 0.8 \) and \( \epsilon = 0.9 \). The type of solution is indicated, where the line "|" shows the transitions from type-I to type-II and from type-II to type-III solutions. Here the solid-lines indicate stable solutions while the dotted region represents unstable solutions.

4.6 A 3-dimensional plot of the solutions at \( Da \) and \( Re \) considered, here \( \beta = 0.2, \chi = 0.8, \epsilon = 0.9. \)

4.7 The stability analysis for \( Da = 0.013 \): (i) is the steady solution, the detailed view in Region I is shown in (ii), whilst (iii) shows the real part of the corresponding eigenvalues in Region I. Here the dashed lines indicate unstable solutions.

4.8 (a) Summary of solution behavior for simulations with \( \beta = 0.2, \chi = 0.8 \) and \( \epsilon = 0.9 \). Nine different solution regions are found and the detailed information about each region is tabulated in Table 4.1. (b) Region C is enlarged.

4.9 Example solutions: the shear stress function \( -f_{yy}(\chi) \) versus \( Re \) for (a) \( Da = 0.013 \), (b) \( Da = 0.026 \) and (c) \( Da = 0.0375 \). The type of the solutions are denoted, and the multiple solution regions are labelled A, B, etc. representing different type, stability and number of solutions. Here solid-line denotes stable solution and dotted-line, the unstable solution.
5.1 The flow fields as estimated by the two different numerical approaches, here 
\( \Delta P_x \) is evaluated at the interface \( y = 0.8 \) and \( \Delta P_y \) is evaluated at \( x = 7.0 \). The porous medium extends from \( y = 0.8 \) to \( y = 1 \), with \( Da = 1 \times 10^{-3} \), 
\( Re = 5 \) and \( \varepsilon = 0.9 \). In the lower panel of the figure, the fully-developed 
profile using the one-domain approach at \( x = 7.0 \) (shown as a dashed line) 
is compared to the similarity solution (shown as a solid line) with \( \beta = 0.2 \).

5.2 The flow field determined as a function of \( \beta \) using the similarity approach. 
The normalized \( x \)-component of velocity (\( f_y \) for fluid region and \( g_y \) for porous 
region) is given in the figure on the left and the \( y \)-component (\( f \) for fluid 
region and \( g \) for porous region) is given on the right. The porous medium 
(shown as the shaded area) extends from \( y = 0.8 \) to \( y = 1 \). This simulation 
was performed with \( Da = 1 \times 10^{-3} \), \( Re = 5 \), and \( \varepsilon = 0.9 \).

5.3 The factors affecting the empirical constant \( \beta \) in the shear stress jump condi­
tion (the tested cases are for \( \chi = 0.9, \varepsilon = 0.9 \)).

6.1 The schematic of the experimental setup.

6.2 The assembled test section in (a) exploded view and (b) assembled view.

6.3 The working mechanism of pulsed ultrasound Doppler anemometry (image 
taken from Signal-Processing SA).

6.4 The interface of the pulsed ultrasound Doppler anemometer Dop2000 from 
Signal Processing SA.

6.5 The velocity measured off the probe is the component velocity of particles 
moving in the ultrasound beam direction. Here \( v \) denotes the flow velocity 
and \( v_p \) denotes the velocity of flow in the ultrasound beam direction. It 
shall be noted that \( \theta \) denotes the Doppler angle, where the refraction of the 
sound beam due to the sound speed difference in plexiglass and fluid has 
been accounted for.

6.6 The schematic of the projections of the \( x \) component of the numerical ve­
locity \( v_x \), \( y \) component \( v_y \) on the ultrasound beam direction, the PUDA 
velocity \( v_p \).
6.7 The similarity numerical solutions shown as solid line (here $\beta = 0.13$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 1 in Table 6.2.

6.8 The similarity numerical solutions shown as solid line (here $\beta = 0.3$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 2 in Table 6.2.

6.9 The similarity numerical solutions shown as solid line (here $\beta = 0.9$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 3 in Table 6.2.

6.10 The similarity numerical solutions shown as solid line (here $\beta = 1.5$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 4 in Table 6.2.

6.11 The similarity numerical solutions shown as solid line (here $\beta = 0.15$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 5 in Table 6.2.

6.12 The similarity numerical solutions shown as solid line (here $\beta = 0.58$) for
the velocity profile in the direction of the ultrasound beam compared with
the experimental results from PUDA (dotted line). The error bars denote
one standard deviation from the mean velocity values. The simulation is for
Experiment No. 6 in Table 6.2.
6.13 The similarity numerical solutions shown as solid line (here $\beta = 1.1$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 7 in Table 6.2.

6.14 The similarity numerical solutions shown as solid line (here $\beta = 1.1$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for the two steady state solutions for Experiment No. 8 in Table 6.2.

B.1 The shear stress function $-f_{yy}(\chi)$ at the interface for different $Re$, it compares the similarity numerical solution ($\circ$) with the asymptotic solution ($\cdot$). The simulation is for the case $Da = 1 \times 10^{-3}$, $\chi = 0.8$, $\beta = 0.2$ and $\epsilon = 0.9$.

C.1 The shear stress function $-g_{yy}(1)$ at the porous wall for different $Re$, it compares the similarity numerical solution ($\cdot$) for $Da = 1000$, $\chi = 0.8$, $\beta = 0$ and $\epsilon = 1$ with Berman solution ($\cdot$) [3, 4].

F.1 the 3-D view of the cover of the upstream-tank
F.2 the dimensions of cover of the upstream-tank
F.3 the 3-D view of the lid of the test channel
F.4 the dimensions of the lid of the test channel
F.5 the 3D view of the screen plate (filter) for the suction
F.6 the dimensions of the screen plate for uniform suction
F.7 the 3-D view of the suction element (diffusor) of the device
F.8 the dimensions of the suction element (diffusor)
F.9 the 3D view of the upstream and downstream tanks connected with a channel
F.10 the dimensions of the upstream and downstream tanks connected with a channel
List of Tables

4.1 The number, type and stability of the solutions for each solution region as shown in Fig. 4.8. ................................................. 43

6.1 The porous fabrics used in the experiment ............................................. 55

6.2 The experimental conditions tested. .................................................... 65
Nomenclature

\( a \)  
yield stress constant

\( b \)  
yield stress constant

\( B \)  
binary constant to turn on/off the sink term

\( c \)  
velocity of sound in a liquid

\( C_f \)  
form drag constant

\( C_m \)  
mass concentration of wood pulp suspensions

\( C_s \)  
sediment concentration

\( d(\xi) \)  
the interface between the paper web and the suspension

\( D \)  
nominal diameter of the fibre

\( D(\phi) \)  
a function of the permeability and compressibility of the fiber network

\( D_a \)  
Darcy number, \( D_a = K/H^2 \)

\( f_e \)  
the emitting frequency of the ultrasound pulse

\( f_d \)  
the frequency shift of the received echo from the particles in PUDA measurement

\( f(y) \)  
similarity velocity function in fluid region

\( F(y) \)  
the perturbed similarity function for the fluid region

\( g(y) \)  
similarity velocity function in porous region

\( g(\xi) \)  
fabric position

\( \bar{g} \)  
acceleration due to gravity

\( G(y) \)  
the perturbed similarity function for steady case problem in the porous region

\( h \)  
height for fluid region

\( H \)  
total height for the channel
\( J(\xi) \) the dewatering rate (flux)
\( k(\phi) \) a function to describe the fiber-fiber contact stress
\( K \) permeability of the porous medium
\( l \) fiber length
\( L_f \) interaction length for crowding number
\( m \) empirical constant of the power-law function
\( \bar{M} \) inter-phase momentum term (vector)
\( n \) empirical constant of the power-law function
\( n_c \) number of contacts between fibers
\( N \) crowding number
\( N_{gel} \) gel crowding number
\( \bar{p} \) unit vector that indicates fiber orientation
\( p(l_i) \) fiber length distribution
\( P \) pressure
\( P_+ \) pressure at position just above the interface
\( P_- \) pressure at position just below the interface
\( P_d \) the depth of the particle measured by ultrasound
\( P_{max} \) the maximum depth of the particle that measurable by PUDA
\( PRF \) pulse repetition frequency
\( q \) rescaled position of porous wall
\( r_d \) aspect ratio of the fiber
\( r_i \) the radius of curvature
\( Re \) suction/injection Reynolds number, \( Re = \frac{U_w H}{\nu} \)
\( S_o \) specific surface of the fiber defined as the ratio of the surface area to its volume
\( t \) time
\( t_d \) the time delay between an emitted burst and the echo issued from the particle
\( t_{pr,f} \) the time between two emissions
Fabric tension

Velocity vector in porous region

Velocity component in porous region

x velocity component in porous region

y velocity component in porous region

Fabric moving velocity

Uniform suction velocity at porous wall

Mean axial velocity at x = 0

Settling velocity of fiber

Velocity vector in fluid region

Velocity component in fluid region

Velocity component in the ultrasound beam direction

x velocity component in fluid region

y velocity component in fluid region

Maximum velocity that can be measured using PUDA

Velocity vector for one-domain approach

Water retention ratio

X Derivative in x direction

Y Derivative in y direction

Subscripts

Superscripts

* The corresponding dimensional properties, SI unit.

The time dependent functions

Greek letters
\( \beta \) shear stress jump coefficient
\( \beta_2 \) the quadratic shear stress jump coefficient
\( \dot{\beta} \) the artificial compressibility constant
\( \chi \) the dimensionless interface position between the fluid and porous region
\( \delta \) phase shift of the received echo for PUDA
\( \delta(\xi) \) web thickness
\( \Delta \rho \) density difference between fiber and surrounding fluid
\( \epsilon \) porosity of the porous media
\( \phi \) solidity or volume concentration of the fibers
\( \phi_0 \) Gel concentration
\( \eta \) the \( \eta \) direction in a nonorthogonal coordinate
\( \lambda \) rescaled similarity velocity function for fluid region
\( \mu \) dynamic viscosity of fluid
\( \mu_f \) apparent viscosity of the pulp suspension
\( \mu_e \) effective viscosity
\( \nu \) kinematic viscosity
\( \Theta(\phi) \) function in Forchheimer model to describe \( \tilde{M} \)
\( \rho \) fluid density
\( \sigma \) the eigenvalues of the linear stability equations
\( \bar{\tau} \) fiber-fiber contact stress tensor
\( \tau \) normal stress
\( \tau_0 \) yield stress of pulp suspension
\( \omega \) fiber coarseness
\( \Omega_1 \) domain for flowing suspension region
\( \Omega_2 \) domain for paper web region
\( \xi \) the \( \xi \) direction in a nonorthogonal coordinate
\( \Xi(\phi) \) function in Forchheimer model to describe \( \tilde{M} \)
\( \psi \) rescaled similarity velocity function for porous region

\( \Psi \) stream function

\( \zeta \) intermediate constant, equal to \( \sqrt{\epsilon/Da} \)

\( \theta \) a dimensionless constant for slip-velocity boundary condition at the interface
ACKNOWLEDGEMENT

I wish to thank those who have offered their help and support to this work. In particular, I sincerely thank my supervisor Mark Martinez for his guidance, encouragement and help. My sincere thanks also goes to Dr. Bruce Bowen, Dr. James Olson, Dr. Richard Branion and Dr. Richard Kerekes, for their time and suggestions in this work. I would like to thank Peter Taylor, Doug Yuen, Tim Paterson and Alex Thng for helping the experiment setup; Ken Wong, Brenda Dutka and Lisa Hudson for their technical and administrative assistance.

My friends Edmond, Jacky, Leila, Monica, Robin, Satya, Yaoguo, Yubing, and Zhaolin, deserve my great thanks for being so supportive. At last, this work could not be accomplished without the consistent support and love from Lili, my wife; and encouragement from my brother, sister and their families, and my parents.

The financial support from Natural Sciences and Engineering Research Council of Canada (NSERC), University of BC Graduate Fellowship (UGF) and Advanced Papermaking Initiative (API) are acknowledged.
Chapter 1

INTRODUCTION

The earliest recording of paper manufacture has been credited to the Chinese around 100 A.D., where a suspension of silk and mulberry was filtered in a bamboo mold. The first industrial paper machine was developed around 1800. Since this time, paper machines have been continually under development leading to the modern machines called twin-wire formers (see Fig. 1.1); the reader is referred to Norman's review for further details [5]. Here the term “wire” (or more recently termed “fabric”) refers to a woven, usually plastic, permeable membrane through which the papermaking suspension is drained.

During processing, the papermaking suspension is initially fluidized by turbulence created locally from a sudden change in geometry in an apparatus called a “headbox”. In this case, the fibre network is broken down into smaller flocs and single fibres with weakly correlated velocities [6]. Suspension fluidization and the associated breakage of flocs is attained by inducing turbulence [7, 8, 9] and is aided by the addition of chemical deflocculants [10, 11, 12, 13, 14, 15, 16]. The fluidized fibre suspension is then formed into a plane liquid jet and distributed in the gap formed between the two moving wires (see Fig. 1.2). The velocity of the jet is between 10 – 35 m/s and it is 5 – 10 m wide and 1 cm thick. Often, however, this fluidized state is transient and the network re-flocculates within a few milliseconds upon the reduction of the applied shear [6, 17].

With non-traditional forming processes, namely high consistency (HC) forming, previous workers have attempted to freeze the structure of the sheet before it leaves the headbox. In the design put forth by Grundström et al. [18], the pulp suspension is first dispersed by turbulence, generated by numerous bends in the flow channel, and then allowed to be re-flocculated in a small straight outlet channel before impinging
During roll-blade forming, water is filtered from the suspension in two distinct steps. In the first step, water is forced through the wire by the momentum of the jet and by the tension and convergence of the fabrics passing over the roll. The fabrics are then passed over a series of blades, in the second step, where greater water removal is achieved. In both these steps the change in curvature of the fabrics generates a pressure field to balance the fabrics’ tension. Early attempts to characterize the action of twin-wire formers were largely concerned with estimating this pressure. Hergert and Sandford [22] measured the pressure distribution on the surface of a roll during one-sided dewatering and measured gauge pressures as high as 5 kPa. Zhao and Kerekes [23] reported gauge pressures as high as 10 kPa in blade formers.

In the late sixties and early seventies both Baines [24] and Meyer [25] made rigorous attempts to model the twin-wire dewatering process in order to better understand the changes in sheet structure. Although most of their assumptions were reasonable, they did not provide any solution to the resulting equations due to the complexity of the problem. In subsequent work, a number of authors have simplified these analyses by considering the suspension as a Newtonian fluid and neglecting the variation of properties in the drainage direction [23, 26, 27, 28, 29]. These models have engineering value but give no insight into the development of paper properties.

A full, theoretical analysis of the forming process would be extremely challenging. Furthermore, validation of such a model would be difficult, if not impossible. An alternative approach is to focus only on one aspect of the process, and neglect the other factors. In our case, the aspect addressed in this thesis is the interface between the suspension and the paper web. The geometry considered in this work is shown in Fig 1.3.

The relevant literature will be reviewed in Chapter 2 of this thesis. At the end of this chapter the current state of knowledge is summarized and the objectives of the research on a moving wire. A similar concept was advanced by Nomura and his co-workers [19, 20, 21] but in this case turbulence was generated by a sudden change in geometry achieved in a saw-tooth shaped channel. Both of these concepts are similar in that turbulence intensity is strongly coupled to the mean velocity of the suspension. Hence floc dispersion is achieved through an increase in jet velocity. Because of this distinctive forming method the structure of the HC paper differs from that of conventional paper.
Figure 1.1: The upper figure is a sketch of the first industrial paper machine, the Fourdrinier former, and the lower figure is a schematic drawing of a modern twin-wire hybrid former [1].
CHAPTER 1. INTRODUCTION

---

Figure 1.2: A twin-wire former of a pilot paper machine (Paprican)

---

Figure 1.3: The channel flow considered.
work are stated. The model equations are presented in Chapter 3. In Chapter 4 the model equations are transformed to a three-point boundary value problem and solved using a similarity ansatz. The stability of the solutions is also considered in this chapter. In Chapter 5, the full set of equations is solved using a standard computational fluid dynamic (CFD) approach and validated experimentally in Chapter 6. In Chapter 7 we summarize the major findings in this work and the recommendations are given in Chapter 8. The appendices at the end provide additional information about the CFD algorithms, the numerical codes and the experimental apparatus.
Chapter 2

LITERATURE REVIEW

2.1 Rheology of Fibre Suspensions

Papermaking fibres are hollow, flexible, rod-like particles that have a wide distribution in both length and diameter depending upon species and growing conditions; when liberated from trees, they can curl or kink. North American fibres contain typically 40-50% cellulose, 20 to 35% hemicellulose and 15 to 35% lignin, with the remaining fraction consisting of resins, tannins, ash and miscellaneous compounds [30]. Papermaking fibre suspensions are a mixture of wood fibres, water, clay, as well as small quantities of inorganic salts and polymeric additives. This physical make-up leads to complex rheological behavior with the suspensions showing complex stress/strain-rate relationships. The major feature of this relationship is that it conforms to that of an inelastic non-Newtonian fluid.

Under typical processing conditions, papermaking fibre suspensions aggregate into coherent networks which possess measurable mechanical strength [7, 8, 10, 11]. The reviews of Kerekes [31], Jokinen and Ebeling [12], and Powell et al. [32] and the references contained therein summarize our current knowledge of fibre suspension rheology. Although fibre networks form by surface charges, Mason [33] found that mechanical entanglement rather than colloidal forces was the principle source of fibre flocculation. Fibres collide because of the relative motion induced by the flow field. They entangle, bend, and remain networked from frictional forces transmitted by fibres that are locked into bent configurations [34]. Soszynski and Kerekes [35] confirmed the influence of bending stresses on
floc strength through stress relaxation experiments. They defined the propensity for papermaking fibres to flocculate in terms of a dimensionless number, termed the crowding number, \( N \), i.e.,

\[
N = \frac{2}{3} \phi \left( \frac{L_f}{D} \right)^2 \approx \frac{C_m L_f^2}{2\omega}
\]  

where \( C_m \) is the concentration of the solid material in the suspension (kg/m\(^3\)); \( L_f \) is an interaction length based upon the length of a fibre (l); \( \omega \) is the fibre coarseness defined as the mass per unit length of fibre; \( D \) is the diameter of the fibre; and \( \phi \) is the solidity or volume concentration of the fibres.\(^1\)

It should be noted that, for polydisperse suspensions, Kerekes and Schell \[37\] recommend the use of the first moment of the fibre length distribution \( p(l_i) \) to estimate the interaction length, i.e.,

\[
L_f = \sum p(l_i)l_i
\]  

This relationship however has yet to be justified. The physical significance of \( N \) for mono-disperse suspensions is given by its definition; it reflects the number of fibres in a spherical volume of diameter equal to \( L_f \). Flocculation does not generally occur when \( N < 1 \) as on average each fibre can rotate freely. The range of \( 1 < N < 60 \) represents a regime of fibre behavior described by Soszynski \[38\] as one of ‘forced collisions’ between fibres in simple shear flow. Finally, \( N \) can be related to the average number of contacts \( n_c \) between fibres in a network using \[39\]

\[
N \approx \frac{n_c}{3} \left( \frac{L_f}{D} \right)
\]  

Most papermaking operations in British Columbia occur in the range between \( 17 < N < 60 \) \[40\].

Understanding the motion of dilute \( (N \ll 1) \) fibre suspensions is difficult. Insight into this phenomenon can be gained by first examining the simplest case of the motion of a single isolated fibre settling under Stokes’ flow conditions, i.e. \[41\],

\[
\dot{V}_{sed} = \frac{\Delta \rho D^2}{16\mu} \left[ (\ln 2r_d + 0.193 + O(\ln 2r_d)^{-1})\vec{g} + (\ln 2r_d - 1.807 + O(\ln 2r_d)^{-1})(\vec{p} - \vec{g})\vec{p} \right], \quad (2.4)
\]

\(^1\)Soszynski \[36\] investigated the relationship between the mass and volume concentration of wood pulp suspensions, i.e. \( \phi \approx (\frac{2}{3} + w_k)C_m \), where \( w_k \) is the water retention value.
where $\Delta \rho$ is the difference in density between the fibre and the surrounding fluid; $r_d$ is the aspect ratio of the fibre defined by $l/D$; $\mu$ is the viscosity of the fluid; $g$ is the acceleration due to gravity; and $\vec{p}$ is a unit vector that indicates fibre orientation. Unlike spheres, fibres can have significant motion perpendicular to gravity with a drift velocity strongly dependent upon its orientation. Ross and Klingenberg [42] considered this case for papermaking fibres by dropping one fibre at a time into a quiescent fluid to create a sediment. Using techniques similar to those employed by Doi and Chen [43] in polymer dynamics, these authors modelled papermaking fibres as a chain of prolate spheroids connected through elastic ball and socket joints in order to account for flexibility effects.

With semi-dilute suspensions ($1 < N < 60$), this situation becomes more complex as each individual fibre moves under the influence of the long-range hydrodynamic disturbances of the other particles. As an example, during sedimentation, these disturbances lead to inhomogeneous settling rates [44] and local floe formation. With monodisperse suspensions, the latter has been observed by Kumar and Ramarao [45], and confirmed by Herzhaft et al. [46] (called clumps by these authors), who reported the existence of flocs for monodisperse glass fibres settling at low crowding numbers. Kumar and Ramarao also noted that, as the fibre concentration increased, the number of flocs increased and caused greater hindrance effects. With bidisperse suspensions, Davis [47] observed size segregation up to a critical concentration, above which particle interlocking prevented further segregation. It was found that the smaller the ratio of the large to small particle diameters, the smaller the critical concentration. Quite possibly due to these complexities, there are only a few mathematical models of the motion of interacting particles [41, 48, 49]. Koch and Shaqfeh [49] considered the sedimentation of stiff, axisymmetric, nonspherical particles in a dilute suspension and demonstrated that particle flocculation can occur. Moreover, from a stability analysis, they suggested that the sedimenting suspension should segregate into regions of high particle density and cause a relatively strong downward convection of particles.

With $N > 60$, many authors consider the network to behave like a compressible porous medium [50, 51]. This approximation stems from the fact that at this level of fibre crowding, there are, on average, 3-4 interfibre contacts [34, 52, 53, 39] and according to Steenberg et
al. [54], this is the minimum number of contacts required to create a network.

In this regime, it is widely known that pulp suspensions do not flow until a certain critical (or yield) shear stress is exceeded. Suspension yield stresses have been either measured directly using viscometers, or inferred from dynamic shear and tensile measurements [31]. For pulp suspensions, the yield stress \( \tau_0 \) has been typically correlated with the suspension mass concentration using equations of the form [55]

\[
\tau_0 = a(C_m - C_s)^b
\]  

(2.5)

where \( a \) and \( b \) depend on the test procedure employed, the pulp type, and the degree of pulp treatment (e.g. beating) prior to testing; and \( C_s \) is the sediment concentration. Values of the constant \( b \) have been reported in the literature to range between 1.25 to 3.02. Few, if any of these works report the yield stress in terms of the crowding number. As an example, we do so, for the data given in [2], see Fig. 2.1, and correlate this data set with an equation of the form

\[
\tau_0 = 5.5 \times 10^{-4}(N - N_{gel})^{2.3}, \quad R^2 = 0.96
\]  

(2.6)

where \( N_{gel} \) is the gel crowding number defined by Martinez et al. [56] to be \( N_{gel} = 16 \).

At this level of fibre crowding, the transport of mass and momentum can be expressed by the Eulerian volume-averaged two-phase flow equations [57, 58, 59]:

\[
\nabla \cdot (\phi \bar{u}^*) = 0
\]  

(2.7)

\[
\nabla \cdot [(1 - \phi)\bar{w}^*] = 0
\]  

(2.8)

\[
\phi \rho_f \bar{u}^* \cdot \nabla \bar{u}^* = -\phi \nabla P^* - \nabla \cdot \bar{f} + \rho_f \phi \bar{g} - \bar{M}
\]  

(2.9)

\[
(1 - \phi)\rho_w \bar{w}^* \cdot \nabla \bar{w}^* = -(1 - \phi) \nabla P^* + \rho_w (1 - \phi) \bar{g} + \nabla \cdot [\mu (1 - \phi) (\nabla \bar{w}^* + \nabla \bar{w}^{*T})]
+ \bar{M}
\]  

(2.10)

where \( \bar{u}^* \) and \( \bar{w}^* \) are the dimensional velocities (vector quantities, and * indicates a dimensional property) of the fibres and the water, respectively; \( P^* \) is the fluid pressure; \( \bar{M} \) is the inter-phase momentum term (vector quantity); \( \rho_f \) and \( \rho_w \) are the densities of the fibre and water, respectively; and \( \bar{f} \) represents the fibre-fibre contact stress tensor as the
Figure 2.1: Yield stress versus crowding number from the data given by Bennington et al. [2]. The line shown is a regression using Eq. (2.6). Two different pulp types are shown: the SBK pulp is shown as a circle (o) and the TMP as a square (□). Bennington et al. report experimental uncertainties as large as 100%. 
CHAPTER 2. LITERATURE REVIEW

fibre phase is conceptually envisioned as a continuum [60]. Surface tension effects between the phases have been neglected.

In the one-dimensional filtration studies of Ingmanson et al. [50] and Meyer [51], where the relative fluid velocity was about 1 m/s,² they described $\bar{M}$ with a Forchheimer-type equation. These expressions have been extended to two dimensions by Zahrai [1], i.e.,

$$\bar{M} = \mu \Theta(\phi) \frac{\phi^2}{(1-\phi)^2} S_o^2 (\bar{u} - \bar{w}^*) + \rho_w \Xi(\phi) \frac{\phi}{(1-\phi)^2} S_o^2 |\bar{u} - \bar{w}^*| (\bar{u}^* - \bar{w}^*)$$  \hspace{1cm} (2.11)

$$\Theta(\phi) = \begin{cases} 3.5(1-\phi)^3 \phi^{-1/2}[1 + 57\phi^3] & \phi \leq 0.3 \\ 5.60 & \phi > 0.3 \end{cases} \hspace{1cm} (2.12)$$

$$\Xi(\phi) = 0.1 \sqrt{\Theta(\phi)} \hspace{1cm} (2.13)$$

where the quantity $S_o$ is the specific surface of the fibre defined as its surface area to volume ratio. The coefficients given in the above expression are taken from [51], but the reader is referred to a number of references found in the literature [50, 51, 61, 62, 63, 64, 65, 66, 67].

Finally, the remaining term which needs to be defined is the particulate stress term, $\bar{f}$. There is evidence in the literature of two different modes of behavior for fibrous suspensions.

i) With regards to the first mode of behavior, a large body of literature is available which ascribes $\bar{f}$ as an implicit function of the local solidity of the network, i.e.,

$$\bar{f} = k(\phi) \bar{I}$$  \hspace{1cm} (2.14)

The number of authors have represented the functional relationship between $k$ and $\phi$ by power-law functions of the form [69, 70, 71]:

$$k(\phi) = m(\phi^n - \phi_g^n) \hspace{1cm} (2.14)$$

or

$$k(\phi) = m(\phi - \phi_g)^n \hspace{1cm} (2.15)$$

where $\phi_g$ is the "gel" point and is defined as the value below which $k(\phi)$ cannot be distinguished from zero. Here, $m$ and $n$ are empirical constants determined from static compressibility tests and are defined uniquely for each relationship given above. It should be noted that this formulation for $\bar{f}$ neglects the shear stress that comes about from network deformation. It thereby assumes isotropic material behavior.

²This represents a Reynolds number of about 35, based upon the diameter of a fibre.
These formulations neglect plastic deformation and time-dependent behavior that have been shown to be important.

ii) On the other hand, pulp suspensions fluidize under applied shear. Bennington and Kerekes [8] reported the apparent viscosity of a semi-bleached kraft suspension (SBK) can be approximately represented by a function of solidity according to

\[ \mu_f \propto \phi^{3.1} \]  

(2.16)

As a result, the particle stress term for the fluid-like model may be estimated as [1]

\[ \bar{\tau} = \mu_f \phi (\nabla \vec{u}^* + \nabla \vec{u}^* T) \]  

(2.17)

To help explain the differences between these expressions for \( \bar{\tau} \), it is helpful to reiterate the assumptions upon which the equations are based. Clearly, both imply that \( \bar{\tau} \) varies with solidity. However, the fluid-like description assumes that \( \bar{\tau} \) varies with the velocity gradient of the fibres. The fluid-like description assumes a complete dispersion of the fibre suspension; it is free of entanglements. This implies that fibre interaction occurs through the fluid phase. The power law model, on the other hand, tacitly assumes mechanical interaction between the fibres. In other words, the fibres in the suspension do not necessarily have to move at the same velocity as the fluid in the suspension; their motion can be hindered by entanglements.

In summary,

i) The mathematical description of the motion of dilute (\( N \ll 1 \)) and semi-dilute suspensions (1 < \( N < 60 \)) is limited. In essence, the description of the long-range hydrodynamic force, fibre collisions and entanglement have yet to be tackled from a theoretical standpoint.

ii) With concentrated suspensions (\( N > 60 \)) the equations of motion are not a closed set of equations. The success of a mathematical model in this case is dependent on the accuracy of the closure equations for inter-phase momentum transport and the fibre-fibre contact stress. In the present state of the literature, the currently
available correlations for these parameters assume isotropic material behavior and neglect plastic deformation and time-dependent behavior.

2.2 Papermaking

In general the models used to describe the dewatering process during papermaking are remarkably similar and are summarized below. They began by considering the passage of two permeable wires over a curved surface with radius $R$ as shown in Fig. 2.2. The fabrics move under tension $T$ and at a constant velocity $U$. A fibrous suspension flows between the two fabrics. It is assumed that the fabric and suspension move at nearly the same velocity far upstream. The formation of the paper web is shown schematically in Fig. 2.3. The
coordinate system is defined with the origin at the point where the lower fabric touches
the curved surface. The position of the fabric is denoted by $g(\xi)$; the interface between the
web and the suspension is given by $d(\xi)$. The web thickness is therefore $\delta(\xi) = g(\xi) - d(\xi)$.
In these modelling studies, the forming web is considered as a separate domain [25, 72].
For clarity, two domains, $\Omega_1$ and $\Omega_2$, are usually defined to denote the regions describing
the flowing suspension and the paper web respectively. In $\Omega_1$, the model equations, in
their most general form, are based upon the following assumptions:

i) The flow of the suspension is considered to be Newtonian, incompressible and irrotational;

ii) The wires are pre-stressed with a constant tension, are inertialess, inextensible, ide­
ally flexible and of negligible thickness.

The incompressibility condition and the assumption that the flow is irrotational can be
expressed by a Laplace equation for the stream function $\Psi$

$$\nabla^2 \Psi = 0 \quad (2.18)$$

The suspension velocity $\vec{u}^*$ is related to the stream function as:

$$\vec{u}^* = (\Psi_y, -\Psi_x) \quad (2.19)$$

where the indices $x$ and $y$ represent differentiation with respect to $x$ and $y$. Based upon a
force balance, the curvature of the fabric is related to the pressure drop across the fabric
($\Delta P^*$) and the applied tension $T$. From classical mechanics this is given by

$$\Delta P^* = T \frac{1}{r_i} \quad (2.20)$$

where $r_i$ is the local radius of curvature of the fabric. The pressure drop across the fabric
can be estimated from the Bernoulli equation as

$$\Delta P^* = \frac{\rho U^2}{2} - \frac{\rho}{2}(\Psi_y^2 + \Psi_x^2) = T \frac{1}{r_i} \quad (2.21)$$

where $U$ is the upstream velocity and $\rho$ is the density of the fluid.
In $\Omega_2$, many authors advance the argument that the drainage resistance of the fiber mat varies in the machine direction. To help address this issue, Moch [73] assumed an empirical relation. Green and Kerekes [74] assumed that the web grows linearly with distance. Martinez [72], on the other hand, derived a diffusive-type differential equation for the solidity $\phi$ in the web using the equations of motion as a starting point. This is given by

$$U \frac{\partial \phi}{\partial \xi} = \frac{\partial}{\partial \eta} \left( D(\phi) \frac{\partial \phi}{\partial \eta} \right) - J(\xi) \frac{\partial \phi}{\partial \eta}$$

where $D(\phi)$ is a function of the compressibility and permeability of the network; and $J(\xi)$ is the dewatering rate. In this derivation, Martinez assumed that the fibres and water move at essentially the same velocity in the machine direction. By doing so, Martinez neglects the viscous stress in his formulation. Zahrai et al. [75] presented the only numerical solution of the coupled problem between domains $\Omega_1$ and $\Omega_2$.

Finally, although these models have engineering value, and can predict the pressure acting along the length of the paper machine and the subsequent dewatering rate, they give no insight into the development of the physical properties of paper. Jansson [76], for example, demonstrated that fibres are randomly oriented on the surface and in the center of paper but become more preferentially oriented away from these three planes of the sheet. Similarly, the optical appearance of the uniformity, traditionally termed the formation, appears less grainy under certain drainage conditions. Both these phenomena cannot be explained by the mathematical formulation given above as the conditions between the suspension and the fibre mat have not been adequately considered in the previous formulations.

## 2.3 Objectives of the Research

The objectives of this work are:

1. To establish a model for the flow of fluid both through and above a porous medium.

2. To develop a robust algorithm to solve the governing equations.
3. To verify the model experimentally by measuring velocity profiles in a laboratory device.
Chapter 3

MODEL EQUATIONS

3.1 Introduction

The geometry considered is shown in Fig. 3.1. The channel consists of a fluid (without any fibers) bounded below by an isotropic non-deformable porous fiber mat, of permeability $K$ and porosity $\epsilon$, and bounded above by a rigid impermeable plate. The total height of the channel is $H$ and the thickness of the clear fluid region is $h$. At the bottom of the porous material (the porous wall), the fluid is withdrawn or injected at a uniform velocity $U_w$.

It is to be noted that the case considered here, although initiated from a pulp and paper application, also occurs in a wide range of fluid processes such as membrane separation, journal bearing lubrication, biological transport processes (blood flow), and solidification of metal alloy melts during processing.

The underlying phenomena and the interactions between the clear fluid and the porous system are diverse and complex. Before presenting the model equations, it is instructive to examine the equations of motion for flow in a porous medium, see for instance the review of Nield and Bejan [77]. Based upon the observations of steady unidirectional flow in uniform porous media, Darcy [78] suggested a linear relationship between velocity of discharge and the pressure gradient in the fluid. In vector notation, this relationship can be expressed as

$$\nabla P^* = -\frac{\mu}{K} \bar{u}^*$$  \hspace{1cm} (3.1)

where $\bar{u}^*$ is the so-called Darcy velocity or seepage velocity (the average of the fluid velocity over a volume element containing both the fluid and solid matrix); Darcy's law is limited
to flow where the Reynolds number, based upon the average grain or pore diameter, is of order unity or smaller.

It is widely known that it is difficult to apply Darcy’s law near the interface between a highly porous medium and a viscous fluid. The difficulty in this case is that in the clear fluid region, the flow is governed by the Navier-Stokes equations that are second-order differential equations, whereas the Darcy equation is only first-order. An intuitive means of resolving this difficulty is to modify the Darcy’s law relationship by including a second-order viscous term. Brinkman [79] was the first to propose this modification and the corresponding equation is known as the Brinkman-extended Darcy equation. Wooding [80] has extended this equation further by adding a convective acceleration term. He does this as the curvature of the streamlines in the interface area may be of the order of the pore or particle diameter and this may lead to non-zero values of the convective acceleration term $(\bar{u}^* \cdot \nabla) \bar{u}^*$. In our notation, the equations of motion for steady incompressible flow in a porous medium are given by

\[ \nabla \cdot \bar{u}^* = 0 \]  \hspace{1cm} (3.2)
\[ \frac{\rho}{\epsilon} (\bar{u}^* \cdot \nabla) \bar{u}^* = -\epsilon \nabla P^* + \epsilon \mu_e \nabla^2 \bar{u}^* - \epsilon \frac{\mu}{K} \bar{u}^* \]  \hspace{1cm} (3.3)

These equations closely resemble the form of the Navier-Stokes equations. The coefficient $\mu_e$ is the “effective viscosity” of the porous medium and is not, in general, the same as the fluid viscosity. Ochoa-Tapia and Whitaker [81, 82] suggested that $\mu_e = \mu/\epsilon$. For
cases where high local velocities prevail in the porous medium, i.e., when the pore-scale
Reynolds number is increased in the range 1 to 10, the linearity of the Darcy surface drag
breaks down. Experimental results are available that show the transition to nonlinearity
[83]. This transition has been characterized by adding a quadratic term in the right-hand
side of the above equation. In vector notation, this term, is usually referred to as the
Forchheimer term, and is expressed as $\rho C_f K^{-1/2} |\vec{u}^*|^2 \vec{u}^*$, where $C_f$ is a dimensionless form
drag constant [77, 83, 84]. This has been studied by Lage [85] and Manole and Lage [86]
and they advance the argument that the convective inertial term is normally small when
compared to the Forchheimer term when Reynolds number is higher than 1. This term is
not considered in this work.

At the porous medium /fluid interface, it has been experimentally confirmed by Beavers
and Joseph [87, 88] that when a viscous fluid flows past the surface of a porous medium, the
effects of viscous shear in the clear fluid will penetrate beneath the permeable surface to
form what is effectively a boundary layer region in the porous medium. The Darcy equation
is clearly not compatible with the existence of a boundary layer region in the porous
medium because no macroscopic shear term is associated with the equation [89]. The work
of Beavers and Joseph [87, 88] was one of the first to address this issue. Beavers and
Joseph circumvent this fundamental inadequacy of the Darcy equation near the interface
region by proposing an empirical mathematical model in which the boundary layer could
be approximated by a slip velocity, i.e., the velocity across the interface need not be
continuous. The magnitude of this slip was postulated to be directly proportional to the
prevailing shear stress, namely

$$v_x^* - u_x^* = \frac{\sqrt{K}}{\vartheta} \frac{\partial v_x^*}{\partial y^*}$$  \hspace{1cm} (3.4)$$

where $v_x^*$ and $u_x^*$ denote the $x$-component of velocity within the channel and porous medium,
respectively; and $\vartheta$ is a dimensionless constant experimentally determined to range between
0.1 $\leq \vartheta \leq$ 4. Some theoretical support for this condition is provided by Taylor [90] and
Richardson [91] based upon an analogous model of a porous medium, and by the statistical
treatments of Saffman [92].

Neale and Nader [89], as well as Vafai and Thiyagaraja [93] and Vafai and Kim [94],
have questioned the validity of the "slip velocity", i.e., there cannot physically be any slip between the fluid in the channel and the fluid within the porous medium in an actual velocity profile. They proposed that both the velocity and shear stress must be continuous at the interface, i.e.

\[
\begin{align*}
    u_x^* &= v_x^* \\
    \mu_c \frac{\partial u_x^*}{\partial y^*} &= \mu \frac{\partial v_x^*}{\partial y^*}.
\end{align*}
\]

To incorporate this extra boundary condition, these authors employed the Brinkman-extended Darcy model to describe the flow through the porous medium. Recently, Ochoa-Tapia and Whitaker [81] utilized a sophisticated volume averaging technique to extend this formulation. They recognized that part of the viscous stress in the fluid region is taken by the porous matrix in porous region across the interface, i.e., the Darcy term related stress. They deduced the form of this Darcy stress (so called the stress jump) from a rigorous stress function analysis for the fluid and porous regions. In their study the jump in shear stress is inversely proportional to the square root of the permeability of the porous medium,

\[
\begin{align*}
    \frac{1}{\epsilon} \frac{\partial u_x^*}{\partial y^*} - \frac{\partial v_x^*}{\partial y^*} &= \frac{\beta}{\sqrt{K}} u_x^*.
\end{align*}
\]

where \( \beta \) is an empirical constant. Ochoa-Tapia and Whitaker derived that the value of \( \beta \) is of the order of the ratio of the thickness of the boundary region near the interface to the square root of the permeability, or its inverse, therefore should be \( O(1) \), and it may be positive or negative as the difference in stress functions can be positive or negative. It follows that Eq. (3.6) is a special case of Eq. (3.8) when \( \beta = 0 \) and \( \mu_c = \mu/\epsilon \). Ochoa-Tapia and Whitaker’s [81] proposed interface conditions have been adopted recently by Kuznetsov [95, 96, 97], and Alazami and Vafai [98]. Ochoa-Tapia and Whitaker [99] also includeed a quadratic jump condition \( \beta_2 \rho u_x^2 \) for flow where inertial effects are evident. This term is not considered in this work.
3.2 Model Equations

A Cartesian co-ordinate system \((x^*, y^*)\) is used to describe the channel shown in Fig. 3.1. Here \(x^*\) measures the distance along the length of the channel and \(y^* \in (0, H)\). The assumptions made in this analysis are as follows:

1. The variation of quantities in the span-wise direction are neglected and the problem is considered two-dimensional;

2. The fluid is Newtonian;

3. The fluid is viscous and the flow laminar;

4. The fluid is incompressible and the body forces are negligible;

5. The porosity \(\epsilon\) is uniform;

6. The governing equation for the fluid region is the Navier-Stokes equation; the Brinkman extended Darcy’s law with the inertial terms retained is used in the porous region \([77, 100]\);

7. The interface boundary conditions used are those outlined by Ochoa-Tapia and Whitaker \([81]\).

When the independent variables are scaled with \(H\), the flow components in each domain by \(U_w\), and the pressure with \(\rho U_w^2\), the equations of motion in dimensionless form reduce to

\[
\begin{align*}
\nabla \cdot \vec{v} &= 0 \\
\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla) \vec{v} &= -\nabla P + \frac{1}{Re} \nabla^2 \vec{v} \\
\nabla \cdot \vec{u} &= 0 \\
\frac{\partial \vec{u}}{\partial t} + \frac{1}{\epsilon} (\vec{u} \cdot \nabla) \vec{u} &= -\epsilon \nabla P + \frac{1}{Re} \nabla^2 \vec{u} - \frac{\epsilon}{Re Da} \vec{u}
\end{align*}
\]

(3.9)

where \(\vec{v}\) and \(\vec{u}\) represent the flow velocities in the clear fluid and porous regions respectively; \(x\) and \(y\) represent the channel coordinates (dimensionless); \(Da\) is the Darcy number defined by \(Da = K/H^2\); \(Re\) is the Reynolds number \((Re = U_w H/\nu)\); and \(\chi\) is the interface
position defined as $\chi = h/H$. The boundary conditions at the upper and lower walls are straightforward and given by

$$\begin{align*}
\bar{v}(x, 0, t) &= (0, 0) \\
\bar{u}(x, 1, t) &= (0, 1)
\end{align*}$$

(3.10)
i.e. no slip. At $y = \chi$, we use the conditions outlined by Ochoa-Tapia and Whitaker [81]. In non-dimensional form, these are given by

$$\bar{v} = \bar{u}$$

(3.11)

$$\frac{1}{\epsilon} \frac{\partial u_x}{\partial y} - \frac{\partial v_x}{\partial y} = \frac{\beta}{\sqrt{Da}} u_x$$

(3.12)

$$\tau_+ = \tau_-$$

(3.13)

where $\beta$ is an empirical constant [81] which must be determined independently; and the subscripts $+$ and $-$ refer to the position above and below the interface, respectively. Equation (3.13) indicates an equilibrium of the normal stresses at the interface. Equations (3.9) through (3.13) represent the model equations.
Chapter 4

SIMILARITY SOLUTIONS

Before a solution to the model equations is given, it is instructive to briefly review other geometries which resemble the flow considered. This will aid in understanding the form of the similarity ansatz considered in this thesis. To begin, this literature is categorized into two groups. The major works in category (I) [81, 82, 87, 89] are fully-developed flows with $U_w = 0$. In general, these works focus on understanding the underlying phenomena and the interactions between the free fluid and the porous system. This literature has been reviewed in Chapter 3.

In the second category (II) are flows with $h = H$. This body of knowledge begins with the classic work of Berman [3] who investigated the problem of flow in a channel with both channel walls equally permeable. Berman assumed that the solutions were symmetrical about the centreline of the channel and reduced the two-dimensional Navier-Stokes equation to a fourth-order non-linear ordinary differential equation using a similarity transform. This equation depends upon a single non-dimensional number, the transverse Reynolds number $Re$ defined in terms of the channel width and suction velocity. Similarity solutions were tested in early experiments, where evidence of good agreement between measurements and theoretical predictions are reported [101]. Many authors have extended Berman’s symmetric series solution [102, 103, 104, 105, 106, 107, 108], and a very rich structure of solutions is evident in this literature.

Recently, Cox and his co-workers [4, 109, 110, 111] considered in detail the asymmetric case of $h = H$ with suction or injection at only one wall, while the other wall is
impermeable. After scaling the Navier-Stokes equation by \( H/2 \) and \( U_w \), they formulate the steady-state problem using the stream function \( \Psi \) of a form originally suggested by Berman, i.e. \( \Psi(x, y) = -xf(y) \). After elimination of pressure from the Navier-Stokes equation, the separable component \( f(y) \) must satisfy the following equation

\[
f'''' + \text{Re}(f'f'' - ff''') = 0 \tag{4.1}
\]

subject to the boundary conditions

\[
f(-1) = f'(-1) = f'(1) = 0, \quad f(1) = 1. \tag{4.2}
\]

Here \( y = -1 \) represents the impermeable wall and \( \text{Re} = U_w H/2\nu \). Cox has shown that there is a unique steady solution for all values of \( \text{Re} \), except in the range \( 7.05 < \text{Re} < 7.31 \) where there are two stable solutions.

In this chapter, we build on the work of Cox and his co-workers [4, 109, 110, 111], and extend their work by considering coupled-flow both through and above a porous medium (i.e., \( h \neq H \)) by extending the similarity ansatz. It is often found that similarity solutions lead to regions of no solutions or multiple solutions. Keeping in mind the limitations of similarity solutions as outlined by Brady and Acrivos [112], we present in §4.1 a general similarity formulation that depends on a number of independent dimensionless parameters. In §4.2, the steady-state solutions are considered in which both low-Reynolds number expansions and numerical solutions are presented. A linear stability analysis is conducted using the method of normal modes in §4.3. Concluding remarks are given in §4.4.

### 4.1 Similarity Formulation

We reduce our model to something more tractable than Eqs. (3.9)-(3.13) by assuming the form of the solution. Following the procedure given by Berman [3], we postulate that the stream function \( \Psi(x, y, t) \) may be written as a product

\[
\Psi(x, y, t) = (\bar{V} - x) \left\{ \begin{array}{ll}
\hat{f}(y, t) & \forall(t, x, 0 \leq y \leq \chi) \\
\hat{g}(y, t) & \forall(t, x, \chi \leq y \leq 1) 
\end{array} \right. \tag{4.3}
\]
where \( \bar{V} \) denotes the average (dimensionless) axial velocity at \( x = 0 \); and \( \hat{f} \) and \( \hat{g} \) are dimensionless time dependent functions which will be subsequently determined. It can be shown from an integral mass balance that \( \bar{V} - x \) represents the average axial velocity in the channel at any downstream location \( x \).\(^1\) The velocity components follow immediately from the definition of \( \Psi \), i.e.

\[
\bar{v} = ((\bar{V} - x)\hat{f}(y, t), \hat{f}(y, t)), \quad \bar{u} = ((\bar{V} - x)\hat{g}(y, t), \hat{g}(y, t)) \tag{4.4}
\]

where the subscript \( y \) or \( t \) denotes the partial derivative with respect to \( y \) or time \( t \), respectively. After elimination of pressure from the equations of motion by cross differentiation, we find that the functions \( \hat{f} \) and \( \hat{g} \) must satisfy

\[
\hat{f}_{yyt} = \frac{1}{Re} \hat{f}_{yyyy} + \left( \hat{f}_y \hat{f}_{yy} - \hat{f}_{yyyy} \right) \tag{4.5}
\]

\[
\hat{g}_{yyt} = \frac{1}{Re} \hat{g}_{yyyy} + \frac{1}{\epsilon} \left( \hat{g}_y \hat{g}_{yy} - \hat{g}_{yyyy} \right) - \frac{\epsilon}{DaRe} \hat{g}_{yy} \tag{4.6}
\]

The boundary conditions at the upper and lower surfaces are

\[
\hat{f}(0, t) = 0, \quad \hat{f}_y(0, t) = 0, \quad \hat{g}(1, t) = 1, \quad \hat{g}_y(1, t) = 0. \tag{4.7}
\]

and at the interface \( y = \chi \),

\[
\hat{g} = \hat{f}, \quad \hat{g}_y = \hat{f}_y, \quad \hat{g}_{yyyy} = \epsilon \left( \hat{f}_{yyyy} + \frac{\beta}{Da} \hat{f}_y \right), \tag{4.8}
\]

\[
\frac{1}{Re} \hat{f}_{yyyy} - \hat{f}_y \hat{f}_{yy} + \hat{f}_y^2 - \hat{f}_{yt} = \frac{\hat{g}_{yyyy}}{Re} - \frac{\hat{g}_y \hat{g}_{yy}}{\epsilon^2} + \frac{\hat{g}_y^2}{DaRe} - \frac{\hat{g}_y}{\epsilon} \hat{g}_{yt}. \tag{4.9}
\]

Equations (4.5) through (4.9) represent a considerable reduction in the complexity of the model equations. It should be noted that the derivation of Eq. (4.9) is difficult and details of this are given in Appendix A.

### 4.2 Steady-State Solutions

In this section we describe the steady solutions for \( f(y) \) and \( g(y) \). In §4.2.1 we present some asymptotic results for \( Re \rightarrow 0 \) using regular perturbation methods. By doing so, we

\(^1\)Assume that the the dimensional average axial velocity at position \( (x^*, y^*, \tau^*) \) is \( \bar{v}_x^* \), the mass balance show that \( \bar{V}^*H = \bar{v}_x^*H + U_wx^* \), hence \( \bar{v}_x^* = \bar{V}^* - U_wx^*/H \), which in dimensionless form, is \( v_x = \bar{V} - x \).
provide results about the basic flow and show that the solution diverges when $Da/\epsilon \rightarrow 0$. In §4.2.2 we have also obtained solutions for small $Re$ using a power series approximation. The prerequisite of this method, however, depends on an appropriate re-scaling of the original equations. In §4.2.3 we describe a numerical technique for solving the equations and show that, at large $Re$, the solution bifurcates.

### 4.2.1 Asymptotic Solutions

To begin, we seek a steady solution of the similarity equations of the form

$$f(y) = \sum_{n=0}^{\infty} Re^n f_n(y)$$
$$g(y) = \sum_{n=0}^{\infty} Re^n g_n(y).$$

With this, the leading terms in these series are readily found to be

$$f_0(y) = \frac{1}{2} C_1 y^2 + \frac{1}{6} C_2 y^3,$$
$$g_0(y) = C_3 + C_4 y + C_5 e^{y/\sqrt{Da}} + C_6 e^{-y/\sqrt{Da}}$$

where $C_i$ are the constants of integration. It should be noted that we have applied the boundary conditions at the upper wall. The point of this exercise is that upon examination of the leading terms, we would anticipate difficulties in the limit of $Da/\epsilon \rightarrow 0$. Clearly, under this condition the form of the solution dictates that $C_5$ must be zero; otherwise, $g$ would grow exponentially with $y$ making it unmatchable with the boundary conditions at $y = 1$. In the limit of $Da/\epsilon \rightarrow 0$, it can be shown that, after a modest number of algebraic manipulations, the flow field is given by

$$f_0(y) = 3\left(\frac{y}{\chi}\right)^2 - 2\left(\frac{y}{\chi}\right)^3 \quad \text{and} \quad g_0(y) = 1$$

When the magnitude of $e^{y/\sqrt{Da}}$ is not large, the constants of integration are determined from the remaining boundary conditions. This leads to a linear system of algebraic
equations, i.e.
\[
\begin{bmatrix}
\chi^2/2 & \chi^3/6 & -1 & -\chi & -\epsilon^\chi & e^{-\epsilon^\chi} \\
\chi & \chi^2/2 & 0 & -1 & -\zeta e^{\epsilon^\chi} & \zeta e^{-\epsilon^\chi} \\
\epsilon & \epsilon \chi & 0 & \sqrt{\epsilon \beta} \zeta & B_1 & -B_2 \\
0 & 1 & 0 & 0 & -\frac{\epsilon^\chi}{\epsilon} & \frac{\epsilon^\chi}{\epsilon} \\
0 & 0 & 1 & \chi & e^\epsilon & e^{-\epsilon} \\
0 & 0 & 0 & 1 & \zeta e^\epsilon & -\zeta e^{-\epsilon}
\end{bmatrix}
\begin{bmatrix}
C_1 \\
C_2 \\
C_3 \\
C_4 \\
C_5 \\
C_6
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
1 \\
0
\end{bmatrix}
\]
(4.13)

where intermediate constants in the above matrix are defined for brevity, i.e.
\[
\zeta = \sqrt{\epsilon/Da}, \\
B_1 = (\sqrt{\epsilon \beta} - 1)\zeta^2 e^{\epsilon \chi}, \\
B_2 = (\sqrt{\epsilon \beta} + 1)\zeta^2 e^{-\epsilon \chi}.
\]

This system of algebraic equations can be solved using standard techniques and the leading order estimates of the \( x \) component of the flow field, represented by \( f_y \) and \( g_y \) are given in Fig. 4.1(i). Here, it is shown that the velocity in the clear fluid region joins that of flow in the porous medium across the interface region. The curvature of the profile at the interface decreases with increasing \( Da/\epsilon \). Due to the momentum of the fluid in the clear fluid region, fluid penetrates into the porous medium and decays rapidly to zero within the porous wall for a low permeability medium. We term the depth of the fluid penetration as the thickness of a “boundary layer”, i.e., the thickness from the interface position to where the \( x \) component velocity \( g_y(y) \) is 0.01. This thickness is normalized with the total thickness of the porous medium and shown in Fig. 4.1(ii). Clearly with \( Da/\epsilon > 10^{-3} \), the porous medium offers very little resistance to the fluid’s motion, and the fluid penetrates at least 90 percent into the porous region. When \( Da/\epsilon \to \infty \) the flow in the channel behaves like a fully-developed Poiseuille flow. On the other hand, with \( Da/\epsilon \to 0 \), there is no longitudinal velocity in the porous medium and the boundary layer diminishes to zero.

It is interesting at this point to examine the solution at higher \( Re \) and we do so by computing one further term in the expansion. This was performed in Maple, a symbolic
Figure 4.1: (i): The x-component of the velocity profiles $f_y$ and $g_y$ (normalized by the averaged velocity $\bar{V} - x$ at position $x$) estimated in the limit of $Re \rightarrow 0$. Four curves are shown: (a) $Da/\epsilon \rightarrow 0$, (b) $Da/\epsilon = 5 \times 10^{-4}$, (c) $Da/\epsilon = 1 \times 10^{-2}$, and (d) $Da/\epsilon \rightarrow \infty$. (ii): An estimate of the boundary layer thickness in the porous medium. The boundary layer thickness has been normalized to the thickness of the porous medium. These estimates were made with $\beta = 0.2$, $\chi = 0.8$ and $\epsilon = 0.9$. 
Figure 4.2: The effect of $Re$ for the case $Da = 1 \times 10^{-3}$ on the $x$ component velocity profiles, $f_y$ and $g_y$, here $\beta = 0.2$, $\chi = 0.8$, $\epsilon = 0.9$. 
CHAPTER 4. SIMILARITY SOLUTIONS

solver, and the resulting algebraic expressions to the general case are too large to be reproduced here. To illustrate this point, the leading order and the first order approximations are given in Appendix B for one particular case, namely with \( Da = 1 \times 10^{-3}, \chi = 0.8, \epsilon = 0.9 \) and \( \beta = 0.2 \). The effect of \( Re \) is also shown in Fig. 4.2. Here we see that, with increasing \( Re \), the location of the maximum longitudinal velocity in the channel shifts towards the porous wall. As we proceed with higher order approximations, we found that the exponential form of the solution within the porous region makes the explicit solutions difficult.

4.2.2 Series Solutions

Equations (4.5)-(4.9) are difficult to solve over wider range values of \( Re \) and \( Da \). When \( Da \) is low, the equations become stiff and consequently difficult to be solved directly. The solution method considered here relies crucially on re-scaling \( f(y), g(y) \) and \( y \) so that the governing equations do not contain the Reynolds number and Darcy number.

If

\[
\begin{align*}
y &= z/q \quad f(y) = q\lambda(z)/Re \\
g(y) &= \epsilon q\psi(z)/Re
\end{align*}
\]

where \( q = \sqrt{\epsilon/\text{Da}} \), the governing equations reduce to

\[
\begin{align*}
\lambda_{zzzz} &= \lambda\lambda_{zz} - \lambda_z\lambda_{zz} \\
\psi_{zzzz} &= \psi\psi_{zz} - \psi_z\psi_{zz} + \psi_{zz}
\end{align*}
\]

subject to

\[
\begin{align*}
\lambda(0) &= 0, \\
\lambda_z(0) &= 0, \\
\lambda(\chi q) &= \epsilon\psi(\chi q), \\
\lambda_z(\chi q) &= \epsilon\psi_z(\chi q), \\
\psi_{zz}(\chi q) &= \lambda_{zz} + \frac{\beta}{\sqrt{\epsilon}}\lambda_z, \\
\psi_{zzz}(\chi q) &= \lambda_{zzz} - \lambda\lambda_{zz} + \lambda_z^2 + \psi\psi_{zz} - \psi_z^2 + \psi_z, \\
\psi(q) &= \frac{Re}{\epsilon q}.
\end{align*}
\]
\[ \psi_z(q) = 0. \] (4.17)

If we seek a solution of the form
\[
\lambda(z) = \sum_{n=0}^{\infty} a_n z^n
\] (4.18)
\[
\psi(z) = \sum_{n=0}^{\infty} b_n (\ddot{z})^n
\] (4.19)

where \( \ddot{z} = z - \chi q \). When the boundary conditions at the impermeable wall are evaluated, the first few terms of these series are given by
\[
\lambda(z) = \frac{1}{2} C_0 z^2 + \frac{1}{6} C_1 z^3 - \frac{1}{120} C_2 z^5 + O(z^6)
\] (4.20)
\[
\psi(z) = C_2 + C_3 \ddot{z} + \frac{1}{2} C_4 \ddot{z}^2 + \frac{1}{6} C_5 \ddot{z}^3 + \frac{1}{24} (C_2 C_5 - C_3 C_4 + C_4) \ddot{z}^4 + \frac{1}{120} (C_2^2 C_5 - C_2 C_3 C_4 + C_2 C_4 + C_5 - C_4^2) \ddot{z}^5 + O(\ddot{z}^6)
\]

where \( C_0, C_1, C_2, C_3, C_4 \) and \( C_5 \) are constants which need to be determined via the remaining six boundary conditions using Newton's method, i.e., a method described in [113]. A representative solution is given in Fig. 4.3. We find that the series solution yielded sensible results in the region \([Re, Da] = [-5, 5] \times [10^{-4}, 1]\).

### 4.2.3 Numerical Solutions

Both solutions presented thus far are useful when \(-5 < Re < 5\), where a negative Reynolds number indicates an injection case, due to the sign change of the velocity at the bottom wall. Outside this region, a higher order approximation is required. As a result we pursue a numerical solution for a wider range of Darcy and Reynolds numbers. In general, we recognize that this is a three-point boundary value problem. We developed two different numerical methods in an attempt to solve this problem. The first algorithm is based on a combination of shooting and Newton’s methods [113]. This method is simple and does not require good initial guess of the solution. This method, however, was found useful only in a range slightly larger than the series solution presented in the previous section. As a result, we developed a second algorithm, based upon the work of Terrill [104], which we call the “reverse solver”. The algorithm was found to be robust for most values of \( Re \) and \( Da \). Both algorithms will be discussed below.
Figure 4.3: The 6th order series solution (−) compared to the similarity numerical solution (•) with $Da = 1 \times 10^{-3}$, $Re = 5$, $\beta = 0.2$, $\chi = 0.9$. 

CHAPTER 4. SIMILARITY SOLUTIONS

Method 1 - The Shooting Method

The first algorithm considers the steady case only, where all the non-dimensional parameters \( \epsilon, \chi, \beta, Da \) and \( Re \) are specified. The numerical procedure is as follows:

1. It is useful to note that Eq. (4.5) may be integrated once in \( y \) to give
   \[
   f_{yy} + Re \left( f_y^2 - ff_{yy} \right) = A_1
   \]
   where \( A_1 \) is the constant of integration (which is unknown at this point). In physical terms \( f_{yy}(0) = A_1 \). Further, we define a second constant \( f_y(0) = A_2 \) which will be used subsequently.

2. If we assume trial values for \( A_1 \) and \( A_2 \), we can treat Eq. (4.21) as an initial value problem and integrate this relationship with a fourth-order embedded Runge-Kutta solver over the domain \( 0 < y < \chi \) [114].

3. At \( y = \chi \) the interface conditions are evaluated using Eqs. (4.8) and (4.9) for a prescribed value of \( \beta \).

4. Equation (4.6) is now integrated over the domain \( \chi < y < 1 \) using the same Runge-Kutta method.

5. The solution with the trial values of \( A_1 \) and \( A_2 \) are compared to the boundary conditions at \( y = 1 \). Newton's method, a root finding procedure [113], is used to update the values of \( A_1 \) and \( A_2 \) until the boundary conditions at \( y = 1 \) are satisfied to within a tolerance of \( 1 \times 10^{-6} \).

Results from this technique yield very good agreement with the solutions obtained from the second numerical method for \(-60 < Re < 16\).

Method 2 - Reverse Solver

The "reverse solver" considered here is designed to solve the re-scaled Eqs. (4.15) to (4.17). By "reverse", we mean that, in the traditional manner of solution, the parameters

\[ ^2 \text{A method to determine } \beta \text{ will be considered in Chapter 5.} \]
CHAPTER 4. SIMILARITY SOLUTIONS

Re and Da are specified and the equations are integrated. The reverse approach involves a transformation of the governing equations and constructing plausible solutions without specifying Re and Da beforehand. Once a solution is constructed, the corresponding Re and Da are determined.

The algorithm is summarized below:

1. Equation (4.15) may be integrated once in \( z \) to give

\[
\lambda_{zzz} + (\lambda_z^2 - \lambda_{zz}) = A_3
\]

where \( A_3 \) is the constant of integration. In physical terms \( \lambda_{zzz}(0) = A_3 \). At this point, we define a second constant \( \lambda_z(0) = A_4 \).

2. We begin the numerical procedure by choosing trial values of \( q, A_3 \) and \( A_4 \).

3. With known values of \( A_3 \) and \( A_4 \), Eq. (4.22) may be solved as an initial value problem using a fourth-order Runge-Kutta method over the domain \( 0 \leq z \leq q \chi \).

4. At \( z = q \chi \) the interface conditions are evaluated using Eq. (4.17) for a prescribed value of \( \beta \).

5. Equation (4.16) may now be integrated over the domain \( q \chi \leq z \leq q \) as an initial value problem using the same method mentioned above.

6. At this point we compare the computed value of \( \psi(q) \) and \( \psi_z(q) \) with the boundary conditions given in Eq. (4.17). We use Brent’s method [115], a root finding procedure to update the final value of \( q \) until we find the point where \( \psi_z(q) = 0 \). It should be noted that under certain combinations of \( A_3 \) and \( A_4 \), we can find two roots. Each root represents a solution to the problem as discussed by Cox and King [111].

7. The solutions with the trial values of \( A_3 \) and \( A_4 \) are now compared to the targeted Darcy and Reynolds numbers and a similar iterative procedure is then invoked until convergence is achieved.

\(^3\)The initial guess was estimated from the results of the shooting method. A linear/nonlinear (e.g., "spline" method) extrapolation was used to estimate \( A_3 \) and \( A_4 \). The technique used is similar to a branch tracing method commonly used in solving nonlinear equations.
CHAPTER 4. SIMILARITY SOLUTIONS

With this method we are able to find solutions for a wide range of Reynolds numbers and Darcy numbers. We validated our solution scheme with solutions from (i) King and Cox [4] for the case when Da is high (e.g., Da = 1000, see Fig. C.1), (ii) the asymptotic expansion in the region \([Re, Da] = [-4.5, 4.5] \times [10^{-4}, 1]\) (see e.g., Fig. B.1) and (iii) the series solution when \([Re, Da] = [-5, 5] \times [10^{-4}, 1]\) (see Fig. 4.5, for example). Good agreement was found for all cases. In fact, we were able to reproduce identically the bifurcation reported by King and Cox [4]. A series of numerical experiments were conducted over the domain \([Re, Da] = [0, 170] \times [0.013, 0.05]\) as this represents the region of interest for papermaking applications [116, 117]. In general, three types of solution behaviors were found.

- type I - no backflow,
- type II - backflow in the channel, no backflow in the porous region,
- type III - backflow in both domains.

Examples of these are shown in Fig. 4.4(a)-(c). The interesting feature here is that under certain conditions, the solutions found were not unique. In other words, we found multiple steady states for certain combinations of \(Re\) and \(Da\). To help illustrate this behavior, a plot of the viscous stress \((-f_{yy}(\chi))\) \(^4\) is given in Fig. 4.5. In this figure, the shear stress is shown as a function of \(Re\) at a constant \(Da\). In regions where multiple steady states were found, there exists multiple values for the shear stress. As shown, multiple solutions were found over the entire domain with the upper branch displaying a turning point bifurcation in the region \(16.541 < Re < 17.169\) and at \(Re = 54.722\). The number and the stability of the solutions depend upon both \(Re\) and \(Da\). The stability of each branch will be discussed shortly. Included in this figure are the regions in which each type of solution exists. Part of the results for all simulations conducted is shown in Fig. 4.6 as a surface plot. To help simplify this figure only type-I and II solutions are included. We do so as type-III solutions are mostly unstable.

\(^4\)For a two-dimensional flow, the shear stress can be expressed as \(\tau^* = \mu (\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y})\). When \(\tau^*\) is scaled with \(\rho U^2\), the non-dimensional form of the shear stress becomes \(\tau = \frac{1}{Re} (\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y})\). Substitution of Eq. (4.4) yields \(\tau(\chi) = \frac{1}{Re} (\bar{V} - x) f_{yy}(\chi)\) at the interface. Hence, a reasonable representation of the shear stress at the interface is \(-f_{yy}(\chi)\).
Figure 4.4: Example solutions: The plot represents the three different families of solutions obtained, here (a): $Re = 10.93$, (b): $Re = 16.68$ and (c): $Re = 32.21$. These simulations were conducted at $Da = 0.05$, $\beta = 0.2$, $\chi = 0.8$ and $\epsilon = 0.9$. The shaded area represents the porous region.
Figure 4.5: Example solutions: The figure represents the shear stress at the interface \(-f_{yy}(\chi)\) (here the circles indicate the series solutions given in §4.2.2). These simulations were conducted at \(Da = 0.05, \beta = 0.2, \chi = 0.8\) and \(\epsilon = 0.9\). The type of solution is indicated, where the line "|" shows the transitions from type-I to type-II and from type-II to type-III solutions. Here the solid-lines indicate stable solutions while the dotted region represents unstable solutions.
Figure 4.6: A 3-dimensional plot of the solutions at $Da$ and $Re$ considered, here $\beta = 0.2$, $\chi = 0.8$, $\epsilon = 0.9$. 
4.3 Stability of the Similarity Solutions

The temporal stability of the steady solutions shown in the previous section is analyzed by introducing small amplitude time-dependent perturbations of the same similarity form suggested in Eq. (4.4). The method we used is called the method of normal modes, which has also been used in the work of [107] and [109]. We consider the solutions are of the form:

\[
\begin{align*}
\tilde{f}(y,t) &= f(y) + e^{\sigma t} F(y) \\
\tilde{g}(y,t) &= g(y) + e^{\sigma t} G(y)
\end{align*}
\]

where \( \sigma = \sigma_r + i\sigma_i \) is complex. Upon substitution of these relationships into Eqs. (4.5)-(4.9), after linearization, the following characteristic value problem for the perturbations \( F(y) \) and \( G(y) \) is obtained.

\[
\begin{align*}
F_{yyyy} - \text{Re} (f F_{yy} + f_{yy} F - f_y F_{yy} - f_{yy} F_y) &= \sigma \text{Re} F_{yy} \\
G_{yyyy} - \frac{\epsilon}{Da} G_{yy} - \text{Re} (g G_{yy} + g_{yy} G - g_y G_{yy} - g_{yy} G_y) &= \sigma \text{Re} G_{yy}
\end{align*}
\]

with

\[
\begin{align*}
F(0) = F_y(0) = G(1) = G_y(1) = 0
\end{align*}
\]

and

\[
\begin{align*}
F &= G \\
F_y &= G_y \\
G_{yy} &= \epsilon \left( F_{yy} + \frac{\beta}{\sqrt{Da}} F_y \right) \\
\left( \frac{F_{yy}}{\text{Re}} - f F_{yy} - f_{yy} F \right) + 2f_y F_y &= \sigma F_y = \frac{G_{yy}}{\epsilon \text{Re}} - \frac{1}{\epsilon^2 g G_{yy}} \\
-\frac{1}{\epsilon^2 g_{yy} G} &= \frac{1}{Da \text{Re}} G_y - \frac{\sigma}{\epsilon} G_y + \frac{2g_y G_y}{\epsilon^2}
\end{align*}
\]

at the interface \( y = \chi \). These equations form an eigenvalue problem. For the suction problem, a given solution is stable if for every eigenvalue \( \sigma_r < 0 \) is obtained. In the case of injection, instead, all eigenvalues must have a positive real part for the solution to be
stable. This is due to the sign change in the reference time \( h/U_w \), since \( U_w < 0 \) in this case.

These equations were solved using a root finding procedure similar to that outlined by Durlofsky and Brady [106] and Zaturska et al. [107], that is, a shooting method combined with Newton's method. A representative example is shown in Fig. 4.7. The shear stress is shown in the top panel of the figure. There are two regions of interest where multiple solutions were found on each branch. These are labelled Region I and Region II. Region I is magnified in Fig. 4.7(ii) and, for this region, \( \sigma_r \) is shown in Fig. 4.7(iii). Clearly, for the dotted region in the range \( 19.731 < Re < 19.833 \), \( \sigma_r > 0 \) and, hence, this represents an unstable solution. The analysis was conducted over the entire domain. Solutions which were found to be stable are presented as a solid line in Fig. 4.7(i).

A summary of the solution behavior is shown in Fig. 4.8 for all \( Re \) and \( Da \) tested. Included with this is a supplemental table (Table 4.1) detailing the nature, number and stability of each solution region. The first observation that can be made from this figure is that multiple solutions exist for all \( Da \) and \( Re \) considered. Unique stable solutions were found in regions \( A, B, D, G \) and \( H \). Regions with multiple stable solutions were generally type-II solutions. These were found in regions \( C, E \) and \( F \). No solutions could be found in region \( I \). To help illustrate the behavior as one crosses over a boundary in this diagram, the shear stress at three different Darcy numbers are shown in Figs. 4.9(a)-(c). In each figure, multiple solution regions labelled \( A, B \), etc. have different type, stability and number of solutions.

### 4.4 Summary

In this section we solved the equations of motion for flow in a channel partially filled with a porous medium using a similarity ansatz. Our numerical analysis has been fairly complete - we have applied regular perturbation techniques, power series approximations and a shooting method for small \( Re \), together with a numerical algorithm for higher Reynolds number solutions. Although not stated in the text, we have attempted to approximate the solution using singular perturbation methods in the limit of \( Re \rightarrow -\infty \) and \( Re \rightarrow +\infty \) but
Figure 4.7: The stability analysis for $Da = 0.013$: (i) is the steady solution, the detailed view in Region I is shown in (ii), whilst (iii) shows the real part of the corresponding eigenvalues in Region I. Here the dashed lines indicate unstable solutions.
Figure 4.8: (a) Summary of solution behavior for simulations with $\beta = 0.2$, $\chi = 0.8$ and $\epsilon = 0.9$. Nine different solution regions are found and the detailed information about each region is tabulated in Table 4.1. (b) Region $C$ is enlarged.
Table 4.1: The number, type and stability of the solutions for each solution region as shown in Fig. 4.8.

<table>
<thead>
<tr>
<th>Region</th>
<th>Type of Solutions</th>
<th>Region</th>
<th>Type of Solutions</th>
<th>Region</th>
<th>Type of Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No. soln&lt;sup&gt;a&lt;/sup&gt;</td>
<td>No. stable&lt;sup&gt;b&lt;/sup&gt;</td>
<td>No. soln</td>
<td>No. stable</td>
<td>No. soln</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<sup>a</sup> Total number of solutions at given \( Da \) and \( Re \).

<sup>b</sup> Total number of stable solutions at given \( Da \) and \( Re \).
Figure 4.9: Example solutions: the shear stress function $-f_{yy}(x)$ versus $Re$ for (a) $Da = 0.013$, (b) $Da = 0.026$ and (c) $Da = 0.0375$. The type of the solutions are denoted, and the multiple solution regions are labelled $A$, $B$, etc. representing different type, stability and number of solutions. Here solid-line denotes stable solution and dotted-line, the unstable solution.
were unsuccessful. Our results at moderate Reynolds numbers are worth highlighting, as we observed bifurcation. The temporal stability of the steady solutions for the problem has been analyzed using the method of normal modes. In this analysis, we represent the time dependent solutions as a sum of a base steady solution and a small time disturbance of similarity form. After linearization, we show that the system of equations is an eigenvalue problem. A shooting method combined with Newton’s root finding procedure was employed to solve the corresponding equations. Nine discrete solution regions were found. In each region, the number of the solutions, as well as the type and stability of these solutions were summarized.

It should be noted that, the similarity variable we selected for the similarity approach can not be extended for models which include the Forchheimer term and the nonlinear shear stress jump term in the interface condition, because, these two terms are quadratic which would leave the transformed differential equations dependent on the x component velocity.

Finally, we use the above results to estimate the shear stress at the interface of a twin-wire machine. As has been discussed, the shear stress at the interface at position $(x, y)$ can be represented as a product of the channel average velocity at $x$ and the second derivative of the function $f$, which depends only on the channel position $y$, i.e., by $\tau^* (\chi) = \frac{1}{Re} (\bar{V} - x)f''(\chi) \rho U_w^2$. Here $\bar{V}$ is the ratio of the inlet mean velocity of the suspension jet to the mean suction velocity. This expression shows that a high dewatering $Re$ reduces the shear activity near the interface and hence a higher jet wire speed difference is required to create more shear. At the early stage of the dewatering, when the formed paper web is very permeable, there may exist a certain suction $Re$ where no shear occurs at the interface, no matter what the jet/wire speed ratio is, as indicated in Fig. 4.5. On the other hand, when the web formed is less permeable, there seems to exist a wide region where high shear occurs at the interface region, as shown in Fig. 4.7(i). It is quite illustrative that, for any suction higher than $Re = 16$, there could be a backflow in the channel, which means flow recirculations that may contribute to flocculation and deterioration of paper formation.

Nevertheless, we roughly estimate from our available information that, the shear stress at the interface will be between 0 and 0.33 kPa for a twin-wire machine running at 1000
m/min with a jet/wire velocity ratio of 1.1 and a mean drainage velocity of 20 cm/s.
Chapter 5

ONE DOMAIN PROBLEM

5.1 Introduction

In Chapter 4, we divided the channel into two domains composed of an unobstructed fluid and a porous region, and solved the coupled problem by matching the constraints at the interface for an arbitrary given value of $\beta$. In this chapter, we consider that these two domains can be collapsed into one composite domain by re-posing the governing equations. By doing so, the interface conditions need not be considered. In this approach, a finite volume computational fluid dynamic (CFD) method was employed to directly solve the original steady transport equations for a finite length channel. To distinguish the two solution methods we employed, the method described in Chapter 4 is called the “similarity approach”, and the one in Chapter 5 is termed the “one-domain approach”. By comparing the output from both methods, we should be able to determine $\beta$.

5.2 One-Domain Formulation

In the one-domain approach, we re-write the steady equations of motion for a two-dimensional channel as

\[
\nabla \cdot \vec{w} = 0 \tag{5.1}
\]

\[
\frac{1}{\epsilon} (\vec{w} \cdot \nabla) \vec{w} = -\epsilon \nabla P + \frac{1}{Re} \nabla^2 \vec{w} - \frac{B \epsilon}{Re Da} \vec{w} \tag{5.2}
\]
where \( \bar{w} \) is the vector velocity in the channel,

\[
    B = \begin{cases} 
        0, & \text{in fluid region} \\
        1, & \text{in porous region} 
    \end{cases}
\]

\[
    \epsilon = \begin{cases} 
        1, & \text{in fluid region} \\
        0 < \epsilon < 1, & \text{in porous region} 
    \end{cases}
\]

We use the following boundary conditions: uniform inlet and fully developed flow at the outlet; no-slip condition on the upper impermeable wall boundary; and uniform suction on the lower porous boundary. The equations of motion were solved using a standard finite-volume formulation based on an artificial compressibility method (where a time derivative term is added in the formulation to facilitate the computing). We use the centered flux evaluations and implicit Euler time advance scheme for the discretization in space and in time respectively. Since the equations are nonlinear, we apply an approximate factorization method to convert them into two sets of coupled linear equations in the \( x \) and \( y \) directions (where for the porous region the sink term is added to the right-hand side fluxes and its Jacobian component is added to the left-hand side Jacobian matrix). The transformed sets of equations are essentially a \( 3 \times 3 \) block tridiagonal problem which can then be readily solved using a block Thomas algorithm. The convergence criterion is defined such that all changes in solution should be less than \( 1 \times 10^{-4} \). We use a uniform grid with the number of grid points being 500 in the flow direction and 80 in the normal direction, at which the grid independence is achieved. Details of the algorithm are presented in Appendix E.

A numerical study was conducted in which the solutions from the one-domain and two-domain approaches were compared in order to determine the relationship \( \beta(Re, Da) \).\(^1\) To do so, \( \beta \) was determined by minimizing the sum of the squared differences of the component velocities from these two numerical solutions. An example of this is shown in the lower panel of Fig. 5.1. The comparison between these two methods is based on the assumption that a similarity of the solution profiles is achieved after a certain length of flow.\(^2\) The pressure drop in the \( x \) direction \( \Delta P_x \) calculated from the similarity solution

\(^1\)The validation of the code is given in [118].

\(^2\)Note that \( \bar{V} \) needs to be set sufficiently large so that a fully-developed flow is achieved to be compared with the similarity solution. Here we set \( \bar{V} = 21 \) for the suction conditions we considered.
assumes a near-linear shape when the suction flow is small;\textsuperscript{3} it is plotted along with $\Delta P_x$ from the one-domain solution. As shown, with $x \geq 4$ the pressure drop from each solution is virtually indistinguishable. For this work we compare the flow profiles only when $x > 4$. At a position $x > 4$, the pressure drop in $y$ direction $\Delta P_y$ also shows good agreement, as both the similarity and the one-domain solutions predict that the pressure drop in $y$ direction occurs mainly in the porous region, see Fig. 5.1. The sensitivity of the solution to $\beta$ is shown using the similarity solution method in Fig. 5.2. Clearly large differences in the $x$ component of the velocity profile and the shear at the interface exist, when $\beta$ varies between $-1 < \beta \leq 1$. The best fit $\beta$ values from the comparison of the two solution methods are shown in Fig. 5.3. For the cases presented, $\beta$ was found to increase with decreasing $Da$ and increasing $Re$. In all cases tested $\beta \sim O(1)$ and is mostly positive.

5.3 Summary

In this chapter we developed a one-domain approach to solve the steady governing equations in a primitive variable form. In this method, the clear fluid domain and the porous region are considered as a composite domain governed by a single transport equation, the Brinkman extended Darcy's law with an inertial term. This equation has the same form as the Navier-Stokes equation, but with a sink term that can be turned on and off by a binary constant $B$. Using this method, the interface boundary conditions need not be considered, i.e., there is no need to select the unknown empirical constant $\beta$ a priori. The similarity solutions presented in the previous chapters were found to agree very well with the current one-domain solutions for low suction conditions, i.e. $Re < 12$, where only the type-I similarity solution was validated. We estimated $\beta$ through a curve-fitting approach. The results indicate that $\beta$ is a function of $Re$ and $Da$ and was found to be $O(1)$.

There are limitations with this approach. It must be noted that, when $12 < Re < 16$, it seems that the entrance effect, i.e., the developing length of the flow, increased and the flow did not quite develop to a similarity solution. With $Re > 16$, the algorithm became unstable in regions where we would expect type-II and type-III similarity solutions.

\textsuperscript{3}The derivation of the pressure drop $\Delta P_x$, $\Delta P_y$ for the similarity approach is shown in Appendix §A.2.
Figure 5.1: The flow fields as estimated by the two different numerical approaches, here $\Delta P_x$ is evaluated at the interface $y = 0.8$ and $\Delta P_y$ is evaluated at $x = 7.0$. The porous medium extends from $y = 0.8$ to $y = 1$, with $Da = 1 \times 10^{-3}$, $Re = 5$ and $\epsilon = 0.9$. In the lower panel of the figure, the fully-developed profile using the one-domain approach at $x = 7.0$ (shown as a dashed line) is compared to the similarity solution (shown as a solid line) with $\beta = 0.2$. 
Figure 5.2: The flow field determined as a function of $\beta$ using the similarity approach. The normalized $x$-component of velocity ($f_y$ for fluid region and $g_y$ for porous region) is given in the figure on the left and the $y$-component ($f$ for fluid region and $g$ for porous region) is given on the right. The porous medium (shown as the shaded area) extends from $y = 0.8$ to $y = 1$. This simulation was performed with $Da = 1 \times 10^{-3}$, $Re = 5$, and $\epsilon = 0.9$. 
Figure 5.3: The factors affecting the empirical constant $\beta$ in the shear stress jump condition (the tested cases are for $\chi = 0.9$, $\epsilon = 0.9$).
Finally, it should be noted that one may alternatively compute $\beta$ solely using the one-domain approach. The slope of the velocity profile near the interface within the fluid region and the porous region could be computed using the velocity data in the computing cells just above and below the interface. The empirical constant $\beta$ could then be obtained from the difference of the computed slope according to the interface boundary condition Eq. (3.12).
Chapter 6

EXPERIMENTAL WORK

6.1 Introduction

The purpose of the experimental work is to confirm our numerical predictions. In particular we are interested in measuring regions with stable type-I, II or III solutions and regions with multiple steady states. This chapter begins with a description of the experimental apparatus, the methodology of the measurement and the experimental procedure. Typical test results are presented, and then compared to our predictions.

6.2 Experimental Apparatus

The experimental channel consists of parallel plexiglass plates separated by a gap of 1.20 cm (see Fig. 6.1 and 6.2). The length and width of the channel are both 12.70 cm. The top plate is impermeable. A large number of holes have been drilled into the lower plate (see Fig. F.6). The open area of the lower plate has been measured to be 70%. Upstream of the channel is a sealed tank which is fed by an elevated reservoir, i.e. the head tank. The downstream tank is open to atmosphere and each tank has a baffle. Detailed drawings are given in Appendix F. In these experiments, loosely woven fabrics, i.e. the porous material, was secured to the bottom of the channel. Two different fabrics (AstenJohnson Inc. (http://www.astenjohnson.com)) were used in this study and their thicknesses and permeabilities are given in the Table 6.1 below. A glycerine-water solution

54
Table 6.1: The porous fabrics used in the experiment

<table>
<thead>
<tr>
<th>Fabric</th>
<th>Thickness (mm)</th>
<th>Porosity(^1)</th>
<th>Permeability (cfm/ft(^2))(^1)</th>
<th>Da</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.92</td>
<td>0.38</td>
<td>500</td>
<td>2.66 \times 10^{-3}</td>
</tr>
<tr>
<td>B</td>
<td>1.01</td>
<td>0.41</td>
<td>305</td>
<td>1.90 \times 10^{-3}</td>
</tr>
</tbody>
</table>

\(^1\) porosity was determined by measuring the displaced volume of water in a graduated cylinder.

\(^1\) permeability was estimated by Darcy's law using the air permeability data in cubic feet per minute per square feet of mesh, at a 0.5 in. water pressure drop, and 23°C measured on a Frazier permeometer.

with a kinematic viscosity of 0.37 cm\(^2\)/s was used in this apparatus. The use of glycerine in the solution ensures a laminar flow in the channel. With this solution, the Reynolds number in the channel, i.e., based upon the channel velocity, is less than 2000 for all experimental conditions tested. The viscosity of the solution was measured using a HAAKE Rotary Viscometer. The suction flow rate was measured using a bucket and stopwatch method. The maximum suction flow velocity was 74 mm/s.
Figure 6.1: the schematic of the experimental setup
Figure 6.2: the assembled test section in (a) exploded view and (b) assembled view
6.3 Pulsed Ultrasound Doppler Anemometry

Particle Image Velocimetry (PIV) and Laser Doppler Velocimetry (LDV) are the most common methods used to obtain experimental velocity and flow field measurements due to their ability to yield non-invasive real-time results. However, these two techniques are expensive, difficult to set up and not suitable for opaque systems. In comparison, a simple, non-intrusive measurement technique, an acoustic method, was found to be more advantageous for our situation. The commercialized unit we used is called the Pulsed Ultrasound Doppler Anemometer (PUDA) (product name: DOP2000) obtained from Signal Processing SA (Lausanne, Switzerland, http://www.signal-processing.com) and it has been applied previously in a number of studies [119, 120, 121, 122].

6.3.1 Measuring Principle

In pulsed Doppler ultrasound, a short ultrasonic burst is sent to the fluid periodically and a receiver continuously collects echoes issuing from targets (e.g., neutrally buoyant particles) that may be present in the path of the ultrasonic beam (see Fig. 6.3). From knowledge of the time delay \( t_d \) between an emitted burst and the echo issuing from the particle, the depth of the particle can be computed as \( P_d = \frac{ct_d}{2} \), where \( c \) is the sound velocity of the ultrasonic wave in the liquid. If the particle is moving at an angle \( \theta \) in relation to the axis of the ultrasonic beam, its velocity can be measured by computing the variation of its depth between two emissions separated in time by \( t_{prf} \):

\[
(P_2 - P_1) = v t_{prf} \cos \theta = \frac{c}{2}(t_2 - t_1) \tag{6.1}
\]

Since the time difference \( (t_2 - t_1) \) is always very short, i.e. less than a microsecond, it is easier to measure it with the phase shift of the received echo \( \delta \), where \( t_2 - t_1 = \delta / (2 \pi f_e) \) and \( f_e \) is the emitting frequency. With this information the velocity of the target is expressed by:

\[
v = \frac{c \delta}{2f_e \cos \theta t_{prf}} = \frac{c f_d}{2f_e \cos \theta} \tag{6.2}
\]

where \( f_d \) corresponds to the frequency shift of the received echo.
CHAPTER 6. EXPERIMENTAL WORK

This technique requires particles present in the fluid to issue echoes. Thus for a fluid without sufficient particles, small particles need to be seeded into the fluid. The ideal particles should be small to minimize gravity effects. As well, since the information from the ultrasound burst is available only periodically, there are limitations of the velocity and the depth that can be measured. The maximum velocity that exists for each pulse repetition frequency (PRF) is

$$v_{\text{max}} = \frac{c}{4f_c \cos \theta t_{\text{prf}}}$$
(6.3)

and the maximum depth is

$$P_{\text{max}} = \frac{t_{\text{prf}} c}{2}.$$  
(6.4)

For this technique, there are a number of parameters that need to be adjusted before a reliable profile is obtained.

6.3.2 Adjustment of Parameters

An example readout is shown in Fig. 6.4. The profile on the right is the echo profile. The two strong echoes at about 4 and 16 millimeters indicate the top wall position and the interface between the porous fabric and the fluid in the channel.\(^1\) The profile on the left,

\(^1\)Note that the wall thickness adjacent to the ultrasound probe is about 4 millimeters and the velocity of flow in the porous region cannot be measured using PUDA.
on the other hand, shows the velocity profile of the fluid flow in the channel. It is clear that the two strong echoes correspond roughly to the zero velocities. The display information on the left of the velocity profile records the number of the measuring profile and the time this profile was recorded. When the moving average filter is on, the average of a total of 128 profiles is displayed.

Typically, before a test, depending on the application, there are many parameters that need to be set up before reliable measurements can be obtained. Knowledge of the flow system helps to make this setup quicker. As stated, the limitations of the PUDA method require that for a shallow channel with relatively low velocities, a high emitting frequency is required. To make the result suited for high flow rate measurement, a low incident angle (or high Doppler angle) of the probe should be used. However too high a Doppler angle results in signal diffusion. Resolution is usually not an issue (0.203 mm) as the number of measuring points (gates) can be chosen to fit the depth of the channel. There are other parameters, e.g., the emitting power, the speed of sound in the fluid and the Doppler angle. The Doppler angle is a measure of the incident angle and the refractive index of the materials of the wall and the fluid. The Doppler angle of our application was determined to be 81° and the sound speed in the glycerine/water mixture is 1621 m/s.
Figure 6.4: The interface of the pulsed ultrasound Doppler anemometer Dop2000 from Signal Processing SA
6.4 Experimental Procedure

Before the start up of any experiment, the reservoir is filled with the water-glycerine solution containing a small amount of fine (~60 μm) polymeric seed particles (see Appendix §F.3 for a detailed description). Once the fluid is mixed thoroughly, the pump is started and the fluid is delivered to the head tank, measurements may commence.

When the suction rate is high, tiny air bubbles accumulate on the top wall, which lowers the visibility and causes acoustic impedance. This can be readily resolved by suddenly opening the outflow whereby the tiny bubbles are flushed away. The system usually needs about five minutes to stabilize and can be run continuously for two to three hours before the temperature of the fluid (about 15°C) is heated up by two degrees. Once the system is stable, the echo profile should indicate two strong echoes at positions near the four millimeter and fifteen millimeter marks as these correspond to the upper wall position and the position of the interface between the fluid and porous fabrics. The strength of the echo profiles are adjusted by controlling the time gain (in the PUDA software panel) such that there is no saturation at any channel depth and there is still enough echo everywhere in the channel. To avoid signal aliasing, the velocity profiles are monitored to be within the maximum velocity by choosing the right velocity mode and correct pulsed repetition frequency (PRF).

The range of the depth of the channel and the velocity of the flow makes the 10 MHz emitting frequency an ideal one; $t_{PRF}$ is typically 64 to 100 μs, and the number of gates is less than 100. The 10 MHz ultrasound probe, attached on a specially made stand-alone protractor, is positioned on the top wall at a 30 degree angle of incidence to the top wall surface, where the region of contact between the probe and the wall surface is sealed with ultrasound gel from the surrounding air. From our one-domain numerical predictions, it has been shown that the similarity of the velocity should have been developed when $x > 4$. The position of the Doppler probe is positioned away from the entrance region, at about $x = 8$ (here $x$ is dimensionless). Once a stable velocity profile is displayed, the profile is

\[2\text{The depth of the fluid region measured in the direction of the Doppler angle is slightly higher than the channel height by } 1.1/\sin(81°) - 1.1 = 0.014 \text{ cm.}\]
Figure 6.5: The velocity measured off the probe is the component velocity of particles moving in the ultrasound beam direction. Here $v$ denotes the flow velocity and $v_p$ denotes the velocity of flow in the ultrasound beam direction. It shall be noted that $\theta$ denotes the Doppler angle, where the refraction of the sound beam due to the sound speed difference in plexiglass and fluid has been accounted for.

recorded in binary and ASCII format files. Note that usually about 10 to 20 profiles are recorded for a given flow condition. Thus, the average and the standard deviation of the velocity data are computed and subsequently nondimensionalized with the average velocity in the direction of the ultrasound beam in order to allow comparison with the similarity numerical predictions. We chose to compare the measurements to the similarity solution since the algorithm is robust over a wide range of $Re$.

6.5 Data Evaluation

When trying to compare the results from the PUDA measurements with the numerical solutions, some complications occur. In the case when there is no suction, i.e., $Re = 0$, the flow field is one-dimensional and the x-direction velocity component $v_x$ is given by $v_x = v_p / \cos \theta$, where $v_p$ is the measured velocity in the direction of the ultrasound beam (non-dimensionalized by the suction velocity $U_w$). This is shown in Fig. 6.5. On the other hand,
when $Re \neq 0$, the flow field is two-dimensional and it is difficult to compare accurately the experimental result with the numerical solutions for pure $x$ or $y$ component velocities. A better alternative, is to use the numerical solutions to compute the velocity component in the direction of the ultrasound beam. With this, the depth that the ultrasound beam travels in the channel is apparently $H/\sin \theta$. Near the impermeable wall, the streamlines will be nearly parallel to the wall. Near the porous wall, however, the streamlines are curved. In this thesis, the numerical estimates will be projected onto the ultrasound beam direction over which the velocities are measured. The projection is given by

$$v_p = v_x \cos \theta + v_y \sin \theta,$$

see Fig. 6.6, where $\theta$ is the Doppler angle, $v_x = (\bar{V} - x)f_y$ and $v_y = f$. Upon examination of this projection we see that at $y = 0$ the measured velocity should be zero as $v_x = v_y = 0$, while at $y = 1$ the velocity should be finite and given by $v_p = v_y \sin \theta$. Here $\bar{V}$ is taken to be the ratio of the average velocity $\bar{V}^*$ and the mean suction velocity $U_w$ at a position where we define $x = 0$. The numerical velocity $v_p$ is normalized with its mean velocity in this direction, which is subsequently compared with the PUDA velocity data.\(^3\)

### 6.6 Results and Discussion

A small number of experiments were conducted to verify two aspects of the similarity numerical predictions, namely (i) the transition from stable type-I to type-II flow \(^4\) with increasing $Re$ and (ii) the existence of multiple steady solutions. In total 8 experiments were conducted and a summary of the experiments and predictions is shown in Table 6.2.

The experimental results are shown in Figs. 6.7 - 6.14, the numerical result\(^5\) is shown as solid line and the mean velocity data from PUDA are shown as dotted line. The standard

\(^3\)The PUDA velocity data are also normalized with its mean velocity.

\(^4\)Type-III solution indicates backflow in the porous region. However, since the PUDA method can not be used to measure the flow in the porous medium, the type-III flow solutions cannot be verified in this work.

\(^5\)It should be noted that $\beta$ used in each case of the numerical experiments is mostly interpolated or extrapolated from the best $\beta$ fit in Fig. 5.3. For Experiment 4, the $Re$ number is far from the region shown in this figure, therefore we assign $\beta = 1.5$ for this case. Ochoa-Tapia and Whitaker [81] suggested...
deviations of the experimental observations are also shown in these figures. Clearly, in general there is good agreement between the numerical predictions based upon the similarity method and the experimental measurements. In particular, excellent agreement was

\[ \beta \text{ should be of order 1 and they found } -1 \leq \beta \leq 1.47 \text{ in a set of experiments conducted by Beavers and Joseph [87, 88].} \]
CHAPTER 6. EXPERIMENTAL WORK

found when there is no suction or when suction $Re$ is small (i.e., type-I solutions). Both results predict a shift of the maximum velocity from the mid-plane of the channel towards the porous medium as the suction $Re$ increases. Included in these figures are the predicted velocity profiles in the porous medium. Our experimental method is incapable of measuring these profiles. We included this as it is still interesting to see that increased suction changes the curvature of the velocity profiles near the interface.

In one case, Experiment 4, we confirmed our predictions regarding backflow in the channel. Clearly the fluctuations in the flow, as given by the error bars in this figure, were found to be larger for type-II than type-I flows. Similarly in Experiment 8 we observed one stable solution and then after about 30 seconds the flow changed to another profile. Under this suction condition, the velocity profile underwent slow oscillations between two different backflow profiles, one with a small backflow and the other with a stronger backflow. Qualitatively, it seems that the two bifurcated numerical solutions agree with the PUDA results. In general the agreement is not as good with type-II flow solutions as with type-I solutions. Experimentally, stable backflows can be observed only when the fluid flow in the apparatus system is started up gently and kept stable under a combination of low inflow and low suction conditions, i.e., $\bar{v}Re$ is small. With this, the disturbances from the surrounding environment, e.g., pump vibrations, upstream flow disturbances, etc., are minimized. Even when backflows are found, they seem to be very sensitive to disturbances. When the suction flow is high, the risk of introducing more disturbance is possible. This is evident in Experiment 4, for which the standard deviations are much higher than in any other cases. The relatively “unstable” backflow-type solutions can be understood from the theory of the instability of the boundary layer, e.g., the flat plate flow. In essence, backflow indicates that there is a point of inflection of the velocity profile in the flow, with an excessive adverse gradient, and the main flow separates from the wall. The backflow-type profile is vulnerable to transitions to instability or turbulence. Furthermore, it seems that the flow is more or less time-dependent; thus a numerical solution of the time-dependent flow might better approximate the experimental results. In this case, the solution is probably of a limit cycle form, i.e., a spatially non-uniform solution that is periodic in time, similar to that suggested by King and Cox [4].
Regardless of the agreement, it should be noted that there are potential errors in the determination of $Re$ values, since the channel height, the viscosity and the density of the fluid, as well as the mean suction velocity, all have inherent error. An error analysis showed that the accuracy of $Re$ based on the uncertainty of viscosity, channel height, suction velocity and density of the fluid is approximately $\pm 5 - 11\%$ of $Re$. In addition to this, it is experimentally difficult to measure $\bar{V}$ at $x = 0$, and it is hard to estimate how uniform the suction flow $U_w$ is, and how it affects the similarity of the flow. Moreover, there are local variations of the channel heights $h$ and $H$. We may have overestimated the porosity $\epsilon$ with our measuring method, since at the edges of the fabrics, the pore openings are larger. As well, the permeability of the porous fabric was obtained from the product supplier, and may not be accurate. Measuring a Darcy permeability is possible, but may not fit our application, since the flow that pastes through the thin porous medium may well exceed a pore-scale Reynolds number of order 1 [77]. Finally, the wall effects in the PUDA measurement have sometimes made measuring flow velocity near the wall difficult, although by varying the angle of the probe with respect to the wall plane, it is possible to minimize this effect. Furthermore, because of the wall effects, and the number of the gates (or resolution) in measuring the velocity profile, the position of the wall or the surface of the porous fabric determined with the echo profiles generally correspond to the channel depth $h / \sin \theta$, but uncertainties exist and we estimated them to be within $\pm 1.5$ mm. Nevertheless, the measurements indicate the instability caused by the onset of the reverse-type flows.
Figure 6.7: The similarity numerical solutions shown as solid line (here $\beta = 0.13$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 1 in Table 6.2.
Figure 6.8: The similarity numerical solutions shown as solid line (here $\beta = 0.3$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 2 in Table 6.2.
Figure 6.9: The similarity numerical solutions shown as solid line (here $\beta = 0.9$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 3 in Table 6.2.
Figure 6.10: The similarity numerical solutions shown as solid line (here $\beta = 1.5$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 4 in Table 6.2.
Figure 6.11: The similarity numerical solutions shown as solid line (here $\beta = 0.15$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 5 in Table 6.2.
Figure 6.12: The similarity numerical solutions shown as solid line (here $\beta = 0.58$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 6 in Table 6.2.
Figure 6.13: The similarity numerical solutions shown as solid line (here $\beta = 1.1$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for Experiment No. 7 in Table 6.2.
Figure 6.14: The similarity numerical solutions shown as solid line (here $\beta = 1.1$) for the velocity profile in the direction of the ultrasound beam compared with the experimental results from PUDA (dotted line). The error bars denote one standard deviation from the mean velocity values. The simulation is for the two steady state solutions for Experiment No. 8 in Table 6.2.
Chapter 7

SUMMARY AND CONCLUSIONS

Twin wire forming is a papermaking process in which a fibre suspension is drained between two permeable wires to form a continuous paper web. This process (somewhat) resembles cross-flow filtration. The purpose of this study is to estimate the shear at this interface.

In this thesis we focused our efforts to understand the coupled two-dimensional flow of a Newtonian fluid both above and through a porous medium. In the fluid-only region, the two-dimensional flow field was governed by the Navier-Stokes equation. We consider the Brinkman-extended Darcy law relationship in the porous medium. Inertial terms are retained in the formulation and the interface conditions between the two domains are those given by Ochoa-Tapia and Whitaker [81]. It should be noted that these interface conditions are formulated with an empirical constant $\beta$ that is unknown a priori. The model equations were solved using two independent methods. In the first method we pose a similarity variable and reduce the governing equations to two, coupled, non-linear ordinary differential equations. Depending on the suction Reynolds and Darcy numbers, three types of solutions were found, i.e. no backflow, backflow near the solid wall, and backflows near the solid wall and in the porous medium. Typically, backflow-type solutions exist at large $Re$. In addition, the solution bifurcates and multiple steady states were found. The characteristics of the non-unique solutions were studied via a linear stability analysis, i.e., the method of normal modes.

In the next section of the work, the governing equations were re-posed as a one-domain problem, using the procedure outlined by Basu and Khalili [100] and solved in primitive
variable form, using a finite volume formulation. Finally, experiments were conducted to measure the velocity profiles and generally good agreement with the numerical predictions was found.
Chapter 8

RECOMMENDATIONS

The numerical methods presented in Chapter 4 are limited to $Re < 170$. This limits the usefulness of the solution. Attempts were made to perform a singular perturbation expansion to analyze the case when $Re \to +/ - \infty$. However, this problem proved to be too difficult and we were unsuccessful. In addition, we explored an extension of the similarity transform for the time dependent case. This posed a problem for the numerical algorithm used. It would be interesting to explore a spectral method to tackle this problem.

Finally, to make this thesis more relevant to the papermaking industry, a particulate phase must be included. At this point we feel that it may be impossible to use the similarity ansatz to solve the flow field when the fluid is a fibre suspension. However, this method may be useful to understand the motion of single particles or even the particle deposition mechanism during papermaking.
Bibliography


Appendix A

DERIVATION OF THE PRESSURE BOUNDARY CONDITION

A.1 Normal Stress

As described in the fluid mechanics textbooks, a dimensional Newtonian normal stress at the interface between the fluid $r_*$ and porous regions $r_*$ are respectively

$$
\tau_- = -P_- + 2\mu \frac{\partial u_y^*}{\partial y^*}
$$

$$
\tau_+ = -P_+ + 2\mu \frac{\partial u_y^*}{\partial y^*}
$$

(A.1)

where $u_y^*$, $v_y^*$, $u_y^*$ and $u_y^*$ denote the $y$ component velocities in fluid and porous regions respectively. When $r_*$ and $P_*$ are scaled with $\rho U_w^2$, we obtained the dimensionless conditions as following:

$$
\tau_+ = -P_+ + \frac{2}{Re} \frac{\partial u_y}{\partial y}
$$

$$
\tau_- = -P_- + \frac{2}{Re} \frac{\partial u_y}{\partial y}
$$

(A.2)

From Eq. (4.4), the above equations reduce to

$$
\tau_+ = -P_+ + \frac{2}{Re} \frac{\partial \hat{y}}{\partial y}
$$

$$
\tau_- = -P_- + \frac{2}{Re} \frac{\partial \hat{y}}{\partial y}
$$

(A.3)
APPENDIX A. DERIVATION OF THE PRESSURE BOUNDARY CONDITION

In the following we will derive Eq. (4.9) from the normal stress matching condition $\tau_+ = \tau_-$. To begin with, we observe that since $\rho_y(\chi, t) f_y(\chi, t)$ (from Eq. (4.8)), Eq. $\tau_+ = \tau_-$ reduces to $P_+ = P_-.

A.1.1 Fluid Region

The equations of motion for each velocity component of fluid region yield

$$\frac{\partial P_-}{\partial x} = -\frac{\partial v_x}{\partial t} - v_x \frac{\partial v_x}{\partial x} + v_y \frac{\partial v_x}{\partial y} + \frac{1}{2} \left( \frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} \right)$$

$$= (\bar{V} - x)(-\dot{f}_y + \frac{1}{Re} \dot{f}_{yy} - \ddot{f} f_y + f_y^2) \quad (A.4)$$

$$\frac{\partial P_-}{\partial y} = -\frac{\partial v_y}{\partial t} - v_x \frac{\partial v_y}{\partial x} + v_y \frac{\partial v_y}{\partial y} + \frac{1}{2} \left( \frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} \right)$$

$$= -\ddot{f}_t + \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y \quad (A.5)$$

By definition

$$\int dP_- = \int \frac{\partial P_-}{\partial x} dx + \int \frac{\partial P_-}{\partial y} dy$$

$$= (-\dot{f}_y + \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y + f_y^2) \int (\bar{V} - x) dx + \int \left( \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y - \dot{f}_t \right) dy \quad (A.6)$$

Upon integration of $dP_-$,

$$P_- = (-\dot{f}_y + \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y + f_y^2)(\bar{V} x - \frac{1}{2} x^2) + \frac{\dot{f}_y}{Re} - \frac{1}{2} \ddot{f}_t - \int \dot{f}_t dy + C_0 \quad (A.7)$$

where $C_0$ is the constant of integration. Assume the pressure at $x = 0$ and interface $y = \chi$ is $P_0$, we obtained that

$$P_0 = (-\dot{f}_y + \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y + f_y^2)0 + \frac{\dot{f}_y}{Re} - \frac{1}{2} \ddot{f}_t - \int \dot{f}_t dy + C_0 \quad (A.8)$$

where $\dot{f}$ is evaluated at interface $y = \chi$, thus

$$\dot{C}_0 = P_0 - \frac{\dot{f}_y(\chi)}{Re} + \frac{1}{2} \ddot{f}_t(\chi) + \int \dot{f}_t(\chi) dy \quad (A.9)$$

Hence,

$$P_- - P_0 = (-\dot{f}_y + \frac{\dot{f}_{yy}}{Re} f_y - \ddot{f} f_y + f_y^2)(\bar{V} x - \frac{1}{2} x^2) + \frac{\dot{f}_y - \dot{f}_y(\chi)}{Re}$$

$$- \frac{1}{2} \left( \ddot{f}_t - \ddot{f}_t(\chi) \right) - \int \left( \dot{f}_t - \dot{f}_t(\chi) \right) dy \quad (A.10)$$
A.1.2 Porous Region

Similarly, for porous region,

\[
\frac{\partial P_+}{\partial x} = -\frac{1}{\varepsilon} \left( u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} \right) + \frac{1}{\varepsilon Re} \left( \frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} \right) - \frac{1}{DaRe} u_x
\]

\[
= (\bar{V} - x) \left( -\frac{1}{\varepsilon} \frac{\partial \hat{g}_y}{\partial x} + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} + \frac{\hat{g}_y}{DaRe} \right) \tag{A.11}
\]

\[
\frac{\partial P_+}{\partial y} = -\frac{1}{\varepsilon} \hat{g}_t + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} - \frac{\hat{g}_y}{DaRe} \tag{A.12}
\]

Integration of \( dp_+ \) gives

\[
\int_{P_0}^{P_+} dP_+ = \left( -\frac{1}{\varepsilon} \frac{\partial \hat{g}_y}{\partial x} + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} + \frac{\hat{g}_y}{DaRe} \right) \int_x^y (\bar{V} - x) \, dx 
\]

\[
+ \int_x^y \left( -\frac{1}{\varepsilon} \hat{g}_t + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} - \frac{\hat{g}_y}{DaRe} \right) \, dy \tag{A.13}
\]

Hence,

\[
P_+ - P_0 = \left( -\frac{1}{\varepsilon} \frac{\partial \hat{g}_y}{\partial x} + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} + \frac{\hat{g}_y}{DaRe} \right) (\bar{V}x - \frac{1}{2}x^2) 
\]

\[
+ \frac{\hat{g}_y - \hat{g}_y(\chi)}{\varepsilon Re} - \frac{\hat{g}_y^2 - \hat{g}_y^2(\chi)}{2\varepsilon^2} - \frac{1}{DaRe} \int_x^y (\hat{g} + \frac{1}{\varepsilon} \hat{g}_t) \, dy \tag{A.14}
\]

Equate Eq. (A.10) and Eq. (A.14) at \( y = \chi \), i.e., \( P_+ = P_- \) yields

\[
-\hat{f}_y + \frac{\hat{f}_{yy}}{Re} = -\frac{1}{\varepsilon} \frac{\partial \hat{g}_y}{\partial x} + \frac{\hat{g}_{yy}}{\varepsilon Re} - \frac{\hat{g}_{yy}^2}{\varepsilon^2} + \frac{\hat{g}_y}{DaRe} \tag{A.15}
\]

Equation (A.15) is identical to Eq. (4.9). It is to be noted that this condition is in effect equivalent to the condition of pressure gradient matching in \( x \) direction.

A.2 Pressure Drop

From the expressions derived in Eqs. (A.10) and (A.14), the steady case pressure drop in \( x \) and \( y \) directions, \( \Delta P_x \) at the interface \( y = \chi \) and \( \Delta P_y \) at a position \( x \) are respectively:

- for fluid region

\[
\Delta P_x = P(x, y) - P(0, y) = \left( \frac{1}{Re} \hat{f}_{yy} - \hat{f} f_y + \hat{f}_y^2 \right) (\bar{V}x - \frac{x^2}{2}) 
\]

\[
= A_1 (\bar{V}x - \frac{x^2}{2}) \tag{A.16}
\]
and

$$\Delta P_y = P(x, y) - P(x, 0) = \frac{1}{Re} f_y - \frac{f^2}{2} \quad (A.17)$$

• for porous region

$$\Delta P_x = P(x, y) - P(0, y) = \left( \frac{1}{Re} g_{yyy} - \frac{1}{DaRe} g_y + \frac{1}{\epsilon^2} g_y^2 - \frac{1}{\epsilon^2} g_{yy} \right) (\bar{V} x - \frac{x^2}{2})$$

$$= \frac{A_2}{\epsilon} (\bar{V} x - \frac{x^2}{2}) \quad (A.18)$$

and

$$\Delta P_y = P(x, y) - P(x, 0) = \frac{1}{Re} g_y - \frac{1}{DaRe} \int_x^y g - \frac{g^2}{2\epsilon^2} \quad (A.19)$$

where $A_1$ and $A_2$ are the previous-mentioned integration constants of the steady equations of Eqs. (4.5) and (4.6) in the numerical procedure. It could be easily identified that $A_1 = A_2/\epsilon$. It has to be noted that the pressure drop $\Delta P_x$ computed from the similarity solution is added a reference pressure in order to compare to the pressure drop from the one-domain solution. Similarly a reference pressure is added to the pressure drop $\Delta P_y$ of the similarity solutions at position $x$. 
Appendix B

THE ASYMPTOTIC SOLUTIONS

The leading order and first order asymptotic approximation functions for \( f_0 \), \( f_1 \) and \( g_0 \), \( g_1 \) with \( Da = 1 \times 10^{-3}, \xi = 0.8, \epsilon = 0.9 \) and \( \beta = 0.2 \) are:

\[
\begin{align*}
    f_0(y) &= -3.5171y^3 + 4.3602y^2 \\
    f_1(y) &= -0.1767y^7 + 0.5112y^6 - 0.6337y^5 + 0.5535y^3 - 0.2700y^2 \\
    g_0(y) &= 0.9796 + 0.0211y - 6.7389 \times 10^{-17}e^{30y} - 1.7872 \times 10^8 e^{-30y} \\
    g_1(y) &= 1.5513 \times 10^{-7}e^{30y-30} - 5.8370 \times 10^{-32}e^{-30y+90} \\
    &+ 3.6291 \times 10^{-19}e^{-30y+60} + 8.4463 \times 10^{-6}e^{-30y+30} \\
    &- 3.4234 \times 10^{-30}ye^{30y+30} - 3.6965 \times 10^{-32}ye^{30y+30} \\
    &+ 6.1728 \times 10^{-17}y^2 - 9.1245 \times 10^{-6}ye^{-30y+30} \\
    &- 9.8032 \times 10^{-8}e^{-30y+30}y^2 - 0.003321y + 3.7519 \times 10^{-30}e^{30y+30} \\
    &+ 8.1680 \times 10^{-18}e^{30y} + 0.003203
\end{align*}
\]

The solutions are compared with the numerical solutions as shown in Fig. B.1, good agreement was found in \(-4.5 < Re < 4.5\).
Figure B.1: The shear stress function $-f_{yy}(\chi)$ at the interface for different $Re$, it compares the similarity numerical solution (o) with the asymptotic solution (-). The simulation is for the case $Da = 1 \times 10^{-3}$, $\chi = 0.8$, $\beta = 0.2$ and $\epsilon = 0.9$. 
Appendix C

CODE VALIDATION

We attempt to validate our numerical routine by examining conditions which should approximate the classical theory given by [3]. To do this, we set $Da = 1000$ and $\beta = 0$. When $Da$ is large, the sink term in Eq. (4.6) vanishes and the problem is reduced to Berman's classical treatment. The validation is shown in Fig. C.1, good agreement was achieved. The discrepancy is higher when $Re$ increases, in this case the sink term becomes increasingly important.
Figure C.1: The shear stress function $-g_{yy}(1)$ at the porous wall for different $Re$, it compares the similarity numerical solution (•) for $Da = 1000$, $\chi = 0.8$, $\beta = 0$ and $\epsilon = 1$ with Berman solution (-) [3, 4].
Appendix D

CODE FOR THE SIMILARITY SOLUTION

The code written in Matlab 6.5 for the similarity solution is included here. Two methods are presented. The first one is based on a regular shooting method in conjunction with Newton’s method and the second is based on our modification of Terill’s method [104], which we called “the reverse solver”.

D.1 Shooting Method

The main function files, are listed in the following.

- similarity_shooting.m
- eqfluid.m
- eqporous.m
- odesolve.m
- rkck.m
- gaussj2.m
Similarity Solution - Shooting Method

Last updated - May 29 2004

'Similarity_shooting.m' is the main script file to call the functions to calculate the similarity solutions using shooting and Newton's method.

Parameters (all dimensionless):
- Re: suction Reynolds number
- Da: Darcy number of the porous medium
- porosityf: the porosity of the fluid, by default is '1'
- porositym: the porosity of the porous medium, <1
- beta: empirical constant of the shear stress jump condition
- height: the height of the fluid region

Intermediate variables:
- xx1: the initial guess xx=[A1 f_yy(0)]
- A1: The integration constant for fluid equation, A1=f_yyy(0)
- A2: The integration constant for porous equation.
- Soltot: Final solution [y f f_y f_yy f_yyy] for fluid region.
- Soltot2: Final solution [y g g_y g_yy g_yyy] for porous region.

clc
clear % clear old workspace
global Re Da xx1 A2 Soltot Soltot2

The required solutions at various Re number are stored in 'Repa' vector
Repa=[13.2]; porosityf=1; Da=2.66e-3;
porositym=.9; iter=0; beta=0.2; valfac=1;
nn=length(Repa); A1pa=zeros(1,nn);
height=.85;

The grid number in fluid/porous.
Re=Repa(ii); A1=-1; gammaa=1;
% region is 50

% Allow Newton's maximum iteration times to be 1000
maxit=1000; tol=1e-9;

% Let xx1 vector store initial guess A1 and gammaa=gl'(0) and it is updated in the following inner 'while' loop by Newton's method,
xx1=[A1 gammaa]; maxdx=1e10;
while iter<maxit & maxdx>tol
    iter=iter+1; maxdx=0;

% Start of computing Jacobin matrix 'a' numerically------

[g0,nfun,hstart]=odesolve('eqfluid_shoot',0,height,[0 0 xx1(2)],...
    1e-6,height,1e-5*1,height);

z_1=porosityf*(xx1(1)*Re/porosityf-...
    Re/porosityf^2*(g0(2)^2-g0(1)*g0(3)));
gp0_2=porositym*(g0(3)+1/sqrt(Da)*g0(2)*beta);
gp0_3=porositym*Re*(z_1/Re+g0(2)/Da/Re*valfac-g0(1)*g0(3)+...
    g0(2)^2*(1-1/porositym^2)+g0(1)*gp0_2/porositym^2);
A2=gp0_3/Re+(g0(2)^2-g0(1)*gp0_2)/porositym-...
    porositym/Da/Re*gp0_2*valfac;

[gg0,nfun,hstart]=odesolve('eqporous_shoot',height,1,...
    [g0(1) g0(2) gp0_2],1e-6,1,1e-5*1,1-height);

xx1(1)=xx1(1)+1e-6*xx1(1);

[g1,nfun,hstart]=odesolve('eqfluid_shoot',0,height,[0 0 xx1(2)],...
    1e-6,height,1e-5*1,height);

z_10=porosityf*(xx1(1)*Re/porosityf-Re/porosityf^2*(g1(2)^2-...
    g1(1)*g1(3)));
gp1_2=porositym*(g1(3)+1/sqrt(Da)*g1(2)*beta);
gp1_3=porositym*Re*(z_10/Re+g1(2)/Da/Re*valfac-g1(1)*g1(3)+...
    g1(2)^2*(1-1/porositym^2)+g1(1)*gp1_2/porositym^2);
A2=gp1_3/Re+(g1(2)^2-g1(1)*gp1_2)/porositym-...
    porositym/Da/Re*gp1_2*valfac;

[gg1,nfun,hstart]=odesolve('eqporous_shoot',height,1,...
    [g1(1) g1(2) gp1_2],1e-6,1,1e-5*1,1-height);

pdeA1=((gg1(1)-l)-(gg0(1)-l))/(1e-6*xx1(1));
pdeA2=((gg1(2)-0)-(gg0(2)-0))/(1e-6*xx1(1));

xx1(1)=xx1(1)-1e-6*xx1(1);
xx1(2)=xx1(2)+1e-6*xx1(2);

[g2,nfun,hstart]=odesolve('eqfluid_shoot',0,height,[0 0 xx1(2)],...
    1e-6,height,1e-5*1,height);

z_12=porosityf*(xx1(1)*Re/porosityf-Re/porosityf^2*(g2(2)^2-...
    g2(1)*g2(3)));
gp2_2=porositym*(g2(3)+1/sqrt(Da)*g2(2)*beta);
gp2_3=porositym*Re*(z_12/Re+g2(2)/Da/Re*valfac-...
    g2(1)*g2(3)+g2(2)^2*(1-1/porositym^2)+...
    g2(1)*gp2_2/porositym^2);
A2=gp2_3/Re+(g2(2)^2-g2(1)*gp2_2)/porositym-...
    porositym/Da/Re*gp2_2*valfac;

[gg2,nfun,hstart]=odesolve('eqporous_shoot',height,1,...
    [g2(1) g2(2) gp2_2],1e-6,1,1e-5*1,1-height);
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

\[
pdegam1 = ((gg2(1)-1) - (gg0(1)-1))/(1e-6*x1(2));
\]
\[
pdegam2 = ((gg2(2)-0) - (gg0(2)-0))/(1e-6*x1(2));
\]
\[
x1(2) = x1(2) - 1e-6*x1(2);
\]
\[
a = [pdeA1, pdeA2, pdegam1, pdegam2; -gg0(1)+1, -gg0(2)+0];
\]

% End of computing Jacobin matrix 'a' -------

% Use Gauss-Jordon method 'gaussj2' to solve
% 'dx=[DeltaA1; Deltagamma]' (increment for solution update)
% The solution dx is of such form that it need to be transposed to
% meet the dimension agreement between xx vectors. xx is updated until
% dx is sufficiently small, i.e., less than the allowed tolerance.

\[
dx = gaussj2(a); dx = dx';
\]
\[
xx1 = xx1 + dx; maxdx = max(abs(dx));
\]
end

if maxdx <= tol | iter >= maxit
    \[
    Alpa1 = xx1(1);
    \]
else
    error ('Newtons method did not converge when Re is %4.0f', Re)
end

% By now, the right initial values xx1 is found, Below is a standard procedure
% to solve the initial value problem:

% Solve the initial value problem- in fluid region:
\[
a = 0; b = height; tol = 1e-6; n = 50;
\]
\[
nm = n - 1; dxx = (b - a)/nm; dtol = tol/nm;
\]
\[
hstart = dxx; hmin = 1e-3 * dxx; hmax = dxx;
\]
\[
x0(1) = a; y0(1,:) = [0 0 xx1(2)]; nfuntot = 0;
\]
for i = 1:nm
    \[
z0(i) = porosityf*(xx1(1)*Re/porosityf-Re/porosityf^2*(y0(i,2)^2-
    \[
y0(i,1)*y0(i,3));
    \]
    ip = i + 1;
    \[
x0(ip) = x0(i) + dxx;
\]
    \[
[y0(ip,:), nfuntot, hstart] = odesolve('eqfluid_shoot', x0(i), x0(ip), y0(i,:),...
    \[
dtol, hstart, hmin, hmax);
\]
    \[
z0(ip) = porosityf*(xx1(1)*Re/porosityf-Re/porosityf^2*(y0(ip,2)^2-
    \[
y0(ip,1)*y0(ip,3)); % g1''(eta)
    \]
nfuntot = nfuntot + nfuntot;
end
\[
Sol = [x0' y0 z0'];
\]
\[
Soltot = [Soltot Sol]; % store all solutions into 'Soltot',
\]
% note the first column is a 0 vector.
% Solve the initial value problem in porous region:
% a=height; b=1; tol=1e-6; n=50;
% nm=n-1; dxx=(b-a)/nm; dtol=tol/nm;
% hstart=dxx; hmin=1e-3*dxx; hmax=dxx;
% x(1)=a; y(1,:)=[g0(1) g0(2) gp0_2]; nfuntot=0;

for i=1:nm
    z(1)=porositym*(A2*Re/porositym-Re/porositym^2*(y(1,2)^2-...
        y(1,1)*y(1,3))+1/Da*y(1,2)*valfac);
    ip=i+1;
    x(ip)=x(i)+dxx;
    [ydp,:),nfuntot]=odesolve('eqporous_shoot',x(i),x(ip),y(i,:),nfun,...
        dtol,hstart,hmin,hmax);
    z(ip)=porositym*(A2*Re/porositym-Re/porositym^2*(ydp,2)^2-...
        ydp, 1)*y(ip,3)+1/Da*y(ip,2)*valfac) ;
    nfuntot=nfuntot+nfun;
end
Sol2=[x' y z'];
Soltot2=[Soltot2 Sol2]; % store all solutions into 'Soltot2',
% note the first column is a 0 vector.

% fder2=Soltot(1,5)*Re*Da^-1.5/porositym^1.5 % calculate the initial
% fder3=Soltot(1,6)*Re*Da^-2/porositym^2 % values in terms of...
% calculate the boundary layer thickness in the porous region:
alpha=sqrt(porositym/Da)
xi=interp1(Soltot2(:,4),Soltot2(:,2),0.01)
end

% End of Main Program %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% eqfluid_shoot.m

function z=eqfluid_shoot(x,g)

% 'eqfluid_shoot.m' is a function m-file that defines the ODE equation
% for the fluid region (the integrated form)
% xx1(1)=A1, A1 is the integration constant.

global Re xx1 Da

z(1)=g(2);
z(2)=g(3);
z(3)=xx1(1)*Re-Re*(g(2)^2-g(1)*g(3));

% End of eqfluid_shoot.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% eqporous_shoot.m

function z=eqporous_shoot(x,g)

% 'eqporous_shoot.m' is a function m-file that defines the ODE equation
% for the porous region (the integrated form)
% A2 is the integration constant.

global Re A2 Da

porosity=.9; valfac=1; valfac2=1;
z(1)=g(2);
z(2)=g(3);
z(3)=A2*Re-valfac2*Re/porosity*(g(2)^2-g(1)*g(3))+1/Da*g(2)*porosity*valfac;

% odesolve.m

function [yb,nfun,hstart]=odesolve(f,a,b,ya,tol,hstart,hmin,hmax)

% 'odesolve.m' is a function m-file that uses the "embedded" Runge-Kutta
% method to solve IVP of simultaneous first order differential equations
% of the form:
% dy1/dx=f1(x,y1,y2,...ym);
% dy2/dx=f2(x,y1,y2,...ym);
% .................
% dym/dx=fm(x,y1,y2,...ym);
% where IV conditions are:
% y1(a)=alpha1, y2(a)=alpha2,..., ym(a)=alpha(m);
% a < x < b.

% The user-prescribed accuracy of the solution is checked by adaptively
% adjusting the step-size of the solution subinterval to ensure that a
% global tolerance is met.

% Input arguments:
% f =dummy name of rhs functions
% a =start of solution interval x=a
% b =end of solution interval x=b
% ya =initial values of vector y[1:m] at x=a
% tol =global accuracy required at x=b
% hstart =initial step-size
% hmin =minimum step-size
% hmax =maximum step-size

% Output arguments:
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

% yb     = values of vector y[1:m] at x=b
% nfun   = number of rhs function evaluations needed
% hstart = last successful value of hnew before return

m=length(ya);

% Initialize variables, where xold is a scalar, yold is a vector
hold=hstart; xold=a; yold=ya; nfun=0; success=0;

% Loop until yb is determined to required tolerance
while success==0

    % Since ya is a vector with length m, number of function of evaluations
    % would be 6 times m everytime call to 'rkck.m'
    nfun=nfun+6*m;
    [ynew,ydiff]=rkck(f,xold,yold,hold);
    gamma=(tol*hold/(ydiff*(b-a)))^0.25;
    hnew=0.8*gamma*hold;

    % If gamma<1, adjust step size and loop to recalculate ynew
    if gamma<1
        if hnew<hold/10
            hnew=hold/10;
        end
        if hnew<hmin
            fprintf('
 odesolve failed at x=%8.3e \n\n,xold',xold)
            error
        end
        hold=hnew;
    % If gamma>=1, accept ynew and add hold to xold
    else
        if hnew>5*hold
            hnew=5*hold;
        end
        if hnew>hmax
            hnew=hmax;
        end
        if xold+hold<b
            xold=xold+hold; hold=hnew; yold=ynew;
        % If xold+hold>=b, calculate yb and return
    end
end
else
    success=1; hstart=hnew; hold=b-xold;
    nfun=nfun+6*m;
    [yb,ydiff]=rkck(f,xold,yold,hold);
end
end

%%%%%% End of odesolve.m %%%%%%%

% rkck.m

function [y2, ydiff]=rkck(f,x1,y1,h)

% 'rkck.m' is a function m-file that evaluates the "embedded" Runge-Kutta
% method, which is due to Cash and Karp (1990). The solution is advanced
% using the 5th-order formula, while the difference between the 4th and
% 5th-order methods is determined to help update the step-size in
% 'odesolve.m'.

% Input arguments:
% f = dummy name of the rhs functions of the equation sets to be solved.
% x1 = value of scalar x at start of solution interval
% y1 = values of vector y at start of solution interval
% h = solution interval step-size

% Output arguments:
% y2 = value of y at end of solution interval
% ydiff= rhs function evaluations needed

% k1, k2,...k6 are vectors with length the same as y1 and y2.

    kl=h*feval(f,x1,y1);
    k2=h*feval(f,x1+h/5,y1+k1/5);
    k3=h*feval(f,x1+3*h/10,y1+3*k1/40+9*k2/40);
    k4=h*feval(f,x1+3*h/5,y1+3*k1/10-9*k2/10+6*k3/5);
    k5=h*feval(f,x1+h,y1-ll*k1/54+5*k2/2-70*k3/27+35*k4/27);
    k6=h*feval(f,x1+7*h/8,y1+1631*k1/55296+175*k2/512+575*k3/13824+...
              44275*k4/110592+253*k5/4096);

    y2=37*kl/378+250*k3/621+125*k4/594+512*k6/1771+y1;
    ydiff=abs((37/378-2825/27648)*kl+(250/621-18575/48384)*k3+...
              (125/594-13525/55296)*k4-277*k5/14336+(512/1771-1/4)*k6);

% The value of scalar 'ydiff' is taken as the maximum of the differences
% between each element of vectors y2(5th order) and y2(4th order). It is
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

% important to give ydiff0 an algebraic form, otherwise, if just put
% ydiff=y2(5th)-y2(4th), the computer will give divided by zero warning
% at 'odesolve at line 48', which is assumed to be the roundoff errors.

ydiff=max(ydiff0);

%%%%%%% End of rkck.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% gaussj2.m

function x=gaussj2(A)

% 'gaussj2' uses the Gauss-Jordan method to solve simultaneous linear
% equations.
% The elimination loop is conducted for n times instead of n-1 and the
% if i=k makes elimination only conducted on elements who are not on
% the diagonal. Note that during the process the A(k,k) has
% been normalized to 1, therefore the final result is x=A(i,n+1).

n=size(A,1);

% partial pivot selection with scaling; the same as described
% in file\gaussj.m in the submited assignment.

for k=1:n
    pivot=k; maxaij=zeros(1,n); spivot=0;
    for i=k:n
        maxaij(i)=abs(A(i,k));
        for j=k:n
            if abs(A(i,j))>maxaij(i)
                maxaij(i)=abs(A(i,j));
                end
        end
        abssi=abs(A(i,k)/maxaij(i));
        if spivot<abssi
            spivot=abssi;
            pivot=i;
        end
    end
    if pivot~=k
        for j=k:n+1
            temp=A(pivot,j);
            A(pivot,j)=A(k,j);
            A(k,j)=temp;
        end
    end
end
% GJ elimination with normalization of $A(k,k)$.

    akk=A(k,k);
    for j=1:n+1
        $A(k,j)=A(k,j)/akk$;
    end
    for i=1:n
        if i\neq k
            aik=A(i,k);
            for j=1:n+1
                $A(i,j)=A(i,j)-aik*A(k,j)$;
            end
        end
    end

if $A(n,n)==0$
    error('zero pivot coefficient encountered in last row!')
end

% Final result

    x=A(1:n, n+1);

End of gaussj2.m


D.2 The Reverse Solver

The code is presented as the main script file following by the various supporting function files.

- stability_shootMain_auto.m
- solveatconstDaMain.m
- coeff.m
- Boundeqn.m
- solveDa2.m
- coxsolvefun.m
- brent.m
- eqfluid.m
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

- eqporous.m
- odesolve1.m
- odesolve2.m
- rkck1.m
- rkck2.m
- solvetoanyposition.m
- solvetoanyposition_p.m
- eqstability.m
- eqstability_p.m
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

% Similarity Solution - A Reverse Solver & Stability analysis
% Last updated - July 06 2004

% Main program file of the Reverse Solver to solve for steady solutions
% and Stability Analysis. Note that the eigenvalues obtained are real roots
% Parameters (all dimensionless):
% B1_r: the integration constant of the stability eqn in fluid region.
% lambda_r: the eigenvalue $\sigma$.
% Re1: the Re number solved
% Totfluid_cal: the intermediate solutions
% bposition1: the rescaling constant 'q', equals sqrt(epsilon/Da).
% targetDa: the targeted Da at which solution is solved.
% Dai: the actual solved Da, an intermediate value.
% f2der: the initial guess, equals to lambda_yy(0).
% f3der_ini: the initial guess, equals to psi_yyy(0)
% beta: the empirical shear stress jump constant
% inter: the dimensionless height of fluid region 'h'.
% Totfluid: final solution for fluid region.
% Totfluid2: final solution for porous region.

clear
clc
global B1_r lambda_r Re1 Totfluid_cal bposition1...
    targetDa Dai f2der porositym beta inter...
    Tot_Totfluidcal record_xx Tot_Totfluidcal_p record_xx_p

show_tot=[];
load ini_guess.dat % to load the previous solution as the initial guess
% to obtain solution for a slightly different Re.
% ini_guess contains the guess of lambda_zz(0), lambda_zzz(0), and the
% guess integration constant B1_r and eigenvalue lambda_r.
% the program is designed to be able to self-iterate to another $Re$.

size=length(ini_guess(:,1));
tol=1e-6; maxit=300;
beta=0.2; porositym=0.9; inter=0.8; targetDa=.0026;
guess=ini_guess(1,4:5);
for i=size
    Tot_Totfluidcal=[]; record_xx=[];Tot_Totfluidcal_p=[]; record_xx_p=[];
f3der_ini=ini_guess(i,2:3); f2der=ini_guess(i,1);
    [Totfluid, Totfluid2, resultv_l]=solveatconstDaMain...
        (targetDa, f2der, f3der_ini); % compute solution at targeted Da.
    iter=0; maxdx=1e10;
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

B1_r=guess(1); lambda_r=guess(2); % start compute eigenvalue of Stability
x=[lambda_r B1_r]; % equation using Newton’s method
while iter<maxit & maxdx>tol
    iter=iter+1;
    maxdx=0;
    a=coeff('Boundeqn',x);
    dx=gaussj2(a)
    x=x+dx';
    maxdx=max(abs(dx));
    Tot_Totfluidcal=[];
    record_xx=[];
    Tot_Totfluidcal_p=[];
    record_xx_p=[];
end

if maxdx<=tol
    fprintf('
 Successful solution in %3.0f iterations 

', iter)
else
    error('Newtons method did not converge')
end
show=[resultv_l' x(2) x(1) guess]'; guess=[x(2) x(1)]; % result output.
show_tot=[show_tot; show];
dlmwrite('show_tot.xls',show_tot,'t');
fprintf('%4.13f
',show_tot)
end

function [Totfluid, Totfluid2, resultv_l]=solveatconstDaMain(targetDa,...
    f2der, f3der_ini)
global Dal Rel porositym Totfluid Totfluid2 beta

% ‘solveatconstDaMain.m’ is the driver file for solveDa2.m
% this method assumes constant second derivative at z=0, but to find third
% derivative, and at the same time get the solutions at target Da numbers.

grid_nfluid=60; grid_nporous=40; % grid_n should be set at the same time in
% file solveDa2.
%
%f3rdder, irr]=brent('solveDa2',34.23405585, 34.27405585,1,1e-3,1e-6);
options = optimset('Tolx',1e-13);
f3rdder = fzero('solveDa2',[f3der_ini(1),f3der_ini(2)],options); % the above two root finding methods could be used alternatively.
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

% fzero function is a matlab built in function to find a single real root
stress_inter_l=-Totfluid(grid_nfluid,5); % the interface shear 'f_yy(chi)
stresswall_l=-Totfluid2(grid_nporous,5); % the shear stress 'f_yy(l)
resultv_l=[Dal f2der f3rdder Re1 stress_inter_l stresswall_l]'; % output

end of solveatconstDaMain.m

% coeff.m is the augmented Jacobian matrix

definition of coeff.m

function a=coeff(f,x)
global Bl_r lambda_r
n=length(x); np=n+1; delx=1e-4*x;
for i=1:n
    Bl_r=x(2); lambda_r=x(1);
    a(i,np)=-feval(f,i,x);
    for j=1:n
        xtemp=x(j);
        x(j)=x(j)+delx(j);
        Bl_r=x(2); lambda_r=x(1);
        ftemp=feval(f,i,x);
        x(j)=xtemp;
        a(i,j)=(ftemp+a(i,np))/delx(j);
    end
end

end of coeff.m

% Boundeqn.m

% equations to be evaluated by coeff.m and solved by Newton's method:
definition of Boundeqn.m

function y=Boundeqn(i,x)
global' Bl_r lambda_r Re1 Dal Totfluid_cal Totfluid_calmed B2 beta ...
porosiymp inter

% integration from y=0 to y=interface
[W_f,nfun,hstart]=odesolve1('eqstability',0,inter,[0 0 1],1e-6,inter,...
                          1e-6*inter,inter);
W_f_3der=Bl_r+Re1*lambda_r*W_f(2)+Re1*(W_f(3)*Totfluid_cal(2)-...
                          2*Totfluid_cal(3)*W_f(2)+Totfluid_cal(4)*W_f(1));
% Evaluate the boundary conditions at the interface.
W_p(1)=W_f(1); W_p(2)=W_f(2);
W_p(3)=porosiymp*(W_f(3)+beta/sqrt(Dal)*W_f(2));

W_p_3der=porosiymp*W_f_3der+(Re1*Totfluid_cal(2)-...
                          Totfluid_cal(2)*porosiymp*Re1*W_f(3)+(2*porosiymp*Re1*Totfluid_cal(3)+...
                          Re1*Totfluid_cal(2)*beta/sqrt(Dal)+porosiymp/Dal-...
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

2*\text{Totfluid}_\text{cal}(3)*\text{Re}_1/\text{porositym}*W_f(2)\ldots
+(-\text{porositym}*\text{Re}_1*\text{Totfluid}_\text{cal}(4)+\text{Re}_1*(\text{Totfluid}_\text{cal}(4)+\ldots
\beta/sqrt(Da_l)*\text{Totfluid}_\text{cal}(3))*W_f(1)-(\text{porositym}-1)*\text{Re}_1*W_f(2)*\lambda_r;
B2=W_p_3der-\text{porositym}/Da_l*W_p(2)-\text{Re}_1/\text{porositym}*(\text{Totfluid}_\text{cal}(2)*W_p(3)\ldots
-2*\text{Totfluid}_\text{cal}(3)*W_p(2)+*(\text{Totfluid}_\text{cal}(4)+\beta/sqrt(Da_l)*\ldots
\text{Totfluid}_\text{cal}(3)))*\text{porositym}+\text{W}_p(1)-\text{Re}_1*\lambda_r*\text{Totfluid}_\text{cal}(3);

% Integrate from the interface to the porous wall.
[W_p_\text{at}1,nfun,h\text{start}]=\text{odesolve2('eqstability}\_p',\text{inter},1,W_p,1e-6,(1-\text{inter}),\ldots
1e-6*(1-\text{inter}),1-\text{inter});

if i==1
y=(W_p_\text{at}1(1));
else
y=(W_p_\text{at}1(2));
end

% 'solveDa2.m' is the driver file to find the position at which \text{f}_y(0)=0
% by brent's root finding method. 'bposition1' can be predicted by the
% formula 'bposition1=sqrt(\text{porositym}/\text{target}_Da)' to help determine the
% efficient guess of the solutions. In this routine the detail solutions
% is computed too. Note to get detailed solution, set grid_nfluid,
% grid_nporous to be larger (41 is good enough), the solutions obtained
% will be at target Da upon success of the iterations.

function zzz=solveDa2(gamma)

global porositym phi_1 interface f2der A1 A2 IndexO phi targetDa...
Dal Re1 Totfluid Totfluid2 bposition1...

IndexO=0;
grid_nfluid=60; grid_nporous=40;
Soltot=zeros(grid_nfluid,1); Soltot2=zeros(grid_nporous,1);
Nosolnf=zeros(grid_nfluid,6); Nosolnp=zeros(grid_nporous,6);
Totfluid=zeros(grid_nfluid,0);
Totfluid2=zeros(grid_nporous,0);
Totfluidm=zeros(grid_nfluid,0);
Totfluidm2=zeros(grid_nporous,0);

A1=gamma;

options = optimset('Tolx',1e-6);
bposition1 = fzero('coxsolvefun',[7.746,7.74592]);
bposition1 = fzero('coxsolvefun',18.6052,150,1,0.0000002,1e-6)
if IndexO==1
Index0=0;
fprintf(’no root found 
\n f2der=\%8.4f f3der=\%8.4f \n’, fsder, A1)
Totfluid=[Totfluid Nosolnf];
Totfluid2=[Totfluid2 Nosolnp];
Totfluidm=[Totfluidm Nosolnf];
Totfluidm2=[Totfluidm2 Nosolnp];
else
 fsder=f2der;
 f3der=A1;
% solution contains Da and Re (intermediate values):
 Dal=porositym/bposition1^2;
 Rel=porositym^1.5/sqrt(Dal)*phi_1(1);
%
% compute detail solution lambda(z)
 a=0; b=interface; tol=1e-6; n=grid_nfluid;
 nm=n-1; dxx=(b-a)/nm; dtol=tol/nm;
 hstart=dxx; hmin=1e-6*dxx; hmax=dxx;
 x0(1)=a; y0(1,:)=[0 0 fsder]; nfuntot=0;
for i=1:nm
 z0(1)=A1-(y0(1,2)^2-y0(1,1)*y0(1,3));
 ip=i+1;
 x0(ip)=x0(i)+dxx;
 [y0(ip,:),nfuntot,hstart]=odesolve(’eqfluid’,x0(i),x0(ip),y0(i,:),... dtol,hstart,hmin,hmax);
 z0(ip)=A1-(y0(ip,2)^2-y0(ip,1)*y0(ip,3)) ; % g1’’’(eta)
 nfuntot=nfuntot+nfuntot;
 end
% rescale lambda(z) back to the desired form: f(y)
 x0=x0/sqrt(porositym/Dal);
 y0(:,1)=y0(:,1)*sqrt(porositym/Dal)/Rel;
 y0(:,2)=y0(:,2)*porositym/Dal/Rel;
 y0(:,3)=y0(:,3)*porositym^1.5/Rel/Dal^-1.5;
 z0=z0*porositym^2/Rel/Dal^-2;
 Sol=[x0’ y0 z0’];
 Soltot=[Soltot Sol];
 Totfluid=[Totfluid Soltot];
 Soltot=zeros(grid_nfluid,1);
%
% compute detail solution psi(z)
 a=interface; b=bposition1; tol=1e-6; n=grid_nporous;
 nm=n-1; dxx=(b-a)/nm; dtol=tol/nm;
 hstart=dxx; hmin=1e-6*dxx; hmax=dxx;
 x(1)=a; y(1,:)=phi; nfuntot=0;
for i=1:nm
 z(1)=(A2-(y(1,2)^2-y(1,1)*y(1,3))+y(1,2));
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

```matlab
ip=i+1;
x(ip)=x(i)+dxx;
[y(ip,:),nfun,hstart]=odesolve('eqporous',x(i),x(ip),y(i,:),...
dtol,hstart,hmin,hmax);
z(ip)=(A2-(y(ip,2)^2-y(ip,1)*y(ip,3))+y(ip,2));
funtot=funtot+nfun;
end
%
% rescale psi(z) back to the desired form: g(y)
x=x/sqrt(porositym/Dal);
y(:,1)=y(:,1)*sqrt(porositym/Dal)*porositym/Rel;
y(:,2)=y(:,2)*porositym^2/Dal/Rel;
y(:,3)=y(:,3)*porositym^2.5/Rel/Dal^1.5;
z=z*porositym^3/Rel/Dal^2;
Sol2=[x' y z'];
Soltot2=[Soltot2 Sol2];
Totfluid2=[Totfluid2 Soltot2];
Soltot2=zeros(grid_nporous,1);
end
zzz=targetDa-Dal % the difference between the intermediate and targeted da.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end of solveDa2.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% coxsolvefun.m
function z=coxsolvefun(omega)
%
% Root finding for bposition1 where f_y(1)=0 or psi_z(q)=0.
%
global A1 A2 porosimy phi_1 interface f2der phi IndexO beta inter
interface=inter*omega;
[lambda,nfun,hstart]=odesolve('eqfluid',0,interface,[0 0 f2der],le-6,...
interface,1e-6*interface,interface);
%
% Evaluate interface conditions
lambda3der=A1-lambda(2)^2+lambda(1)*lambda(3);
phi(1)=lambda(1)/porosimy;
phi(2)=lambda(2)/porosimy;
phi(3)=lambda(3)+1/sqrt(porosimy)*beta*lambda(2);
phi3der=lambda3der+lambda(2)^2-lambda(1)*lambda(3)+phi(1)*phi(3)-...
phi(2)*2+phi(2);
A2=phi3der+phi(2)^2-phi(2)-phi(1)*phi(3);
%
% Solve to porous wall y=1
[phi_1,nfun,hstart]=odesolve('eqporous',interface,omega,phi,1e-6,...
omega-interface,1e-6*(omega-interface),omega-interface);
Datem=porosimy/omega^2;
Retem=porosimy^1.5/sqrt(Datem)*phi_1(1);
```

APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

\[ z = \phi_1(2) \cdot \text{porosity} \cdot D_{\text{atm}} / \text{Retem} \]

if abs(z) >= 10000000
    Index0 = 1;
end

%%% end of coxsolvefun.m

% brent.m

function [R, ir] = brent(f, xi, xf, nr, dx, tol)

% 'Brent' function uses Brent's method to find multiple real roots for an
% algebraic equation. The incremental search procedure is employed to find
% bracket of a real root or a singularity. The bisection method is used to
% narrow the interval as well to find whether singularity is present. If
% no singularity found, Brent's method will be used to "polish" the root value.

% Input arguments:
%  f - dummy name of algebraic expression whose roots are sought
%  xi - start of search interval
%  xf - end of search interval
%  nr - number of real roots sought
%  dx - initial search increment used to locate bracket for a single root.
%  tol - absolute root tolerance or accuracy

% Output arguments:
%  R - Array of roots found
%  ir - number of roots found in interval [xi xf]

% Initialize the variable

global Index0

x1 = xi; y1 = feval(f, x1); ir = 0;

% Find if the first root found if f(xi)=0
if y1 == 0
    ir = ir + 1; R(ir) = x1; x1 = x1 + dx; y1 = feval(f, x1);
end

% Loop to find remaining roots when ir<=nr and x<=xf
while ir < nr
    % Incremental search to locate interval [x1, x3] having single real root or
    % singularity
    interval_found = 0;
    while interval_found == 0
        % ...
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

\[
x_3 = x_1 + dx;
\]

if (dx > 0 & x3 > xf) | (dx < 0 & x3 < xf)
    fprintf('\n Only %5.0f roots found. \n', ir)
    return
end

y3 = feval(f, x3);
if y1 * y3 > 0
    x1 = x3; y1 = y3;
    if (abs(IndexO - l) < le - 6) % for solvepositionbauto only
        ir = 1;
        % for solvepositionbauto only
        R(ir) = 0;
        % for solvepositionbauto only
        root_found = 0;
        singularity = 0;
        dyold = abs(y3 - y1);
        for i = 1:3
            x2 = (x1 + x3) / 2; y2 = feval(f, x2);
            if y1 * y2 > 0
                x1 = x2; y1 = y2;
            else
                x3 = x2; y3 = y2;
            end
        end
        x2 = (x1 + x3) / 2; y2 = feval(f, x2);
        dy = abs(y3 - y1);
        if dy > dyold
            singularity = 1;
        else
            root_found = 0;
        end
    % Brent’s method to polish the root
    while root_found == 0 & singularity == 0
        r = y2 / y3; s = y2 / y1; t = y1 / y3;
        q = (r - 1) * (s - 1) * (t - 1);
        p = s * (t * (r - t) * (x3 - x2) - (1 - r) * (x2 - x1));
        x4 = x2 + p / q;
        y4 = feval(f, x4);
        if abs((p / q) / x4) <= tol
            ir = ir + 1; R(ir) = x4; root_found = 1;
            break
        end
    end
\]
if (dx>0 & x1<x4 & x4<x2) | (dx<0 & x1>x4 & x4>x2)
x3=x2; y3=y2; x2=x4; y2=y4;
elseif (dx>0 & x2<x4 & x4<x3) | (dx<0 & x2>x4 & x4>x3)
x1=x2; y1=y2; x2=x4; y2=y4;
end
end
x1=x3; y1=y3;

% end of brent.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z=eqfluid(x,g)

% 'eqfluid.m' is a function m-file that defines the equations set for
% the original transformed from the 3rd order ODE
% xx1(1)=A1

global A1
z(1)=g(2);
z(2)=g(3);
z(3)=A1-(g(2)^2-g(1)*g(3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end of eqfluid.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function z=eqporous(x,g)

% 'eqporous.m' is a function m-file that defines the equations set for
% the original transformed from the 3rd order ODE

global A2
valfac=1; valfac2=1;
z(1)=g(2);
z(2)=g(3);
z(3)=A2-valfac2*(g(2)^2-g(1)*g(3))+g(2)*valfac;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% end of eqporous.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [yb,nfun,hstart]=odesolve1(f,a,b,ya,tol,hstart,hmin,hmax)

% 'odesolve1.m' is a slight modification of 'odesolve.m'
% here it calls $rkck1$ routine instead of $rkck$

m=length(ya);

hold=hstart; xold=a; yold=ya; nfun=0; success=0;
while success==0
    nfun=nfun+6*m;
    [ynew,ydiff]=rkck1(f,xold,yold,hold);
    gamma=(tol*hold/(ydiff*(b-a)))^0.25;
    hnew=0.8*gamma*hold;
    if gamma<1
        if hnew<hold/10
            hnew=hold/10;
        end
        if hnew<hmin
            fprintf('
 odesolve failed at x=%8.3e 

,xold,xold')
            error
        end
        hold=hnew;
    else
        if hnew>5*hold
            hnew=5*hold;
        end
        if hnew>hmax
            hnew=hmax;
        end
        if xold+hold<b
            xold=xold+hold; hold=hnew; yold=ynew;
        else
            success=1; hstart=hnew; hold=b-xold;
            nfun=nfun+6*m;
            [yb,ydiff]=rkck1(f,xold,yold,hold);
        end
    end
end

function [yb,nfun,hstart]=odesolve2(f,a,b,ya,tol,hstart,hmin,hmax)

% 'odesolve2.m' is a slight modification of 'odesolve.m'
% here it calls $rkck2$ routine instead of $rkck$

m=length(ya);

hold=hstart; xold=a; yold=ya; nfun=0; success=0;
while success==0

    nfun=nfun+6*m;
    [ynew,ydiff]=rkck2(f,xold,yold,hold);
    gamma=(tol*hold/(ydiff*(b-a)))^0.25;
    hnew=0.8*gamma*hold;

    if gamma<1
        if hnew<hold/10
            hnew=hold/10;
        end
        if hnew<hmin
            fprintf('
 odesolve failed at x=%3.3e \n\n,xold',xold)
            error
            hold=hnew;
        end
    else
        if hnew>5*hold
            hnew=5*hold;
        end
        if hnew>hmax
            hnew=hmax;
        end
        if xold+hold<b
            xold=xold+hold; hold=hnew; yold=ynew;
        else
            success=1; hstart=hnew; hold=b-xold;
            nfun=nfun+6*m;
            [yb,ydiff]=rkck2(f,xold,yold,hold);
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of odesolve2.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [y2, ydiff]=rkck1(f,x1,y1,h)

    global Totfluid_cal
    % 'rkck1.m' make used of the base solutions f(y) in calculating
    % the perturbed solutions F(y) in fluid region.
    Totfluid_cal=solvetoanyposition(x1);
    kl=h*feval(f,x1,y1);

    Totfluid_cal=solvetoanyposition(x1+h/5);
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

k2 = h*feval(f, x1 + h/5, y1 + k1/5);

Totfluid_cal = solvetoanyposition(x1 + 3*h/10);
k3 = h*feval(f, x1 + 3*h/10, y1 + 3*k1/40 + 9*k2/40);

Totfluid_cal = solvetoanyposition(x1 + 3*h/5);
k4 = h*feval(f, x1 + 3*h/5, y1 + 3*k1/10 - 9*k2/10 + 6*k3/5);

Totfluid_cal = solvetoanyposition(x1 + 7*h/8);
k5 = h*feval(f, x1 + y1 - 11*k1/54 + 5*k2/2 - 70*k3/27 + 35*k4/27);

Totfluid_cal = solvetoanyposition(x1 + h);
k6 = h*feval(f, x1 + 7*h/8, y1 + 1631*k1/55296 + 175*k2/512 + 575*k3/13824 +
          44275*k4/110592 + 253*k5/4096);

y2 = 37*k1/378 + 250*k3/621 + 2825/27648 + 175*k2/512 + 575*k3/13824 +
    44275*k4/110592 + 253*k5/4096;

ydiff0 = abs((37/378 - 2825/27648) * k1 + (250/621 - 18575/48384) * k3 +
             (125/594 - 13525/55296) * k4 - 277*k5/14336 + (512/1771 - 1/4) * k6);

ydiff = max(ydiff0);

% end of rkck1.m

function [y2, ydiff] = rkck2(f, x1, y1, h)

global Totfluid_calmed

% 'rkck2.m' make used of the base solutions f(y) and g(y) in calculating
% the perturbed solutions G(y) in medium region.

Totfluid_calmed = solvetoanyposition_p(x1);

k1 = h*feval(f, x1, y1);

Totfluid_calmed = solvetoanyposition_p(x1 + h/5);
k2 = h*feval(f, x1 + h/5, y1 + k1/5);

Totfluid_calmed = solvetoanyposition_p(x1 + 3*h/10);
k3 = h*feval(f, x1 + 3*h/10, y1 + 3*k1/40 + 9*k2/40);

Totfluid_calmed = solvetoanyposition_p(x1 + 3*h/5);
k4 = h*feval(f, x1 + 3*h/5, y1 + 3*k1/10 - 9*k2/10 + 6*k3/5);

Totfluid_calmed = solvetoanyposition_p(x1 + h);
k5 = h*feval(f, x1 + y1 - 11*k1/54 + 5*k2/2 - 70*k3/27 + 35*k4/27);

Totfluid_calmed = solvetoanyposition_p(x1 + 7*h/8);
k6 = h*feval(f, x1 + 7*h/8, y1 + 1631*k1/55296 + 175*k2/512 + 575*k3/13824 +
          44275*k4/110592 + 253*k5/4096);

y2 = 37*k1/378 + 250*k3/621 + 125*k4/594 + 512*k6/1771 + y1;
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

% The value of scalar 'ydiff' is taken as the maximum of the differences % between each element of vectors y2(5th order) and y2(4th order). It is % important to give ydiff0 a algebraic form, otherwise, if just put % ydiff=y2(5th)-y2(4th), the computer will give divided by zero warning % at 'odesolve at line 48', which is assumed to be the roundoff errors.

    ydiff0=abs((37/378-2825/27648)*k1+(250/621-18575/48384)*k3+... 
    (125/594-13525/55296)*k4-277*k5/14336+(512/1771-1/4)*k6);

    ydiff=max(ydiff0);

% end of rkck2.m

% solvetoanyposition.m

function Totfluid_cal=solvetoanyposition(xx)
% to get base solution at any desired position in the fluid region,
% basically it is an initial value solver.

global f2der A1 bpositionl Rel inter Tot_Totfluidcal record_xx
if length(Tot_Totfluidcal)==0
    hit=0;
else
    hit=length(Tot_Totfluidcal(:,1));
end
if hit>=10
    for i=1:hit
        if xx==record_xx(i)
            Totfluid_cal=Tot_Totfluidcal(i,:);
            return
        end
    end
end

if xx==0
    x0=0;
    y0=[0 0 f2der];
    z0=A1-(y0(2)^2-y0(1)*y0(3));
else
    [y0,nfun,hstart]=odesolve('eqfluid',0,bposition1*xx,[0 0 f2der],1e-6,...
        bposition1*inter,1e-6*bposition1*inter,bposition1*xx);
    x0=xx;
    z0=A1-(y0(2)^2-y0(1)*y0(3));
end

y0(1)=y0(1)*bposition1/Rel;
y0(2)=y0(2)*bposition1^2/Rel;
y0(3)=y0(3)*bposition1^3/Rel;
z0=z0*bposition1^4/Rel;

Totfluid_cal=[x0 y0 z0];
APPENDIX D. CODE FOR THE SIMILARITY SOLUTION

Tot_Totfluidcal=[Tot_Totfluidcal; Totfluid_cal];
record_xx=[record_xx xx];

% solvetoanyposition_p.m

function Totfluid_calmed=solvetoanyposition_p(xx)
% to get base solution at any desired position in the porous region,
% basically it is an initial value solver.
global A2 bpositionl Re1 Dal porositym phi interface inter...
    Tot_Totfluidcal_p record_xx_p
if length(Tot_Totfluidcal_p)==0
    hit=0;
else
    hit=length(Tot_Totfluidcal_p(:,1));
end
if hit>=10
    for i=l:hit
        if xx==record_xx_p(i)
            Totfluid_calmed=Tot_Totfluidcal_p(i,:);
            return
        end
    end
end
if xx==inter
    xO=xx;
    yO=phi;
    zO=A2-(yO(2)^2-y0(1)*y0(3))+yO(2);
else
    [yO,nfun,hstart]=odesolve('eqporous',interface, xx*bposition1,phi,le-6,...
                   xx*bposition1-interface,le-6*(bposition1*xx-interface),...  
                   bposition1*xx-interface);
    xO=xx;
    zO=A2-(y0(2)^2-y0(1)*y0(3))+y0(2);
end
y0(1)=y0(1)*sqrt(porositym/Dal)*porositym/Re1;
y0(2)=y0(2)*porositym^2/Dal/Re1;
y0(3)=y0(3)*porositym^2.5/Re1/Dal^1.5;
z0=z0*porositym^3/Re1/Dal^2;
Totfluid_calmed=[xO yO zO];
Tot_Totfluidcal_p=[Tot_Totfluidcal_p; Totfluid_calmed];
record_xx_p=[record_xx_p xx];

% eqstability.m
function z=eqstability(x,g)
    global B1_r lambda_r Re1 Totfluid_cal
    z(1)=g(2);
    z(2)=g(3);
    z(3)=B1_r+Re1*(g(3)*Totfluid_cal(2)-2*Totfluid_cal(3)*g(2)+Totfluid_cal(4)*... 
        g(1))+Re1*(lambda_r)*g(2);

function z=eqstability_p(x,g)
    global lambda_r Re1 Da1 Totfluid_calmed B2 porositym
    z(1)=g(2);
    z(2)=g(3);
    z(3)=B2+Da1/porositym*(g(3)*Totfluid_calmed(2)-2*Totfluid_calmed(3)*g(2)+... 
        Totfluid_calmed(4)*g(1))+Re1*(lambda_r)*g(2)+porositym/Da1*g(2);

end of eqstability.m

end of eqstability_p.m
Appendix E

CODE FOR THE ONE DOMAIN PROBLEM

E.1 Introduction

Here we present the detailed computer algorithm and code written in FORTRAN 90 for the one-domain approach outlined in Chapter 5.

E.1.1 Formulation of the Problem

Equation

The equations of dimensionless form is as follows:

\[
\frac{\partial P}{\partial t} + \frac{\partial w_x}{\partial x} + \frac{\partial w_y}{\partial y} = 0
\]  
(E.1)

\[
\frac{\partial w_x}{\partial t} + \frac{1}{\epsilon} w_x \frac{\partial w_x}{\partial x} + \frac{1}{\epsilon} w_y \frac{\partial w_x}{\partial y} = -\frac{\epsilon}{\partial x} \left( \frac{\partial^2 w_x}{\partial x^2} + \frac{\partial^2 w_x}{\partial y^2} \right) - \frac{Be}{DaRe} w_x
\]  
(E.2)

\[
\frac{\partial w_y}{\partial t} + \frac{1}{\epsilon} w_x \frac{\partial w_y}{\partial x} + \frac{1}{\epsilon} w_y \frac{\partial w_y}{\partial y} = -\frac{\epsilon}{\partial y} \left( \frac{\partial^2 w_y}{\partial x^2} + \frac{\partial^2 w_y}{\partial y^2} \right) - \frac{Be}{DaRe} w_y
\]  
(E.3)

where \( Da = K/H^2 \), \( Re = \rho U_w H/\mu \), and \( w_x \) and \( w_y \) denote the \( x \) and \( y \) velocity components respectively. The reference parameters are: the reference length is the total channel height "\( H \)", reference velocity: "\( U_w \)", reference time: \( t = H/U_w \) and pressure \( P \) is nondimensionlized with "\( \rho f U_w^2 \)". Porosity \( \epsilon = 1 \) in fluid region and \( \epsilon < 1 \) in porous medium region. \( B \) is a binary constant equals 0 for fluid region and equals 1 for porous region.
To solve the equations for steady, incompressible flow using artificial compressibility method, the pressure term in continuity equation is added an artificial compressibility $\beta$, so that the equation becomes:

\[
\frac{\partial P}{\partial t} + \frac{\partial w_x}{\beta \epsilon \partial x} + \frac{\partial w_y}{\beta \epsilon \partial y} = 0 \tag{E.4}
\]

\[
\frac{\partial w_x}{\partial t} + \frac{1}{\epsilon} \frac{\partial w_x^2}{\partial x} + \frac{1}{\epsilon} \frac{\partial (w_x w_y)}{\partial y} = -\frac{\partial P}{\partial x} + \frac{1}{Re} \left( \frac{\partial^2 w_x}{\partial x^2} + \frac{\partial^2 w_x}{\partial y^2} \right) - \frac{B \epsilon}{Da Re} w_x \tag{E.5}
\]

\[
\frac{\partial w_y}{\partial t} + \frac{1}{\epsilon} \frac{\partial (w_x w_y)}{\partial x} + \frac{1}{\epsilon} \frac{\partial w_y^2}{\partial y} = -\frac{\partial P}{\partial y} + \frac{1}{Re} \left( \frac{\partial^2 w_y}{\partial x^2} + \frac{\partial^2 w_y}{\partial y^2} \right) - \frac{B \epsilon}{Da Re} w_y \tag{E.6}
\]

This equation set can be written into the form of

\[
\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S \tag{E.7}
\]

where

\[
U = \begin{pmatrix} P \\ w_x \\ w_y \end{pmatrix}
\]

\[
F = \begin{pmatrix} \frac{w_x}{\epsilon \beta} \\ \frac{w_x^2}{\epsilon} + \epsilon P - \frac{1}{Re} \frac{\partial w_x}{\partial x} \\ \frac{w_x w_y}{\epsilon} - \frac{1}{Re} \frac{\partial w_y}{\partial x} \end{pmatrix}
\]

\[
G = \begin{pmatrix} \frac{w_y}{\epsilon \beta} \\ \frac{w_x w_y}{\epsilon} - \frac{1}{Re} \frac{\partial w_x}{\partial y} \\ \frac{w_y^2}{\epsilon} - \frac{1}{Re} \frac{\partial w_y}{\partial y} + \epsilon P \end{pmatrix}
\]

\[
S = \begin{pmatrix} 0 \\ -\frac{B \epsilon}{Da Re} w_x \\ -\frac{B \epsilon}{Da Re} w_y \end{pmatrix}
\]

Equation (E.7) in control volume finite form is:

\[
\frac{dU_{i,j}}{dt} + \frac{F_{i+1/2,j} - F_{i-1/2,j}}{\Delta x} + \frac{G_{i,j+1/2} - G_{i,j-1/2}}{\Delta y} = S_{i,j} \tag{E.8}
\]
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

A fully discrete form using implicit Euler time advance scheme yields

\[ \frac{U_{i,j}^{n+1} - U_{i,j}^n}{\Delta t} = \frac{\delta U_{i,j}^{n+1}}{\Delta t} = -F_{i+1/2,j}^{n+1} - F_{i-1/2,j}^{n+1} - \frac{G_{i,j+1/2}^{n+1} - G_{i,j-1/2}^{n+1}}{\Delta y} + S_{i,j} \]  
(E.9)

It is reasonable to assume that \( F_{i+1/2,j}^{n+1} \) is a function of \( U_{i,j}^{n+1} \) and \( U_{i+1,j}^{n+1} \), as well for \( G_{i,j+1/2}^{n+1} \) is a function of \( U_{i,j}^{n+1} \) and \( U_{i,j+1}^{n+1} \) and \( S_{i,j}^{n+1} \) is a function of \( U_{i,j}^{n+1} \).

The quantities at time step \( n+1 \) can be approximated with quantities at older time step \( n \) by Taylor series expansion, from which we know that for example,

\[ F_{i+1/2,j}^{n+1} = F(U_{i,j}^n, U_{i+1,j}^n) \]

\[ = F(U_{i,j}^n + \delta U_{i,j}^n, U_{i+1,j}^n + \delta U_{i+1,j}^n) \]

\[ = F(U_{i,j}^n, U_{i+1,j}^n) + \frac{\partial F_{i+1/2,j}^n}{\partial U_{i,j}} \delta U_{i,j}^n + \frac{\partial F_{i+1/2,j}^n}{\partial U_{i+1,j}} \delta U_{i+1,j}^n + O(\delta U^2) \]  
(E.10)

Hence,

\[ \frac{\delta U_{i,j}^{n+1}}{\delta t} = -\frac{1}{\Delta x}(F_{i+1/2,j}^n) + \frac{\partial F_{i+1/2,j}^n}{\partial U_{i,j}} \delta U_{i,j}^n + \frac{\partial F_{i+1/2,j}^n}{\partial U_{i+1,j}} \delta U_{i+1,j}^n \]

\[ + \frac{1}{\Delta x}(F_{i-1/2,j}^n) \delta U_{i,j}^n + \frac{\partial F_{i-1/2,j}^n}{\partial U_{i,j}} \delta U_{i,j}^n + \frac{\partial F_{i-1/2,j}^n}{\partial U_{i-1,j}} \delta U_{i-1,j}^n \]

\[ - \frac{1}{\Delta y}(G_{i,j+1/2}^n) \delta U_{i,j}^n + \frac{\partial G_{i,j+1/2}^n}{\partial U_{i,j}} \delta U_{i,j+1}^n + \frac{\partial G_{i,j+1/2}^n}{\partial U_{i,j+1}} \delta U_{i,j+1}^n \]

\[ + \frac{1}{\Delta y}(G_{i,j-1/2}^n) \delta U_{i,j}^n + \frac{\partial G_{i,j-1/2}^n}{\partial U_{i,j-1}} \delta U_{i,j-1}^n + \frac{\partial G_{i,j-1/2}^n}{\partial U_{i,j-1}} \delta U_{i,j-1}^n \]

\[ + \frac{\partial S_{i,j}^n}{\partial U_{i,j}} \delta U_{i,j}^n \]

(E.11)

Therefore

\[ \left( \frac{I}{\Delta t} + \frac{1}{\Delta x} \frac{\partial F_{i+1/2,j}^n}{\partial U_{i,j}} - \frac{1}{\Delta x} \frac{\partial F_{i-1/2,j}^n}{\partial U_{i,j}} + \frac{1}{\Delta y} \frac{\partial G_{i,j+1/2}^n}{\partial U_{i,j}} - \frac{1}{\Delta y} \frac{\partial G_{i,j-1/2}^n}{\partial U_{i,j}} \right) \delta U_{i,j}^{n+1} \]

\[ + \frac{1}{\Delta x} \frac{\partial F_{i+1/2,j}^n}{\partial U_{i+1,j}} \delta U_{i,j+1}^{n+1} - \frac{1}{\Delta x} \frac{\partial F_{i-1/2,j}^n}{\partial U_{i-1,j}} \delta U_{i,j-1}^{n+1} + \frac{1}{\Delta y} \frac{\partial G_{i,j+1/2}^n}{\partial U_{i+1,j}} \delta U_{i,j+1}^{n+1} \]

\[ - \frac{1}{\Delta y} \frac{\partial G_{i,j-1/2}^n}{\partial U_{i,j-1}} \delta U_{i,j-1}^{n+1} = \frac{F_{i+1/2,j}^n - F_{i-1/2,j}^n}{\Delta x} - \frac{G_{i,j+1/2}^n - G_{i,j-1/2}^n}{\Delta y} + S_{i,j}^n \]  
(E.12)

Equation (E.12) could be written as

\[ (I + \Delta tB_x + \Delta tB_y) \delta U_{i,j}^{n+1} + \Delta tA_x \delta U_{i-1,j}^{n+1} + \Delta tC_x \delta U_{i+1,j}^{n+1} + \Delta tA_y \delta U_{i,j+1}^{n+1} + \Delta tC_y \delta U_{i,j+1}^{n+1} \]

\[ = \Delta t(FI)_{i,j}^n \]  
(E.13)
where

\[
A_x = \frac{1}{\Delta x} \frac{\partial F^n_{i-1/2,j}}{\partial U_{i-1,j}}
\]

\[
B_x = \frac{1}{\Delta x} \frac{\partial F^n_{i+1/2,j}}{\partial U_{i,j}} - \frac{1}{\Delta x} \frac{\partial F^n_{i-1/2,j}}{\partial U_{i,j}} - \frac{1}{2} \frac{\partial S^n_{i,j}}{\partial U_{i,j}}
\]

\[
C_x = \frac{1}{\Delta x} \frac{\partial F^n_{i+1/2,j}}{\partial U_{i+1,j}}
\]

\[
A_y = -\frac{1}{\Delta y} \frac{\partial G^n_{i,j-1/2}}{\partial U_{i,j-1}}
\]

\[
B_y = +\frac{1}{\Delta y} \frac{\partial G^n_{i,j+1/2}}{\partial U_{i,j+1}} - \frac{1}{\Delta y} \frac{\partial G^n_{i,j-1/2}}{\partial U_{i,j-1}} - \frac{1}{2} \frac{\partial S^n_{i,j}}{\partial U_{i,j}}
\]

\[
C_y = \frac{1}{\Delta y} \frac{\partial G^n_{i,j+1/2}}{\partial U_{i,j+1}}
\]

\[
FI = -\frac{F^n_{i+1/2,j} - F^n_{i-1/2,j}}{\Delta x} - \frac{G^n_{i+1/2,j} - G^n_{i,j-1/2}}{\Delta y} + S^n_{i,j}
\] (E.14)

Since

\[
F_{i+1/2,j} = \begin{pmatrix}
\frac{w_x(i+1/2,j)}{\varepsilon} \\
\frac{w_x(i+1/2,j)}{\varepsilon} + \varepsilon P_{i+1/2,j} - \frac{1}{Re} \frac{\partial w_x}{\partial x} \bigg|_{i+1/2,j} \\
\frac{w_x(i+1/2,j)}{\varepsilon} \frac{w_y(i+1/2,j)}{\varepsilon} - \frac{1}{Re} \frac{\partial w_y}{\partial x} \bigg|_{i+1/2,j}
\end{pmatrix}
\]

\[
\begin{pmatrix}
\frac{w_x(i,j)+w_x(i+1,j)}{2\varepsilon} \\
\left(\frac{w_x(i,j)+w_x(i+1,j)}{2}\right)^2 \frac{1}{\varepsilon} + \frac{\varepsilon(P_{i,j}+P_{i+1,j})}{2} - \frac{1}{Re} \frac{w_x(i,j)-w_x(i+1,j)}{\Delta x} \\
\frac{w_y(i,j)+w_y(i+1,j)}{2} \\
\left(\frac{w_y(i,j)+w_y(i+1,j)}{2}\right)^2 \frac{1}{\varepsilon} + \frac{\varepsilon(P_{i,j}+P_{i+1,j})}{2} - \frac{1}{Re} \frac{w_y(i,j)-w_y(i+1,j)}{\Delta y}
\end{pmatrix}
\]

\[
G_{i,j+1/2} = \begin{pmatrix}
\frac{w_y(i,j)+w_y(i,j+1)}{2\varepsilon} \\
\left(\frac{w_y(i,j)+w_y(i,j+1)}{2}\right)^2 \frac{1}{\varepsilon} + \frac{\varepsilon(P_{i,j}+P_{i,j+1})}{2} - \frac{1}{Re} \frac{w_y(i,j)-w_y(i,j+1)}{\Delta y}
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 \\
\frac{-Re}{D_0 Re} w_x(i,j) \\
\frac{-Re}{D_0 Re} w_y(i,j)
\end{pmatrix}
\]
Using the method of approximate factorization, Solving Eq. (E.13) equals to solve the equation shown as following:

\[
(I + \Delta t A_x E_{-1,0} + \Delta t B_x + \Delta t C_x E_{1,0})(I + \Delta t A_y E_{0,-1} + \Delta t B_y + \Delta t C_y E_{0,1}) \delta U_{i+1,j}^{n+1} = \Delta t (F I)_{i,j}^{n}
\]

where \(E_{i,j}^{n}\) is subscript operator,

This leads to

\[
A_x = -\frac{1}{\Delta x} \begin{bmatrix}
0 & \frac{1}{2\epsilon} \\
\frac{\epsilon}{2} & \frac{1}{2\epsilon}(\overline{w}_x i,j + \overline{w}_x i-1,j) + \frac{1}{\Re \Delta x} & 0 \\
0 & \frac{1}{4\epsilon}(\overline{w}_y i,j + \overline{w}_y i-1,j) + \frac{1}{4\epsilon}(\overline{w}_x i,j + \overline{w}_x i-1,j) + \frac{1}{\Re \Delta x}
\end{bmatrix}
\]

\[
B_x = -\frac{1}{\Delta x} \begin{bmatrix}
0 & 0 \\
0 & -\frac{2}{\Re \Delta x} + \frac{1}{2\epsilon}(\overline{w}_x i-1,j - \overline{w}_x i+1,j) - \frac{B_c}{2DaRe \Delta x} & 0 \\
0 & \frac{1}{4\epsilon}(\overline{w}_y i-1,j + \overline{w}_y i+1,j) - \frac{1}{4\epsilon}(\overline{w}_x i-1,j + \overline{w}_x i+1,j) - \frac{2}{\Re \Delta x} - \frac{B_c}{2DaRe \Delta x}
\end{bmatrix}
\]

\[
C_x = \frac{1}{\Delta x} \begin{bmatrix}
0 & \frac{1}{2\epsilon} \\
\frac{\epsilon}{2} & \frac{1}{2\epsilon}(\overline{w}_x i,j + \overline{w}_x i+1,j) - \frac{1}{\Re \Delta x} & 0 \\
0 & \frac{1}{4\epsilon}(\overline{w}_y i,j + \overline{w}_y i+1,j) & \frac{1}{4\epsilon}(\overline{w}_x i,j + \overline{w}_x i+1,j) - \frac{1}{\Re \Delta x}
\end{bmatrix}
\]

\[
A_y = -\frac{1}{\Delta y} \begin{bmatrix}
0 & 0 \\
0 & \frac{1}{4\epsilon}(\overline{w}_y i,j-1 + \overline{w}_y i,j) + \frac{1}{\Re \Delta y} & \frac{1}{4\epsilon}(\overline{w}_x i,j-1 + \overline{w}_x i,j) \\
\frac{\epsilon}{2} & 0 & \frac{1}{2\epsilon}(\overline{w}_y i,j+1 + \overline{w}_y i,j-1) + \frac{1}{\Re \Delta y}
\end{bmatrix}
\]
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

\[ B_y = -\frac{1}{\Delta y} \begin{bmatrix} 0 & 0 \\ 0 & -\frac{2}{Re\Delta y} + \frac{1}{4\epsilon} (\overline{w}_y \ i,j-1 - \overline{w}_y \ i,j+1) - \frac{Re}{2DaRe} \Delta y \ \\
0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{2\epsilon} (\overline{w}_y \ i,j-1 + \overline{w}_y \ i,j+1) - \frac{2}{Re\Delta y} - \frac{Re}{2DaRe} \Delta y \end{bmatrix} \]

\[ C_y = \frac{1}{\Delta y} \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{4\epsilon} (\overline{w}_y \ i,j + \overline{w}_y \ i,j+1) - \frac{1}{Re\Delta y} \frac{1}{2\epsilon} (\overline{w}_x \ i,j + \overline{w}_x \ i,j+1) \\ \frac{1}{2\epsilon} (\overline{w}_y \ i,j + \overline{w}_y \ i,j+1) - \frac{1}{Re\Delta y} \end{bmatrix} \]

E.1.2 Setup of the Boundary Conditions

The boundary conditions include implicit and explicit conditions, where the explicit ones are used to determine the ghost cell values and the implicit ones are for the update of the cell values. The boundary conditions include upper impermeable wall and lower porous wall; determination of inflow and outflow conditions.

- **Upper Wall**

The upper wall is a solid impermeable wall, therefore, the boundary conditions are: \( w_x = w_y = 0 \) and \( \frac{\partial P}{\partial y} = 0 \). The values in the ghost cells for our coordinate could be set as:

\[ P(i,0) = P(i,1) \quad (E.16) \]
\[ w_x(i,0) = -w_x(i,1) \quad (E.17) \]
\[ w_y(i,0) = -w_y(i,1) \quad (E.18) \]

The corresponding implicit boundary condition is:

\[
\begin{pmatrix}
1 & 1 \\
1 & -1 \\
1 & -1
\end{pmatrix}
\begin{pmatrix}
\delta P_{i,0}^{n+1} \\
\delta w_x_{i,0}^{n+1} \\
\delta w_y_{i,0}^{n+1}
\end{pmatrix} =
\begin{pmatrix}
1 & 1 \\
0 & -1 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
\delta P_{i,1}^{n+1} \\
\delta w_x_{i,1}^{n+1} \\
\delta w_y_{i,1}^{n+1}
\end{pmatrix}
\]

This condition is used to construct the coefficient matrix for ghost cells at the upper wall.
• **Bottom Wall**

The bottom wall is a porous wall, which implies a uniform suction in $y$ direction and zero velocity at $x$ direction. The explicit pressure boundary condition is derived from the $y$ momentum equation, i.e.,

\[
\frac{1}{\epsilon} \frac{\partial (w_x w_y)}{\partial x} + \frac{1}{\epsilon} \frac{\partial w_y^2}{\partial y} = -\epsilon \frac{\partial P}{\partial y} + \frac{1}{Re} \left( \frac{\partial^2 w_y}{\partial x^2} + \frac{\partial^2 w_y}{\partial y^2} \right) - \frac{Be}{DaRe} w_y
\]  

(E.19)

Since $\frac{\partial}{\partial x} = 0$ because $w_z = 0$ throughout $x$ direction, from continuity we thus have also $\frac{\partial w_y}{\partial y} = 0$. Therefore, the above equation reduces to:

\[
\frac{\partial P}{\partial y} = \frac{1}{DaRe} w_y
\]  

(E.20)

where $w_{y\text{ wall}} = \frac{U_m}{U_w} = 1$. This means that the explicit conditions become:

\[
P(i, \text{gridy} + 1) = P(i, \text{gridy}) - \frac{\Delta y}{ReDa}
\]  

(E.21)

\[
w_x(i, \text{gridy} + 1) = -w_x(i, \text{gridy})
\]  

(E.22)

\[
w_y(i, \text{gridy} + 1) = -w_y(i, \text{gridy}) + 2w_{y\text{ wall}}
\]  

(E.23)

The corresponding implicit conditions are:

\[
\begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix}
\begin{pmatrix}
\delta P^{n+1}_{i, \text{gridy}} \\
\delta w^{n+1}_{x,i, \text{gridy}} \\
\delta w^{n+1}_{y,i, \text{gridy}}
\end{pmatrix}
= \begin{pmatrix}
1 \\
-1 \\
-1
\end{pmatrix}
\begin{pmatrix}
\delta P^{n+1}_{i, \text{gridy}+1} \\
\delta w^{n+1}_{x,i, \text{gridy}+1} \\
\delta w^{n+1}_{y,i, \text{gridy}+1}
\end{pmatrix}
\]

which is the same with the implicit upper wall condition.

• **Inflow**

In this case, we specify the inflow velocity $w_{x\text{ in}}$ and $w_{y\text{ in}}$, where $w_{y\text{ in}} = 0$. The velocity $w_{x\text{ in}}$ can be specified to different values, it is reasonable to assume uniform inflow of velocity $U_{\text{in}} = w_{x\text{ in}}/U_{\text{suc}}$. A pressure gradient can be assumed at the inflow position, which finally gives:

\[
P(0, j) = +P(1, j) - \Delta x \frac{\partial P}{\partial x_{\text{in}}}
\]  

(E.24)

\[
w_x(0, j) = -w_x(1, j) + 2U_{\text{in}}
\]  

(E.25)

\[
w_y(0, j) = -w_y(1, j)
\]  

(E.26)
The corresponding implicit boundary conditions are:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\delta P^n_{i,j} \\
\delta w^i_x_{i,j} \\
\delta w^i_y_{i,j}
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}
\begin{pmatrix}
\delta P^1_{i,j} \\
\delta w^1_x_{i,j} \\
\delta w^1_y_{i,j}
\end{pmatrix}
\]

**Outflow**

At channel outlet we assume the flow will be fully-developed, i.e., the pressure, velocity components are not the functions of position \(x\) anymore;

Following the above, we have:

We could assume the pressure \(P_{out}\), and both the velocity gradients along the \(x\) and \(y\) directions should be zeros. These give:

\[
P(gridx + 1, j) = -P(gridx, j) + 2P_{out}
\]  
(E.27)

\[
w_x(gridx + 1, j) = w_x(gridx, j)
\]  
(E.28)

\[
w_y(gridx + 1, j) = w_y(gridx, j)
\]  
(E.29)

This corresponds to the implicit conditions as following:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\delta P^{n+1}_{gridx,j} \\
\delta w^{n+1}_x{gridx,j} \\
\delta w^{n+1}_y{gridx,j}
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{pmatrix}
\begin{pmatrix}
\delta P^{n+1}_{gridx+1,j} \\
\delta w^{n+1}_x{gridx+1,j} \\
\delta w^{n+1}_y{gridx+1,j}
\end{pmatrix}
\]

**E.1.3 Linearization and Solving the Linear Equations**

The approximate factorization method is used to reduce the problem to solve a system of a block tri-diagonal matrix for constant "i" and "j" values respectively. For example, for constant "i", we have:
where the right hand side $RHS_i$ is the sum of source term and flux integral at time step $n$. In this solution, to dampen the fluctuation of the pressure terms, an artificial total pressure gradient is also added on the right hand side expression. This treatment is found necessary to have converged solution. The above block tri-diagonal matrix is solved using standard Thomas algorithm. The control of the source term and the porosity values in different flow domains are operated through turning on/off the source term values of cell number falling into the region.

The code for the One-domain problem is is written in Fortran90. We presented it here as onedomaincode.f90. The main program, is listed first followed by the subroutines.
One-Domain Solution for
Fluid flow in a channel partially filled with
a porous medium
by Chuntao Deng
last updated July 6, 2004

This program codes the Navier-Stokes Equation for fluid flow in a channel
partially filled with a porous medium by a one-domain approach.
The algorithm and the parameters in this code could be found in the
preceding descriptions in the chapter.

program onedomaincode

implicit none
integer i,j,k,m,gridx,gridy,maxiter,iterno
parameter (gridx=480,gridy=60,maxiter=100)
double precision p(0:gridx+1,0:gridy+1),
                        u(0:gridx+1,0:gridy+1),
                        v(0:gridx+1,0:gridy+1),
                        x(0:gridx+1),y(0:gridy+1),divp(3,gridx,gridy),
                        errors(3), deltaU(3,gridx,gridy),
                        Iden(3,3), flcom(3,gridx,gridy),
                        RHSxx(3,0:gridy+1),RHSyy(3,0:gridx+1),
                        LHSxx(3,3,0:gridy+1),LHSyy(3,3,0:gridx+1),
                        xdis, ydis,deltax,deltay,deltat,conv_cri,errmax,
                        ORfactor, p0,u0,v0,pi,LHSx(3,3,3,2),LHSy (3,3,3,2),
                        LHSxA (3,3,gridx,gridy),LHSxB (3,3,gridx,gridy),
                        LHSyC(3,3,gridx,gridy), LHSxA(3,3,gridx,gridy),
                        LHSxB(3,3,gridx,gridy),LHSxC(3,3,gridx,gridy),
                        sij(3,gridx, gridy), Da, lambda,Re

parameter(Re=5.d0,Da=1.e-3,lambda=1.d0)
parameter(p0=1.d0, u0=1.d0,v0=1.d0, pi=3.1415926535897932d0)
parameter(xdis=8.0d0, ydis=1.0d0,deltat=0.02d0,conv_cri=1.d-4)
parameter(ORfactor=1.2d0)

! gridx : grid number in x direction
! gridy : grid number in y direction
! maxiter : allowed maximum iteration times
! p : pressure
! u,v : velocity in x and y direction respectively
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 135

! flcom : flux integral computed
! divp : source term added to suppress pressure oscillation
! deltat : length of time step
! x,y : x and y coordinates
! deltax : sizes of cells in x and y direction respectively
! deltay :
! LHSxx,LHSyy: lefthand side matrix of Block Tridiagonal Algebraic Equation
! LHSxx,LHSyy: righthand side vector of Block Tridiagonal Algebraic Equation
! deltaU : changes in solution
! Iden : 3 by 3 identity matrix
! ORfactor : over-relaxation factor
! errors : L2 norm of each component
! errmax : solution update
! conv_cri : tolerance
! xdis,ydis : domain length w and height l

! define identity matrix

Iden(1:3,1:3)=0.0d0
Iden(1,1) =1.0d0
Iden(2,2) =1.0d0
Iden(3,3) =1.0d0

! meshing the domain

! call meshing(xdis,ydis,gridx,gridy,x,y,deltax,deltay)

! Initialize the quantity in each cell (p,u,v) with Sec.1.1 data

do i=0, gridx+1
    do j=0,gridy+1
        p(i,j)=p0*dcos(pi*x(i))*dcos(pi*y(j))
        u(i,j)=u0*dsin(pi*x(i))*dsin(2.d0*pi*y(j))
        v(i,j)=v0*dsin(2.d0*pi*x(i))*dsin(pi*y(j))
    endo
enddo

! set values in ghost cells by boundary conditions

call BC_cavity(p,u,v,gridx,gridy,deltax, deltay)
open(unit=ll,file='TestO.dat',status='unknown')
open(unit=10,file='Error.dat',status='unknown')
errmax=1.0d0
iterno=0

do while ((errmax.ge.conv_cri) .and. (iterno.lt.maxiter))
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 136

! compute Jacobian coefficient

call coeff_matrixx(LHSxA, LHSxB, LHSxC, gridx, gridy, u, v, deltax)
call coeff_matrixy(LHSyA, LHSyB, LHSyC, gridx, gridy, u, v, deltay)

! compute flux integral

call flux_comput(p, u, v, gridx, gridy, deltax, deltay, flcom)
call source(u, v, gridx, gridy, sij)

! compute A*deltax^2*divP

call addedterm(p, gridx, gridy, deltax, deltay, divp)

! Along constant i, solve for deltaU~

do i=1, gridx

! At boundary

call boundary_LHSx(LHSx)
LHSxx(1:3, 1:3, 1:3, 0) = LHSx(1:3, 1:3, 1:3, 1)
LHSxx(1:3, 1:3, 1:3, gridy+1) = LHSx(1:3, 1:3, 1:3, 2)
RHSxx(1:3, 0) = 0.d0
RHSxx(1:3, gridy+1) = 0.d0

! At interior

do j=1, gridy
  do k=1, 3
    RHSxx(k, j) = deltai*(flcom(k, i, j) + divp(k, i, j) + sij(k, i, j))
  end do
  do m=1, 3
    Ax=-LHSyA/deltay; Bx=-LHSyB/deltay; Cx=LHSyC/deltay
    LHSxx(k, m, 1, j) = -deltai*LHSyA(k, m, i, j)/deltay
    LHSxx(k, m, 2, j) = -deltai*LHSyB(k, m, i, j)/deltay + Iden(k, m)
    LHSxx(k, m, 3, j) = deltat*LHSyC(k, m, i, j)/deltay
  end do
end do
end do

call SolveBlockTri(LHSxx, RHSxx, gridy+2, gridy+2)

! prepare to solve deltaU

do j=1, gridy
  do k=1, 3
    
end do
end do
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

\[
\text{deltaU}(k,i,j) = \text{RHSxx}(k,j)
\]
end do
end do
end do

! along constant j

\[
\text{do j=1,gridy}
\]
call boundary_LHSy(LHSy)
LHSyy(1:3,1:3,1:3,0) = LHSy(1:3,1:3,1:3,1)
LHSyy(1:3,1:3,1:3,\text{gridx+1}) = \text{LHSy}(1:3,1:3,1:3,2)
RHSyy(1:3,0) = 0.d0
RHSyy(1:3,\text{gridx+1}) = 0.d0
do i=1,\text{gridx}
do k=1,3
\[
\text{RHSyy}(k,i) = \text{deltaU}(k,i,j)
\]
do m=1,3
\[
\text{LHSyy}(k,m,1,i) = -\text{deltat} \times \text{LHSxA}(k,m,i,j)/\text{deltax}
\]
\[
\text{LHSyy}(k,m,2,i) = -\text{deltat} \times \text{LHSxB}(k,m,i,j)/\text{deltax} + \text{Iden}(k,m)
\]
\[
\text{LHSyy}(k,m,3,i) = \text{deltat} \times \text{LHSxC}(k,m,i,j)/\text{deltax}
\]
end do
end do
end do
call SolveBlockTri(LHSyy, RHSyy, \text{gridx+2}, \text{gridx+2})
do i=1,\text{gridx}
\[
\text{p}(i,j) = \text{p}(i,j) + \text{RHSyy}(1,i) \times \text{ORfactor}
\]
\[
\text{u}(i,j) = \text{u}(i,j) + \text{RHSyy}(2,i) \times \text{ORfactor}
\]
\[
\text{v}(i,j) = \text{v}(i,j) + \text{RHSyy}(3,i) \times \text{ORfactor}
\]
do k=1,3
\[
\text{deltaU}(k,i,j) = \text{RHSyy}(k,i)
\]
end do
end do
end do

! boundary fit

call BC_cavity(p, u, v, gridx, gridy, deltay)
errmax = 0.
call errcomput(deltaU, gridx, gridy, errors, errmax)
iterno = iterno + 1

! output L2 norm versus iteration time steps

write(11,105) iterno, errors(1), errors(2), errors(3)
end do

! call Usymmetry (u, gridx, gridy, Usym)
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

! output
!
!! open(unit=10,file='Error0.dat',status='unknown')
!
! do i=1,gridx
!   do j=1,gridy
!     write(10,100) p(i,j),u(i,j),v(i,j)
!   end do
! end do
!
! write(10,120)(x(i),i=1,gridx)
!
! do j=1,gridy
!   write(10,120)(p(i,j),i=1,gridx)
!   end do
!
! write(10,120)(y(j),j=1,gridy)
! write(10,120)(Usym(j),j=1,gridy)
!
! close(10)
! close(11)
105 format(i6,3e18.8)
100 format(3e18.8)
120 format(1e11.4)
!
stop
end
!
! ----------------------------- END of Main Program -----------------------------
!
! mesh generation
!
subroutine meshing(xdis0, ydis0, gridx0,gridy0, x0,y0,deltax0,deltaxy0)
implicit none
integer i, j, gridx0,gridy0
double precision x0(0:gridx0+l),y0(0:gridy0+l),xdis0,ydis0,deltax0,deltaxy0
!
deltax0=xdis0/dfloat(gridx0)
deltay0=ydis0/dfloat(gridy0)
!
! do i=0,gridx0+1
!   x0(i) = (1.0d0*i-0.5d0)*deltax0
! enddo
!
! do j=0,gridy0+1
!   y0(j) = (1.0d0*j-0.5d0)*deltay0
! enddo
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

return
eend

! ------------------------ END of subroutine meshing ------------------------
! Compute coefficients for Ay,By,Cy

subroutine coeff_matrixy(LHSyA, LHSyB, LHSyC, gridx, gridy, u, v, deltax)
implied none
integer i, j, gridx, gridy
double precision beta, LHSyA(3,3,gridx,gridy), u(0:gridx+1,0:gridy+1), &
v(0:gridx+1,0:gridy+1), Re, deltax, LHSyB(3,3,gridx,gridy), &
LHSyC(3,3,gridx,gridy), Da, epsl, lambda, h, tem, B
parameter (beta=1.d0, Re=5.d0, Da=1.e-3)

tem=dfloat(gridy)

h=int(0.9d0*tem)

do i = 1, gridx
do j=1, gridy
if (j>=h) then
  epsl=.9d0
  lambda=1.d0/epsl
  B=1.d0
else
  epsl=1.d0
  lambda=1.d0
  B=0.d0
endif
LHSyA(1,1,i,j) = 0.d0
LHSyA(1,2,i,j) = 0.d0
LHSyA(1,3,i,j) = 1.d0/2.d0/beta/epsl
LHSyA(2,1,i,j) = 0.d0
LHSyA(2,2,i,j) = 1.d0/4.d0*(v(i,j)-v(i,j-1))/epsl &
  +1.d0/Re/deltay*epsl*lambda
LHSyA(2,3,i,j) = 1.d0/4.d0*(u(i,j)-u(i,j-1))/epsl
LHSyA(3,1,i,j) = 1.d0/2.d0*epsl
LHSyA(3,2,i,j) = 0.d0
LHSyA(3,3,i,j) = 1.d0/2.d0*(v(i,j)+v(i,j-1))/epsl &
  +1.d0/Re/deltay*epsl*lambda
LHSyB(1,1,i,j) = 0.d0
LHSyB(1,2,i,j) = 0.d0
LHSyB(1,3,i,j) = 0.d0
LHSyB(2,1,i,j) = 0.d0
LHSyB(2,2,i,j) = 1.d0/4.d0*(v(i,j)-v(i,j+1))/epsl &
  -2.d0/Re/deltay*epsl*lambda &
  -1.d0/2.d0*(B*(epsl/Da/Re))*deltay
LHSyB(2,3,i,j) = 1.d0/4.d0*(u(i,j)-u(i,j+1))/epsl
LHSyB(3,1,i,j) = 0.d0
LHSyB(3,2,i,j) = 0.d0
LHSyB(3,3,i,j) = 1.d0/2.d0*(v(i,j-1)-v(i,j+1))/eps1 &
-2.d0/Re/deltay*eps1*lambda &
-1.d0/2.d0*(B*(eps1/Da/Re))*deltay

LHSyC(1,1,i,j) = 0.d0
LHSyC(1,2,i,j) = 0.d0
LHSyC(1,3,i,j) = 1.d0/2.d0/beta/eps1
LHSyC(2,1,i,j) = 0.d0
LHSyC(2,2,i,j) = 1.d0/4.d0*(v(i,j)+v(i,j+1))/eps1 &
-1.d0/Re/deltay*eps1/lambda &
LHSyC(2,3,i,j) = 1.d0/4.d0*(u(i,j)+u(i,j+1))/eps1
LHSyC(3,1,i,j) = 1.d0/2.d0*eps1 &
LHSyC(3,2,i,j) = 0.d0 &
LHSyC(3,3,i,j) = 1.d0/2.d0*(v(i,j)+v(i,j+1))/eps1 &
-1.d0/Re/deltay*eps1*lambda

end do
end do
return
end

! END of subroutine coeff_matrixy ---------------------------------

! Compute coefficients for Ax,Bx,Cx

subroutine coeff_matrixx(LHSxA,LHSxB,LHSxC,gridx,gridy,u,v,deltax)
implicit none
integer i,j,gridx, gridy
double precision beta, u(0:gridx+1,0:gridy+1), &
v(0:gridx+1,0:gridy+1), Re, deltax,B, &
LHSxA (3,3,gridx,gridy),LHSxB (3,3,gridx,gridy), &
LHSxC (3,3,gridx,gridy), Da, epsl, lambda,h,tem
parameter (beta=1.d0, Re=5.d0, Da=1.e-3)

tem=dfloat(gridy)
h=int(0.9d0*tem)
do i = 1, gridx
  do j=1, gridy
    if (j>=h) then
      epsl=.9d0
      B=1.d0
      lambda=1.d0/eps1
    else
      B=0.d0
      epsl=1.d0
      lambda=1.d0
    endif
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 141

LHSxA(1,1,i,j) = 0.d0
LHSxA(1,2,i,j) = 1.d0/2.d0/beta/epsl
LHSxA(1,3,i,j) = 0.d0
LHSxA(2,1,i,j) = 1.d0/2.d0*epsl
LHSxA(2,2,i,j) = 1.d0/2.d0*(u(i-1,j)+u(i-1,j))/epsl & 
    +1.d0/Re/deltax*epsl*lambda 
LHSxA(2,3,i,j) = 0.d0
LHSxA(3,1,i,j) = 0.d0
LHSxA(3,2,i,j) = 1.d0/4.d0*(v(i-1,j)+v(i,j))/epsl & LHSxA(3,3,i,j) = 1.d0/4.d0*(u(i,j)+u(i,j))/epsl 
    +1.d0/Re/deltax*epsl*lambda

LHSxB(1,1,i,j) = 0.d0
LHSxB(1,2,i,j) = 0.d0
LHSxB(1,3,i,j) = 0.d0
LHSxB(2,1,i,j) = 0.d0
LHSxB(2,2,i,j) = 1.d0/2.d0*(u(i-1,j)-u(i+1,j))/epsl & 
    -2.d0/Re/deltax*epsl*lambda 
LHSxB(2,3,i,j) = 0.d0
LHSxB(3,1,i,j) = 0.d0
LHSxB(3,2,i,j) = 1.d0/4.d0*(v(i-1,j)-v(i+1,j))/epsl & LHSxB(3,3,i,j) = 1.d0/4.d0*(u(i-1,j)+u(i+1,j))/epsl 
    -2.d0/Re/deltax*epsl*lambda 

LHSxC(1,1,i,j) = 0.d0
LHSxC(1,2,i,j) = 1.d0/2.d0/beta/epsl
LHSxC(1,3,i,j) = 0.d0
LHSxC(2,1,i,j) = 1.d0/2.d0*epsl
LHSxC(2,2,i,j) = 1.d0/2.d0*(u(i,j)+u(i+1,j))/epsl & 
    -1.d0/Re/deltax*epsl*lambda 
LHSxC(2,3,i,j) = 0.d0
LHSxC(3,1,i,j) = 0.d0
LHSxC(3,2,i,j) = 1.d0/4.d0*(v(i,j)+v(i+1,j))/epsl & LHSxC(3,3,i,j) = 1.d0/4.d0*(u(i,j)+u(i+1,j))/epsl 
    -1.d0/Re/deltax*epsl*lambda

enddo
end do
return 
end

END of subroutine coeff_matrixx

! compute flux integral

subroutine flux_comput (pp,uu,vv,gridx0, gridy0,deltax, deltax, flcom0)
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

implicit none
integer k, i, j, gridx0, gridy0, ip, im, jp, jm

double precision flcom0(3,gridx0, gridy0), pp(0:gridx0+1,0:gridy0+1), &
uu(0:gridx0+1,0:gridy0+1), vv(0:gridx0+1,0:gridy0+1), &
Re, beta, deltax, deltay, Da, epsl, lambda, h, tem

parameter (Re=5.d0, beta=1.d0, Da=1.e-3)

tem=dfloat(gridy0)
h=int(0.9d0*tem)

do k=1,3
  do i=1, gridx0
    ip=i+1; im=i-1
    do j=1, gridy0
      jp=j+1; jm=j-1
      if (j>=h) then
        epsl=.9d0
        lambda=1.d0/epsl
      else
        epsl=1.d0
        lambda=1.d0
      endif

      if (k==1) then
        flcom0(k,i,j)=-(uu(ip,j)-uu(im,j))/2.d0/beta/deltax/epsl &
        -(vv(i,jp)-vv(i,jm))/beta/deltay/2.d0/epsl &
      elseif (k==2) then
        flcom0(k,i,j)=-(uu(i+1,j)+uu(i,j))**2/4.d0/epsl &
        -(uu(i,j)+uu(i-1,j))**2/4.d0/epsl &
        +(pp(i+1,j)-pp(i-1,j))/2.d0*epsl &
        -(uu(i+1,j)-2.0d0*uu(i,j)+uu(i-1,j)) &
        /(Re*deltax)*epsl*lambda)/deltax &
        -((uu(i,j+1)+uu(i,j))*(vv(i,j+1) &
        +vv(i,j))/4.d0/epsl &
        -(uu(i,j-1)+uu(i,j))*(vv(i,j-1) &
        +vv(i,j))/4.d0/epsl &
        -(uu(i,j+1)-2.d0*uu(i,j)+uu(i,j-1)) &
        /(Re*deltay)*epsl*lambda)/deltay &
      else
        flcom0(k,i,j)=-(uu(ip,j)+uu(ip,j))/2.d0*(vv(ip,j) &
        +vv(i,j))/2.d0/epsl-(uu(im,j)+uu(i,j))/2.d0*(vv(im,j) &
        +vv(i,j))/2.d0/epsl-(vv(ip,j)-2.d0*vv(i,j)+vv(ip,j)) &
        /(Re*deltax)*epsl*lambda)/deltax &
        -((vv(i,jp)+vv(i,jm))*2/4.d0/epsl &
        -(vv(i,j)+vv(i,jm))*2/4.d0/epsl &
        +(pp(ip,j)-pp(ip,jm))/2.d0*epsl &
      endif
  enddo
enddo
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

```
-(vv(i,jp)-2.d0*vv(i,j)+vv(i,jm)) &
/(Re*deltay)*epsl*lambda)/deltay
endif
enddo
enddo
return
end

END of subroutine flux_comput

subroutine source(uu,vv,gridxO, gridyO,sij)
implicit none
integer k, i, j, gridxO, gridyO,h

double precision sij(3,gridxO, gridyO),
uu(0:gridxO+1,0:gridyO+1),vv(0:gridxO+1,0:gridyO+1),
&
Re, B, epsl, Da,tem
parameter (Re=5.d0, Da=l.e-3)

tem=dfloat(gridyO)
h=int(0.9d0*tem)
do k=1,3 
do i=1, gridxO
  do j=1, gridyO
    if (j>=h) then
      B=l.d0
      epsl=.9d0
    else
      B=0.d0
      epsl=1.d0
    endif
    if (k==1) then
      sij(k,i,j)=0.d0
    elseif (k==2) then
      sij(k,i,j)=-B*epsl/Da/Re*uu(i,j)
    else
      sij(k,i,j)=-B*epsl/Da/Re*vv(i,j)
    endif
  enddo
enddo
return
end
```

! --------- END of subroutine source ---------

subroutine errcomput(deltaU, gridx, gridy, errors, errmax)
implicit none
integer m, i, j, gridx, gridy
double precision deltaU(3, gridx, gridy), errors(3), errmax

do m=1,3
   errors(m)=0.
   do i=1, gridx
      do j=1, gridy
         errors(m) = errors(m) + deltaU(m, i, j)**2.d0
      end do
   end do
   errors(m)=dsqrt(errors(m)/dfloat(gridx)/dfloat(gridy))
end do
errmax=errors(1)
! errmax=errors(2)
do m=2,3
   if (errors(m) .gt. errmax) then
      errmax=errors(m)
   end if
end do
return
end

I END of subroutine errcomput -----------------

! set ghost cell values

subroutine BC_cavity(p, u, v, gridx, gridy, deltax, deltay)
implicit none
integer i, j, gridx, gridy
double precision p(0:gridx+l,0:gridy+l), u(0:gridx+1,0:gridy+1), &
v(0:gridx+l,0:gridy+l), deltax, deltay, Uin, &
Vw, Rat, ki, Re, Da, tem, h
parameter(Vw=1.d0, Re=5.d0, Da=1.e-3)

Uin=21.d0
Rat=Uin/Vw
ki=100.d0
tem=dfloat(gridy)
h=int(0.9d0*tem)
do i=1, gridx
   p(i, gridy+1)= p(i, gridy)-deltay/Re/Da ! at bottom wall, suction
   u(i,0)= u(i,1)
   v(i,gridy+1)=-v(i,gridy)

!
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 145

\[
\begin{align*}
  p(i,0) &= p(i,1) \quad \text{at upper wall} \\
  u(i,0) &= -u(i,1) \\
  v(i,0) &= -v(i,1)
\end{align*}
\]

end do

do \(j=1\), gridy

\[
\begin{align*}
  p(0,j) &= p(1,j) - \text{deltax} \times k_i \quad \text{at left} \\
  u(0,j) &= -u(1,j) \\
  v(0,j) &= -v(1,j)
\end{align*}
\]

end do

return
end

! END of subroutine BC_cavity

! Suppression for pressure oscillation, add a source term routine

subroutine addedterm(p, gridx, gridy, deltax, deltay, divp)
  implicit none
  integer i, j, gridx, gridy
  double precision divp(3, gridx, gridy), p(0:gridx+1, 0:gridy+1), &
  deltax, deltay, A
  A = 0.5d0
  divp(1:3, 1:gridx, 1:gridy) = 0.
  do i=1, gridx
    do j=1, gridy

\[
\begin{align*}
  p(0,j) &= p(1,j) - \text{deltax} \times k_i \quad \text{at left} \\
  p(0,j) &= p(1,j) \\
  \text{if (} j \geq 1 \text{)} p(0,j) &= -p(1,j) \quad \text{fixed pressure at one point} \\
  \text{if (} j \geq h \text{)} \text{ then} \\
  \text{u}(0,j) &= -u(1,j) \\
  \text{else} \\
  \text{u}(0,j) &= -u(1,j) + R_t \times 2.0 \\
  \text{endif} \\
  \text{u}(0,j) &= -u(1,j) \\
  \text{v}(0,j) &= -v(1,j)
\end{align*}
\]

end do

\[
\begin{align*}
  p(0,j) &= -p(1,j) + 0.08d0 \times 2.0 \quad \text{at right wall, fully developed} \\
  u(0,j) &= u(1,j) \\
  v(0,j) &= v(1,j)
\end{align*}
\]

end do

return
end
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 146

\[
\text{divp}(1,i,j) = A \cdot \text{deltax} \cdot \text{deltax} \cdot (1.0 / 2.0 \cdot (p(i+1,j) - 2.0 \cdot p(i,j) + p(i-1,j)) / \text{deltax} / \text{deltax} \\
+ 1.0 / 2.0 \cdot (p(i,j+1) - 2.0 \cdot p(i,j) + p(i,j-1)) / \text{deltay} / \text{deltay})
\]

end do
end do
return
end

I END of subroutine addedterm

! Sanity check for symmetry

subroutine Usymmetry (u, gridx, gridy, u_sym)
implicit none
integer j, m, gridx, gridy
double precision u(0:gridx+l,0:gridy+1), u_sym(gridy)
m=gridx/2
do j=1,gridy
    u_sym(j)=(u(m,j)+u(m+1,j))*0.5d0
end do
return
end

! ------------------ END of subroutine Usymmetry ------------------

! Ax,Bx,Cx and Ay,By,Cy at boundary

subroutine boundary_LHSy(LHSy)
implicit none
double precision LHSy(3,3,3,2)
LHSy(1,1,1,1) = 0.d0
LHSy(1,2,1,1) = 0.d0
LHSy(1,3,1,1) = 0.d0
LHSy(2,1,1,1) = 0.d0
LHSy(2,2,1,1) = 0.d0
LHSy(2,3,1,1) = 0.d0
LHSy(3,1,1,1) = 0.d0
LHSy(3,2,1,1) = 0.d0
LHSy(3,3,1,1) = 0.d0
LHSy(1,1,2,1) = 1.d0
LHSy(1,2,2,1) = 0.d0
LHSy(1,3,2,1) = 0.d0
LHSy(2,1,2,1) = 0.d0
LHSy(2,2,2,1) = 1.d0
LHSy(2,3,2,1) = 0.d0
LHSy(3,1,2,1) = 0.d0
LHSy(3,2,2,1) = 0.d0
LHSy(3,3,2,1) = 1.d0
LHSy(1,1,3,1) =-1.d0
LHSy(1,2,3,1) = 0.d0
LHSy(1,3,3,1) = 0.d0
LHSy(2,1,3,1) = 0.d0
LHSy(2,2,3,1) = 1.d0
LHSy(2,3,3,1) = 0.d0
LHSy(3,1,3,1) = 0.d0
LHSy(3,2,3,1) = 0.d0
LHSy(3,3,3,1) = 1.d0
LHSy(1,1,1,2) = 1.d0
LHSy(1,2,1,2) = 0.d0
LHSy(1,3,1,2) = 0.d0
LHSy(2,1,1,2) = 0.d0
LHSy(2,2,1,2) =-1.d0
LHSy(2,3,1,2) = 0.d0
LHSy(3,1,1,2) = 0.d0
LHSy(3,2,1,2) = 0.d0
LHSy(3,3,1,2) =-1.d0
LHSy(1,1,2,2) = 1.d0
LHSy(1,2,2,2) = 0.d0
LHSy(1,3,2,2) = 0.d0
LHSy(2,1,2,2) = 0.d0
LHSy(2,2,2,2) = 1.d0
LHSy(2,3,2,2) = 0.d0
LHSy(3,1,2,2) = 0.d0
LHSy(3,2,2,2) = 0.d0
LHSy(3,3,2,2) = 1.d0
LHSy(1,1,3,2) = 0.d0
LHSy(1,2,3,2) = 0.d0
LHSy(1,3,3,2) = 0.d0
LHSy(2,1,3,2) = 0.d0
LHSy(2,2,3,2) = 0.d0
LHSy(2,3,3,2) = 0.d0
LHSy(3,1,3,2) = 0.d0
LHSy(3,2,3,2) = 0.d0
LHSy(3,3,3,2) = 0.d0

return
end
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM 148

END of subroutine boundary_LHSy

subroutine boundary_LHSx(LHSy)
implicit none
double precision LHSy (3,3,3,2)

LHSy (1,1,1,1) = 0.d0
LHSy (1,2,1,1) = 0.d0
LHSy (1,3,1,1) = 0.d0
LHSy (2,1,1,1) = 0.d0
LHSy (2,2,1,1) = 0.d0
LHSy (2,3,1,1) = 0.d0
LHSy (3,1,1,1) = 0.d0
LHSy (3,2,1,1) = 0.d0
LHSy (3,3,1,1) = 0.d0

LHSy (1,1,2,1) = 1.d0
LHSy (1,2,2,1) = 0.d0
LHSy (1,3,2,1) = 0.d0
LHSy (2,1,2,1) = 0.d0
LHSy (2,2,2,1) = 1.d0
LHSy (2,3,2,1) = 0.d0
LHSy (3,1,2,1) = 0.d0
LHSy (3,2,2,1) = 0.d0
LHSy (3,3,2,1) = 1.d0

LHSy (1,1,3,1) = -1.d0
LHSy (1,2,3,1) = 0.d0
LHSy (1,3,3,1) = 0.d0
LHSy (2,1,3,1) = 0.d0
LHSy (2,2,3,1) = 1.d0
LHSy (2,3,3,1) = 0.d0
LHSy (3,1,3,1) = 0.d0
LHSy (3,2,3,1) = 0.d0
LHSy (3,3,3,1) = 1.d0

LHSy (1,1,1,2) = -1.d0
LHSy (1,2,1,2) = 0.d0
LHSy (1,3,1,2) = 0.d0
LHSy (2,1,1,2) = 0.d0
LHSy (2,2,1,2) = 1.d0
LHSy (2,3,1,2) = 0.d0
LHSy (3,1,1,2) = 0.d0
LHSy (3,2,1,2) = 0.d0
LHSy (3,3,1,2) = 1.d0

LHSy (1,1,2,2) = 1.d0
LHSy (1,2,2,2) = 0.d0
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

LHSy(1,3,2,2) = 0.d0
LHSy(2,1,2,2) = 0.d0
LHSy(2,2,2,2) = 1.d0
LHSy(2,3,2,2) = 0.d0
LHSy(3,1,2,2) = 0.d0
LHSy(3,2,2,2) = 0.d0
LHSy(3,3,2,2) = 1.d0

LHSy(1,1,3,2) = 0.d0
LHSy(1,2,3,2) = 0.d0
LHSy(1,3,3,2) = 0.d0
LHSy(2,1,3,2) = 0.d0
LHSy(2,2,3,2) = 0.d0
LHSy(2,3,3,2) = 0.d0
LHSy(3,1,3,2) = 0.d0
LHSy(3,2,3,2) = 0.d0
LHSy(3,3,3,2) = 0.d0

return
end
! ---------------------- END of subroutine boundary_LHSx ----------------------

! following subroutines are for Tomas Block triangle algorithm

subroutine SolveBlockTri(LHS, RHS, NRows, MaxSize)
  double precision LHS(3, 3, 3, MaxSize), RHS(3, MaxSize)
  integer NRows, MaxSize

  integer j
  double precision Inv(3,3), TempMat(3,3), TempVec(3)
  double precision TempMat2(3,3), TVec2(3,3)

  do 10 j = 1, NRows-1
  ! Compute the inverse of the main block diagonal.
    call Invert3x3(LHS(l,1,2,j), Inv)
  ! Scale the right-most block diagonal by the inverse.
    call Mult3x3(Inv, LHS(1,1,3,j), TempMat)
    call Copy3x3(TempMat, LHS(1,1,3,j))
  ! Scale the right-hand side by the inverse.
    call MultVec(Inv, RHS(l,j), TempVec)
    call CopyVec(TempVec, RHS(l,j))
  ! Left-multiply the jth row by the sub-diagonal on the j+1st row and
  ! subtract from the j+1st row. This involves the super-diagonal
  ! term and the RHS of the jth row.
  ! First the LHS manipulation
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

```fortran
! Now the RHS manipulation
call MultVec(LHS(1,1,1,j+1), RHS(1,j), TempVec)
call AddVec(RHS(1,j+1), TempVec, -1.d0, TVec2)
call CopyVec(TVec2, RHS(1,j+1))
end do

! Done with forward elimination loop

! Compute the inverse of the last main block diagonal.
! j = NRows

call Invert3x3(LHS(1,1,2,j), Inv)

! Scale the right-hand side by the inverse.

call MultVec(Inv, RHS(1,j), TempVec)
call CopyVec(TempVec, RHS(1,j))

! Now do the back-substitution.
do 20 j = NRows - 1, 1, -1
! Matrix-vector multiply and subtract.
RHS(1,j) = RHS(1,j) - (LHS(1,1,3,j)*RHS(1,j+1) + 
                  LHS(1,2,3,j)*RHS(2,j+1) + 
                  LHS(1,3,3,j)*RHS(3,j+1))
RHS(2,j) = RHS(2,j) - (LHS(2,1,3,j)*RHS(1,j+1) + 
                  LHS(2,2,3,j)*RHS(2,j+1) + 
                  LHS(2,3,3,j)*RHS(3,j+1))
RHS(3,j) = RHS(3,j) - (LHS(3,1,3,j)*RHS(1,j+1) + 
                  LHS(3,2,3,j)*RHS(2,j+1) + 
                  LHS(3,3,3,j)*RHS(3,j+1))
20 end do
return
end

subroutine SpewMatrix(Source)
double precision Source(3,3)
write(6,*) Source(1,1), Source(2,1), Source(3,1)
write(6,*) Source(1,2), Source(2,2), Source(3,2)
write(6,*) Source(1,3), Source(2,3), Source(3,3)
return
end

subroutine SpewVector(Source)
double precision Source(3)
write(6,*) Source(1), Source(2), Source(3)
return
end

subroutine CopyVec(Source, Target)
double precision Source(3), Target(3)
Target(1) = Source(1)
Target(2) = Source(2)
Target(3) = Source(3)
return
end

subroutine Copy3x3(Source, Target)
double precision Source(3,3), Target(3,3)
Target(1,1) = Source(1,1)
Target(1,2) = Source(1,2)
Target(1,3) = Source(1,3)
Target(2,1) = Source(2,1)
Target(2,2) = Source(2,2)
Target(2,3) = Source(2,3)
Target(3,1) = Source(3,1)
Target(3,2) = Source(3,2)
Target(3,3) = Source(3,3)
return
end

subroutine MultVec(A, Vec, Result0)
double precision A(3,3), Vec(3), Result0(3)
Result0(1) = A(1,1)*Vec(1) + A(1,2)*Vec(2) + A(1,3)*Vec(3)
Result0(2) = A(2,1)*Vec(1) + A(2,2)*Vec(2) + A(2,3)*Vec(3)
Result0(3) = A(3,1)*Vec(1) + A(3,2)*Vec(2) + A(3,3)*Vec(3)
return
end

subroutine Mult3x3(A, B, C)
double precision A(3,3), B(3,3), C(3,3)
C(1,1) = A(1,1)*B(1,1) + A(1,2)*B(2,1) + A(1,3)*B(3,1)
C(1,2) = A(1,1)*B(1,2) + A(1,2)*B(2,2) + A(1,3)*B(3,2)
C(1,3) = A(1,1)*B(1,3) + A(1,2)*B(2,3) + A(1,3)*B(3,3)
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

\[
C(2,1) = A(2,1)B(1,1) + A(2,2)B(2,1) + A(2,3)B(3,1) \\
C(2,2) = A(2,1)B(1,2) + A(2,2)B(2,2) + A(2,3)B(3,2) \\
C(2,3) = A(2,1)B(1,3) + A(2,2)B(2,3) + A(2,3)B(3,3)
\]

\[
C(3,1) = A(3,1)B(1,1) + A(3,2)B(2,1) + A(3,3)B(3,1) \\
C(3,2) = A(3,1)B(1,2) + A(3,2)B(2,2) + A(3,3)B(3,2) \\
C(3,3) = A(3,1)B(1,3) + A(3,2)B(2,3) + A(3,3)B(3,3)
\]

return
end subroutine Add3x3(A, B, Factor, C)

double precision A(3,3), B(3,3), Factor, C(3,3)

C(1,1) = A(1,1) + Factor * B(1,1) \\
C(1,2) = A(1,2) + Factor * B(1,2) \\
C(1,3) = A(1,3) + Factor * B(1,3) \\

C(2,1) = A(2,1) + Factor * B(2,1) \\
C(2,2) = A(2,2) + Factor * B(2,2) \\
C(2,3) = A(2,3) + Factor * B(2,3) \\

C(3,1) = A(3,1) + Factor * B(3,1) \\
C(3,2) = A(3,2) + Factor * B(3,2) \\
C(3,3) = A(3,3) + Factor * B(3,3)
return
end subroutine AddVec(A, B, Factor, C)

double precision A(3), B(3), Factor, C(3)

C(1) = A(1) + Factor * B(1) \\
C(2) = A(2) + Factor * B(2) \\
C(3) = A(3) + Factor * B(3)
return
end subroutine Invert3x3(Block, Inverse)

double precision Block(3,3), Inverse(3,3)

double precision DetInv

\[
DetInv = 1.0 / (+ Block(1,1)*Block(2,2)*Block(3,3) &
+ Block(1,2)*Block(2,3)*Block(3,1) &
+ Block(1,3)*Block(2,1)*Block(3,2) &
- Block(1,3)*Block(2,2)*Block(3,1) &
- Block(1,2)*Block(2,1)*Block(3,3) &
- Block(1,1)*Block(2,3)*Block(3,2))
\]
APPENDIX E. THE ALGORITHM AND CODE FOR ONE DOMAIN PROBLEM

! Expand by minors to compute the inverse
Inverse(1,1) = + DetInv * (Block(2,2)*Block(3,3) -
    Block(3,2)*Block(2,3))
Inverse(2,1) = - DetInv * (Block(2,1)*Block(3,3) -
    Block(3,1)*Block(2,3))
Inverse(3,1) = + DetInv * (Block(2,1)*Block(3,2) -
    Block(3,1)*Block(2,2))
Inverse(1,2) = - DetInv * (Block(1,2)*Block(3,3) -
    Block(3,2)*Block(1,3))
Inverse(2,2) = + DetInv * (Block(1,1)*Block(3,3) -
    Block(3,1)*Block(1,3))
Inverse(3,2) = - DetInv * (Block(1,1)*Block(3,2) -
    Block(3,1)*Block(1,2))
Inverse(1,3) = + DetInv * (Block(1,2)*Block(2,3) -
    Block(2,2)*Block(1,3))
Inverse(2,3) = - DetInv * (Block(1,1)*Block(2,3) -
    Block(2,1)*Block(1,3))
Inverse(3,3) = + DetInv * (Block(1,1)*Block(2,2) -
    Block(2,1)*Block(1,2))

return
end

! ---------------- END of subroutine SolveBlockTri ----------------
Appendix F

DETAILED DRAWINGS OF THE APPARATUS

F.1 Detailed Drawing of the Apparatus
Figure F.1: the 3-D view of the cover of the upstream-tank
Figure F.2: the dimensions of cover of the upstream-tank
Figure F.3: the 3-D view of the lid of the test channel
Figure F.4: the dimensions of the lid of the test channel
Figure F.5: the 3D view of the screen plate (filter) for the suction
Figure F.6: the dimensions of the screen plate for uniform suction.
Figure F.7: the 3-D view of the suction element (diffusor) of the device
Figure F.8: the dimensions of the suction element (diffusor)
Figure F.9: the 3D view of the upstream and downstream tanks connected with a channel
Figure F.10: the dimensions of the upstream and downstream tanks connected with a channel
F.2 Description of the Pump

The centrifugal pump we used is an all bronze pump purchased from Pricepump (http://www.pricepump.com), the following is the brief description:

- Model CD150-AB-362-6A111-100-36-1T6
- 1 horse power, motor speed 3600 RPM
- the range of flow rate is 25 to 160 GPM
- the optimum working region is: 75-120 GPM at the pressure head of 30-50 feet

F.3 Seeding particles

The seeding particles are the polymeric particles produced by EMS-CHEMIE AG in Switzerland (http://www.ems-group.com/en/index1.html). The product is named Griltex 2A P1, whose average particle size is around 60 microns with a density a little bit above that of water (1.04) but lower than our fluid (1.085).

It has to be noted that before adding the particles into the fluid, a little bit of soap was added to provide reasonable wetting of the particles, together with particles into a beaker of water and to thoroughly mix before adding to the reservoir. The amount of the particles to be provided has to be sufficient to get enough echo at medium emitting power.