DEFECT CORRECTION BASED DOMAIN DECOMPOSITION METHODS FOR SOME NONLINEAR PROBLEMS

ANTONY SIAHAAN

A thesis submitted in partial fulfilment of the requirements of the University of Greenwich for the degree of Doctor of Philosophy
JUNE 2011

DECLARATION

I certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of Doctor of Philisophy (PhD) being studied at the University of Greenwich. I also declare that this work is the result of my own investigations except where otherwise identified by references and that I have not plagiarised the work of others.

Student :

Antony Siahaan

First Supervisor :

Prof. Choi-Hong Lai

Second Supervisor :

Prof. Koulis Pericleous

Acknowledgments

I would like to thank my supervisors Prof. Choi-Hong Lai and Prof. Koulis Pericleous for their help. I would also thank Dr. Mayur Patel for providing me with the details of the PHOENICS code.

I would also express my gratitude to the University of Greenwich for the financial support and the opportunity to conduct this research.

Final thanks go to my family for their encouragement and also those who pray for me in all my joy and trouble.

Abstract

Defect correction schemes as a class of nonoverlapping domain decomposition methods offer several advantages in the ways they split a complex problem into several subdomain problems with less complexity. The schemes need a nonlinear solver to take care of the residual at the interface. The adaptive-α solver can converge locally in the ∞-norm, where the sufficient condition requires a relatively small local neighbourhood and the problem must have a strongly diagonal dominant Jacobian matrix with a very small condition number. Yet its advantage can be of high significance in the computational cost where it simply needs a scalar as the approximation of Jacobian matrix. Other nonlinear solvers employed for the schemes are a Newton-GMRES method, a Newton method with a finite difference Jacobian approximation, and nonlinear conjugate gradient solvers with Fletcher-Reeves and Pollak-Ribiere searching direction formulas.

The schemes are applied to three nonlinear problems. The first problem is a heat conduction in a multichip module where there the domain is assembled from many components of different conductivities and physical sizes. Here the implementations of the schemes satisfy the component meshing and gluing concept. A finite difference approximation of the residual of the governing equation turns out to be a better defect equation than the equality of normal derivative. Of all the nonlinear solvers implemented in the defect correction scheme, the nonlinear conjugate gradient method with Fletcher-Reeves searching direction has the best performance.

The second problem is a 2D single-phase fluid flow with heat transfer where the PHOENICS CFD code is used to run the subdomain computation. The Newton method with a finite difference Jacobian is a reasonable interface solver in coupling these subdomain computations. The final problem is a multiphase heat and moisture transfer in a porous textile. The PHOENICS code is also used to solve the system of partial differential equations governing the multiphase process in each subdomain while the coupling of the subdomain solutions is taken care of with some FORTRAN codes by the defect correction schemes. A scheme using a modified-α method fails to obtain decent solutions in both single and two layers case. On the other hand, the scheme using the above Newton method produces satisfying results for both cases where it can lead an initially distant interface data into a good convergent solution. However, it is found that in general the number of nonlinear iteration of the defect correction schemes increases with the mesh refinement.

Contents

Declaration **i**

Acknowledgements **ii**

Abstract **iii**

Contents **v**

List of Figures **viii**

List of Tables **xiii**

Glossary **xiv**

1 Introduction **1**

1.1 Geometrical and Physical Decomposition 6

1.2 Nonoverlapping Domain Decomposition 8

1.3 Parallel Implementation 9

1.4 Research Objectives and Proposed Methods 12

1.5 Thesis Outline 13

2 A Defect Correction Scheme **15**

2.1 A Nonoverlapping Scheme 16

2.2 Defect Equations 17

2.3 A Normal Derivative Approximation For Defect Equations 20

2.4 Nonlinear Solvers 23

2.4.1 Adaptive α method 24

2.4.2 A Newton Method with Finite Difference Jacobian 33
2.4.3 Newton-GMRES method 35
2.4.4 Nonlinear Conjugate Gradient 37

3 Nonoverlapping Domain Decomposition Methods 38

3.1 Schur Complements 38
3.2 Classical Iterative Substructuring Methods 40
3.2.1 The Dirichlet-Neumann Method 40
3.2.2 The Neumann-Neumann Method 43
3.3 Parallel-Motivated Preconditioners 45
3.4 Nonconforming methods 47
3.5 Closure 52

4 A Nonlinear Heat Transfer Problem 53

4.1 A multi-chip electronic module 54
4.2 Numerical Schemes 55
4.2.1 Linearisation Method 57
4.2.2 Conjugate Gradient Solver 58
4.2.3 Preconditioned Conjugate Gradient Method 59
4.2.4 SSOR Preconditioner 60
4.3 Numerical Results 60
4.3.1 Comparison of Defect Functions 62
4.3.2 Comparison of Nonlinear Solvers 64
4.4 Closure 65

5 Some Fluid Flow and Heat Transfer Problems Using the PHOENICS CFD Code 67

5.1 Computational Procedures in PHOENICS 67
5.1.1 Finite Volume Discretisation 68
5.1.2 Discretisation of the Momentum Equations 72
5.1.3 Pressure Correction Equation 76
5.1.4 SIMPLE Algorithm 78
5.1.5 TDMA 78
5.1.6 Stone method 80

5.2 A 2D Single-Phase Test Problem 82
5.3 A Multiphase Flow Problem with Heat and Moisture Transfer in Porous Textile Materials 87
5.3.1 Introduction 87
5.3.2 A Multiphase Process in Fabric 89
5.3.3 Problem Description 91
5.3.4 Single layer of fabric 94
5.3.5 Two layers of fabric 98
5.4 Closure 101

6 Defect Correction Scheme in a CFD Code 105
6.1 PHOENICS setting 106
6.2 2D Single Phase Test Problem 106
6.3 Multiphase problem 111
6.3.1 A single layer of fabric 117
6.3.2 Two layers of fabric 133
6.4 Closure 144

7 Conclusions and Further Works 150

Bibliography 153

List of Figures

1.1 Decomposition into two subdomains 4
1.2 Nonoverlapping decomposition into two subdomains 4
1.3 Component-based decomposition into a PCB and a chip 6
1.4 Decomposition into Darcy and Stokes region 7
1.5 Physics and geometrical based decomposition in magnetohydrodynamics . . 7

2.1 Vertical line perpendicular to interface line 20
2.2 Discretisation in orthogonal grid 21
2.3 Subdivision in orthogonal grid 22

4.1 Multi-chip module of five different layers 54
4.2 Five points stencil at the interface 56
4.3 Temperature Contour in the Multi-Chip 61

5.1 Control volume in two-dimensions 71
5.2 Placement of variables in the PHOENICS grid setting 73
5.3 Control volume for u 73
5.4 Control volume for v 74
5.5 Control volume for the continuity equation 76
5.6 Domain of 2D problem with fluid flow and heat transfer 83
5.7 Scale of the 2D problem 83
5.8 Contour of pressure 85
5.9 Contour of temperature 86
5.10 A cell of fabric 89
5.11 Condition at the left and right boundary 93
5.12 Pressure of gas 95

5.13 Density of gas 96
5.14 Evaporation 97
5.15 Volume fraction of gas 97
5.16 Velocity of gas 98
5.17 Peclet number 99
5.18 Temperature of gas 99
5.19 Concentration of vapour 100
5.20 vapour content in gas 100
5.21 Two layers of fabric 101
5.22 Pressure of gas in two layers of fabric 102
5.23 Velocity of gas in two layers of fabric 102
5.24 Volume fraction of gas in two layers of fabric 103
5.25 Concentration of vapour in two layers of fabric 103
5.26 vapour content in gas in two layers of fabric 104

6.1 Domain and mesh decomposition 107
6.2 A cell in the interface layer 108
6.3 Domain of fabric 111
6.4 A mesh of one single domain in PHOENICS 112
6.5 Interface line 112
6.6 Decomposition in multiphase process 113
6.7 Boundary condition of Subdomain-1 113
6.8 Boundary condition of Subdomain-2 114
6.9 Interface Cell 115
6.10 Iteration of interface pressure - single layer problem (Newton method) 119
6.11 Iteration of interface liquid volume fraction - single layer problem (Newton method) 119
6.12 Iteration of interface gas enthalpy - single layer problem (Newton method) . 120
6.13 Iteration of interface liquid enthalpy - single layer problem (Newton method) 120
6.14 Iteration of interface vapor content in gas - single layer problem (Newton method) 121
6.15 Normalized total mass residual at interface - single layer problem (Newton method) 122

6.16 Normalized liquid mass residual at interface - single layer problem (Newton method) 122
6.17 Normalized gas enthalpy residual at interface - single layer problem (Newton method) 123
6.18 Normalized liquid enthalpy residual at interface - single layer problem (Newton method) 123
6.19 Normalized vapor mass residual at interface - single layer problem (Newton method) 124
6.20 Comparison of pressure between domain decomposition and reference solution 125
6.21 Difference of pressure between domain decomposition and reference solution 125
6.22 Comparison of liquid volume fraction between domain decomposition and reference solution 126
6.23 Difference of liquid volume fraction between domain decomposition and reference solution 126
6.24 Comparison of gas enthalpy between domain decomposition and reference solution 127
6.25 Difference of gas enthalpy between domain decomposition and reference solution 127
6.26 Comparison of liquid enthalpy between domain decomposition and reference solution 128
6.27 Difference of liquid enthalpy between domain decomposition and reference solution 128
6.28 Comparison of vapor content (in gas) between domain decomposition and reference solution 129
6.29 Difference of vapor content (in gas) between domain decomposition and reference solution 129
6.30 Iteration of interface pressure - single layer problem (Modified α method) . . 131
6.31 Iteration of liquid volume fraction at interface - single layer problem (Modified α method) 131
6.32 Iteration of interface gas enthalpy - single layer problem (Modified α method) 132
6.33 Iteration of interface liquid enthalpy - single layer problem (Modified α method) 132
6.34 Iteration of interface vapor content in gas - single layer problem (Modified α method) 133

6.35 Normalized total mass residual at interface - single layer problem (Modified α method) . . . 134
6.36 Normalized liquid mass residual at interface - single layer problem (Modified α method) . . . 134
6.37 Normalized gas enthalpy residual at interface - single layer problem (Modified α method) . . . 135
6.38 Normalized liquid enthalpy residual at interface - single layer problem (Modified α method) . . . 135
6.39 Normalized vapor mass residual at interface - single layer problem (Modified α method) . . . 136
6.40 Iteration of interface pressure - two layers problem (modified α method) . . . 137
6.41 Normalized total mass residual at interface - two layers problem (modified α method) . . . 138
6.42 Iteration of interface pressure - two layers problem (Newton method) . . . 139
6.43 Iteration of interface liquid volume fraction - two layers problem (Newton method) . . . 139
6.44 Iteration of interface gas enthalpy - two layers problem (Newton method) . . . 140
6.45 Iteration of interface liquid enthalpy - two layers problem (Newton method) . . . 140
6.46 Iteration of interface vapor content in gas - two layers problem (Newton method) 141
6.47 Normalized total mass residual at interface - two layers problem (Newton method) . . . 141
6.48 Normalized liquid mass residual at interface - two layers problem (Newton method) . . . 142
6.49 Normalized gas enthalpy residual at interface - two layers problem (Newton method) . . . 142
6.50 Normalized liquid enthalpy residual at interface - two layers problem (Newton method) . . . 143
6.51 Normalized vapor mass residual at interface - two layers problem (Newton method) . . . 143
6.52 Comparison of pressure between domain decomposition and reference solution (two layers) . . . 144
6.53 Difference of pressure between domain decomposition and reference solution (two layers) . . . 145

6.54 Comparison of liquid volume fraction between domain decomposition and reference solution (two layers) 145
6.55 Difference of liquid volume fraction between domain decomposition and reference solution (two layers) 146
6.56 Comparison of gas enthalpy between domain decomposition and reference solution (two layers) 146
6.57 Difference of gas enthalpy between domain decomposition and reference solution (two layers) 147
6.58 Comparison of liquid enthalpy between domain decomposition and reference solution (two layers) 147
6.59 Difference of liquid enthalpy between domain decomposition and reference solution (two layers) 148
6.60 Comparison of vapor content (in gas) between domain decomposition and reference solution (two layers) 148
6.61 Difference of vapor content (in gas) between domain decomposition and reference solution (two layers) 149

List of Tables

4.1 Error with respect to the reference solution 61
4.2 Comparison of Defect Equation 63
4.3 Comparison of Defect Equation 63
4.4 Performance of Nonlinear Solvers 66

5.1 Error with respect to the reference solution 87
5.2 ∞-norm error 94
5.3 Normalised L^2-norm error 94
5.4 ∞-norm error 101
5.5 Normalised L^2-norm error 104

6.1 Error with respect to the conventional solution 110
6.2 Performance of Newton-finite difference 111

Glossary

C_a	concentration of vapour, 90
C_f	density of fibre, 90
C_g	density of gas, 90
$C_a^*(T)$	saturated water vapour concentration at T, 92
D_M	molecular diffusivity, 90
D_f	diffusion coefficient of water vapour in fibres, 90
H_g	enthalpy of gas, 91
H_l	enthalpy of liquid, 91
K	intrinsical permeability of the fibre, 90
K_{rg}	relative permeability of gas, 90
K_{rl}	relative permeability of liquid, 90
Q_p	evaporation rate, 90
R	gas constant, 93
S_v	specific volume of fabric, 92
T_g	temperature of gas, 91
T_l	temperature of liquid, 91
V_g	velocity of gas, 90
V_l	velocity of liquid, 90
Θ	contact angle, 92
β	average angle of the capillaries in fibres, 93
ϵ	porosity of the fibre, 93
ϵ_f	volume fraction of solid, 90
ϵ_g	volume fraction of gas, 90
ϵ_l	volume fraction of liquid, 90

λ	latent heat of evaporation, 92
μ_g	dynamic viscosity of gas, 90
μ_l	dynamic viscosity of liquid, 90
ρ_l	density of liquid, 90
σ	surface tension, 92
d_c	effective pore radius, 92
$h_{l\leftrightarrow g}$	mass transfer coefficient, 92
k	heat conductivity, 54
k_g	heat conductivity of gas, 91
k_l	heat conductivity of liquid, 91
p_c	capillary pressure, 90
p_g	pressure of gas, 90
r	radial coordinate of a fibre, 90

Chapter 1

Introduction

It is believed that the idea of a domain decomposition related method was first invented by H. Schwarz in 1869 [105]. This method is known as the Schwarz alternating method which was used by Schwarz to obtain a convergence proof of an iterative scheme leading to the solution of an elliptic equation in a flask region [105]. However, the renewed interest in the method began to grow in the 1980s with the increased availability of parallel computers [31] [53] and due to the applications to the numerical analysis of boundary value problems [32] [83]. Since then many different variants of the alternating methods were developed leading to a class of domain decomposition methods [12] [36].

In mathematics, numerical analysis, and numerical partial differential equations, domain decomposition methods are now regular tools for solving partial differential equations by partitioning a computational domain into several subdomains of smaller size. They solve a boundary value problem in the computational domain by splitting it into smaller boundary value problems within those several subdomains and iterating to coordinate the solution between the subdomains [21]. The original computational problem is then turned into a number of computational subproblems of reduced dimensions which are coupled by suitable transmission conditions in an iterative means [94].

While the original Schwarz alternating method does not possess any parallel or distributed properties [12], many of its variants exhibit such properties, making them extremely useful in this modern computing era [60]. On the other hand, it should be stressed that they can be very effective even on sequential computers [134].

Domain decomposition methods have attracted wide research interest particularly with the increasing performance of parallel computing. First developed for elliptical partial differential equations [32] [35], they have been applied as well in parabolic [46] [18] and hyperbolic [128] [33] operators and also in problems involving mixed type of elliptic and hyperbolic operators [107] [50].

Their significances have been experienced in a wide variety of applications. In computational fluid dynamics, they have been implemented in a number of subjects. Heat conduction [27] is a Poisson problem which was very essential in the early development of the methods. It was extended into the advection-diffusion problems, where domain decomposition has been used as a preconditioner [1], applied in blood transport problem [96], and also with finite volume discretisation [127]. Other fields worth mentioning are multiphase flows [124],[122], Stokes problem [88], free surface flows [100], compressible flows [34], and porous media [44] [38]. A wider range of applications cover the problems of elasticity [42], fluid-solid interaction [30] [41], and magnetohydrodynamic [89].

In the multiscale modelling, a homogenisation technique is used to model and separate a heteregenous problem into several different scales [92]. Domain decomposition methods can be applied to decouple the subproblem of each scale [55]. This is also implemented in a problem with rapidly changing properties [120].

According to [107] and [117], the term domain decomposition has been used within the discipline of Partial Differential Equations (PDEs) where it encompasses some different emphases:

1. In parallel computing, it often means the process of distributing data from a computational model among the processors in a distributed memory computer. In this context, domain decomposition refers to the techniques for decomposing a data structure and can be independent of the numerical solution methods. Data decomposition is perhaps a more appropriate term for this process.

2. In continuous levels, domain decomposition deals with the separation of the physical domain into regions that can be modelled with different equations, with interfaces between the domains handled by various conditions (e.g. continuity). In this context, domain decomposition refers to the determination of which PDEs to solve.

3. In discretisation levels, where it is convenient to employ different approximation and solution methods in different regions.

4. In preconditioning methods, domain decomposition refers to the process of subdividing

the solution of a large linear system into smaller problems whose solutions can be used to produce a preconditioner (or solver) for the system of equations that results from discretising the PDEs on the entire domain. In this context, domain decomposition refers only to the solution method for the algebraic system of equations arising from discretisation.

It must be noted that all those aspects can take place in the same time when solving a PDE. An example is a multi-physics and multi-scale process which can be more efficient to solve locally [49]. It is also known that certain physical models or those with complex domains can be solved more conveniently and accurately with certain discretisation. If a uniform and structured mesh can be generated in the local subdomains resulting from the physics or geometry based partitioning, then a fast computational solver can be easily applied to provide a good convergence [94]. In the implementation stage, for a problem of large and complex sysytem, a data decomposition is necessary for parallelisation [67] [72].

Given the above aspects, various interests inspire some important motivations for domain decomposition methods :

- decoupling of highly coupled physics,
- simplification of problems on complicated geometry,
- separation of homogenized region in problems with spatial varying properties,
- ease of parallelisation and good parallel performance,
- superior convergence properties

Domain decomposition algorithms can be divided into two classes according to the partitioning, i.e. those that use overlapping domains and those that use nonoverlapping domains [13]. In overlapping domain decomposition methods, the subdomains overlap by more than the interface, such as the one illustrated in Fig.1.1. In non-overlapping methods, the subdomains intersect only on their interface. An example of nonoverlapping partitioning is given in Fig.1.2

The original Schwarz alternating method [105] is based on an overlapping method, which is also known as the multiplicative Schwarz alternating method [83] or Block Gauss-Seidel iterative method [107].

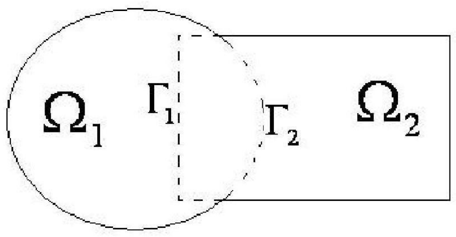

Figure 1.1: Decomposition into two subdomains

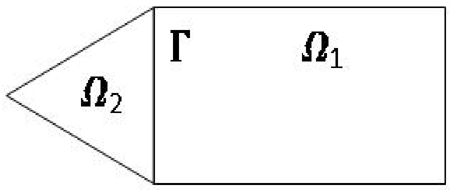

Figure 1.2: Nonoverlapping decomposition into two subdomains

Consider the following linear elliptic Poisson equation in a domain Ω :

$$\begin{aligned} \Delta u &= f \quad \text{in } \Omega \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

where $\partial\Omega$ denotes the boundary of Ω, while f and g are given functions.

The decomposition of the domain Ω into two overlapping subdomains Ω_1 and Ω_2 with the corresponding boundaries $\partial\Omega_1$ and $\partial\Omega_2$ yields the artificial interior boundaries $\Gamma_1 = \partial\Omega_1 \cap \Omega_2$ and $\Gamma_2 = \partial\Omega_2 \cap \Omega_1$. Let $u_1^{(k)}$ and $u_2^{(k)}$ be the solutions in Ω_1 and Ω_2 respectively at the iteration-k, then the algorithm of Schwarz alternating method runs as follows:

```
$k := 0;$ Initial guess $:u^{(0)};$
$u_1^{(0)} := u^{(0)}|_{\Omega_1}; u_2^{(0)} := u^{(0)}|_{\Omega_2};$
do
    Solve $\begin{cases} \Delta u_1^{(k+1)} = f & \text{in } \Omega_1 \\ u_1^{(k+1)} = g & \text{on } \partial\Omega_1 \cap \partial\Omega \\ u_1^{(k+1)} = u_2^{(k)} & \text{on } \Gamma_1 \end{cases}$
                                                                        (1.1)
    Solve $\begin{cases} \Delta u_2^{(k+1)} = f & \text{in } \Omega_2 \\ u_2^{(k+1)} = g & \text{on } \partial\Omega_2 \cap \partial\Omega \\ u_2^{(k+1)} = u_1^{(k+1)} & \text{on } \Gamma_2 \end{cases}$

    $k := k + 1;$
Until converged;
```

For the non-overlapping domain decomposition methods, a unique interface suffices, which is $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$, as can be seen again in Fig.1.2. An example of the methods is Dirichlet-Neumann of which the algorithm is demostrated in the following:

```
$k := 0;$ Initial guess $:\lambda^{(0)};$ input$\theta > 0;$
do
    Solve $\begin{cases} \Delta u_1^{(k+1)} = f & \text{in } \Omega_1 \\ u_1^{(k+1)} = g & \text{on } \partial\Omega_1 \cap \partial\Omega \\ u_1^{(k+1)} = \lambda^{(k)} & \text{on } \Gamma \end{cases}$

    Solve $\begin{cases} \Delta u_2^{(k+1)} = f & \text{in } \Omega_2 \\ u_2^{(k+1)} = g & \text{on } \partial\Omega_2 \cap \partial\Omega \\ \frac{\partial u_2^{(k+1)}}{\partial n} = \frac{\partial u_1^{(k+1)}}{\partial n} & \text{on } \Gamma \end{cases}$        (1.2)

    $\lambda^{(k+1)} := \theta u_2^{(k+1)}|_{\Gamma} + (1-\theta)\lambda^{(k)};$
    $k := k + 1;$
Until converged;
```

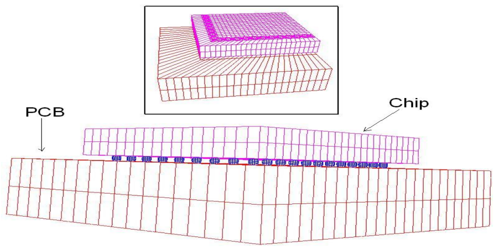

Figure 1.3: Component-based decomposition into a PCB and a chip

1.1 Geometrical and Physical Decomposition

There are certain situations where either partitioning is more beneficial. From the geometrical point of view, the overlapping method may be favoured for the decomposition of the domain in Fig.1.1. An overlapped partitioning into a rectangular and a spherical domain will make the mesh generation easier to handle. It is also more convenient to create a structured mesh for which a fast iterative solver such as the fast Poission solver is very efficient [94]. The same principle also underlies the service of a nonoverlapping method for the problem in Fig.1.2.

For multi-component products in particular, the current state of the electronic packaging industries encourages the employment of nonoverlapping domain decomposition in the computation associated with component gluing [24]. In those industries, many products are assembled from various components, often made by different companies [58]. For example in Fig.1.3, a single chip from one company is bonded to a PCB of another company to build a specific component.

The final product of the assembly manufacturing might lead to irregular shapes with complex geometry [79]. Among all stages in the computational analysis, mesh generation accordingly poses the greatest difficulty. It demands high computing resources and above all, it is manpower intensive which is the most time-consuming [25].

On its own, the individual component can be simpler in domain and may already have an existing mesh, discretisation and solver built by the original manufacturer, or if not, a more convenient grid and numerical technique can be designed [87]. It is therefore of great

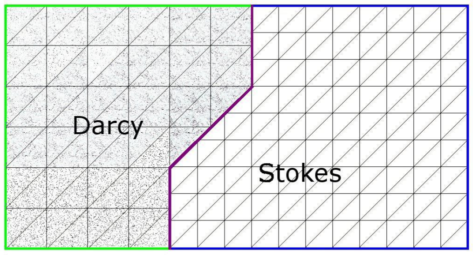

Figure 1.4: Decomposition into Darcy and Stokes region

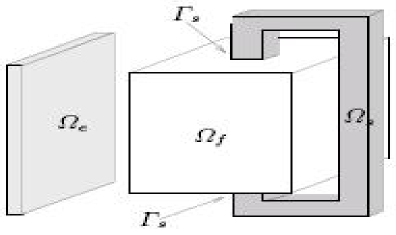

Figure 1.5: Physics and geometrical based decomposition in magnetohydrodynamics

interest to perform the computational analysis of the final product by collaborating the component-based meshing and solution. For this component meshing and gluing technique [25], the nonoverlapping methods appear practically more suitable in the task of coupling the solution in each component which may be obtained by its own existing solvers.

When it comes to physical consideration, nonoverlapped partitioning will be of much help. When solving the compressible Navier-Stokes equations in an exterior domain, it is of interest in the computation to select regions where the viscosity is small and to solve the Euler equations in these regions, since the Euler equations are less costly computationally [47]. Another decomposition approach for the viscid/inviscid interaction is given in [26].

In Stokes-Darcy systems, one can split the domain problem into Stokes equations in the fluid region and Darcy equations for the filtration velocity in the porous medium [44], where such problems appear in several applications, such as well-reservoir coupling in petroleum engineering or the transport of substances accross groundwater.

In the modelling of large scale multiphase flows in permeable media which usually have multi-scale heterogeneities and multi-physics, an efficient parallel simulator which utilizes domain decomposition based upon the physics of the media, is important [124].

1.2 Nonoverlapping Domain Decomposition

Some advantages that certain nonoverlapping domain decomposition methods can contribute are described in the previous section where they can couple problems of different physics, accommodate different numerical treatments, and enable the gluing of simpler component-based geometry for certain complex geometry. This thesis intends to examine a framework of nonoverlapping domain decomposition with such capabilities.

This section briefly examines the traditional nonoverlapping domain decomposition methods for both differential and discrete forms [95] [107] [117].

Consider the following general differential problem :

$$\begin{aligned} Lu &= f \quad \text{in } \ \Omega \\ \gamma u &= c \quad \text{on } \partial\Omega \end{aligned} \tag{1.3}$$

where L is a second order elliptic partial differential operator, γ is a boundary operator, f and c are given functions, and $\partial\Omega$ is the boundary of the domain Ω. Assume that Ω is partitioned into two disjoint subdomains Ω_1 and Ω_2 with an interface Γ as illustrated in Fig.1.2, the splitting version of the problem can be presented as follows. Let u_i be the restriction of u to Ω_i, it follows that

$$\begin{cases} Lu_1 = f & \text{in } \Omega_1 \\ \gamma u_1 = g & \text{on } \ \partial\Omega_1 \cap \partial\Omega \\ Lu_2 = f & \text{in } \Omega_2 \\ \gamma u_2 = g & \text{on } \ \partial\Omega_2 \cap \partial\Omega \end{cases} \tag{1.4}$$

One needs to enforce transmission conditions [95] between u_1 and u_2 across Γ,

$$\begin{cases} \Phi(u_1) = \Phi(u_2) & \text{on } \Gamma \\ \Psi(u_1) = \Psi(u_2) & \text{on } \Gamma \end{cases} \tag{1.5}$$

The matching of the transmission conditions Φ and Ψ can be expressed as certain boundary operator equalities involving the so-called Steklov-Poincare operators [119] and their expressions are determined by the problem. For the case of Poisson problem $\Delta u = f$, the transmission conditions Φ and Ψ represent the continuity of the solution and the normal derivative accross the interface, i.e. $\Phi(u) = u$ and $\Psi(u) = \dfrac{\partial u}{\partial n}$. In other problems such as the Navier Stokes equation, the second transmission condition will need the presence of pressure [95].

It is clear that the nonoverlapping Dirichlet-Neumann scheme, as shown in the formulation (1.2), obeys both transmission conditions where the Dirichlet boundary solves in Ω_1 satisfy the condition Φ and the Neumann boundary solves in Ω_2 satisfy Ψ.

From the linear algebraic point of view, the nonoverlapping decomposition can be realized by first considering the two subdomains as individual problems on their own in the arrangement of the following matrix equation [107] :

$$\begin{bmatrix} A_{\Omega_1} & 0 & A_{1\Gamma} \\ 0 & A_{\Omega_2} & A_{2\Gamma} \\ A_{\Gamma_1} & A_{\Gamma_2} & A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_\Gamma \end{bmatrix} \tag{1.6}$$

where the first, the second and the last equation represent the discretisation in Ω_1, Ω_2 and on the interface Γ respectively.

The above matrix equation can be factorized [107] into

$$\begin{bmatrix} I & 0 & A_{\Omega_1}^{-1} A_{1\Gamma} \\ 0 & I & A_{\Omega_2}^{-1} A_{2\Gamma} \\ 0 & 0 & S^{(1)} + S^{(2)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} A_{\Omega_1}^{-1} & 0 & 0 \\ 0 & A_{\Omega_2}^{-1} & 0 \\ -A_{\Gamma_1} A_{\Omega_1}^{-1} & -A_{\Gamma_2} A_{\Omega_2}^{-1} & I \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_\Gamma \end{bmatrix} \tag{1.7}$$

where $S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma_i} A_{\Omega_i}^{-1} A_{i\Gamma}$. Once the interface solution u_Γ is given, the subdomain solutions u_1 and u_2 can be solved independently using the first and the second equations in (1.7). It can be seen that the original problem (1.6) has been turned into an interface problem, i.e. the third equation in (1.7).

It is known from the spectral properties of elliptic PDEs [107], that the condition number of matrix $S = S^{(1)} + S^{(2)}$ is better than that of matrix A. The nonoverlapping decomposition problem is also known as the Schur problem, after the Schur complement matrix $S^{(i)}$ [54]. In linear algebraic problems, the existing nonoverlapping methods are generally derived by applying a specific preconditioner to the interface equation, hence different preconditioners [15] [68] [107] will lead to different methods.

1.3 Parallel Implementation

While the previous section briefly addresses the specific nonoverlapping method which is examined in this thesis, this section reveals the idea of the method by way of a parallel implementation [56]. The implementation, when run under several processors, might involve the role of a server processor. The description of the idea is given below by showing an implementation of the Dirichlet-Neumann algorithm in a multiprocessor system.

Suppose a domain Ω is partitioned into n nonoverlapped subdomains $\Omega_i, i = 1, ...n$ and each subdomain Ω_i is assigned to one processor-i. In each subdomain the discretisation of the boundary value problem can be done in the conventional way in all interior points and on each point belonging to the real boundary $\partial\Omega_i \cap \partial\Omega$. The only different treatment is for the interface points on the interface boundary Γ_{ij} between subdomain Ω_i and each neighbouring subdomain Ω_j. For this, the discretisation on Γ_{ij} needs external information as the necessary Dirichlet data which must be provided only by the processor-j. In the same way, Ω_j needs Neumann information from the processor-i. Note that due to the possible different meshing and discretisation in each subdomain, the processor-i needs the list and position of each node of both Ω_i and Ω_j at Γ_{ij}, so that the interface boundary condition given by processor-j can be adjusted for the discretisation of Ω_i. The same also applies for Ω_j.

It is now obvious that for each interface Γ_{ij}, there is a sequential procedure :

- processor-j sends the local interface solution value to processor-i as the Dirichlet data,
- after receiving the data sent by processor-j, processor-i solves the corresponding Dirichlet problem
- processor-i sends the local interface normal boundary value processor-j as the Neumann data
- after receiving the data sent by processor-i, processor-j solves the corresponding Neumann problem

In a two subdomain problem, it is not easy to appreciate the parallel property of the Neumann-Dirichlet algorithm. The parallelism of the algorithm can be enjoyed in a multi-subdomain splitting where half of the system solving the corresponding boundary problem can be run in parallel while the other half wait for the interface data. Surely this can only be done at the expense of extra effort in arranging the ordering of the data transfer and subdomain solves. For each subdomain bordering with more than one other subdomain, definitions of the sender of Neumann or Dirichlet data must be carefully set.

One way to exploit the paralellism in a nonoverlapping decomposition is to enforce one of the first transmission condition, either Neumann or Dirichlet, for both subdomains, and update it until the other transmission condition is satisfied.

In the two subdomains problem, the local equations associated with (1.3)- (1.5) are then given by

$$\begin{cases} Lu_i = f & \text{in } \Omega_i \\ \gamma u_i = g & \text{on } \ \partial\Omega_i \cap \partial\Omega \\ \Phi(u_i) = \lambda & \text{on } \Gamma \end{cases}$$

If the above formulation represents the Poisson problem, then Φ can correspond to either Dirichlet or Neumann condition. This automatically satisfies the matching of the transmission condition Φ. The interface residual resulting from the subdomain solution is defined by the other transmission operator Ψ through $D = \Psi(u_1) - \Psi(u_2)$.

An iterative procedure is required to update λ in order to decrease the residual D. Convergence is achieved when $D = 0$, at which point the matching of the second transmission condition Ψ is also satisfied.

Based on the above technique, the procedural steps for each processor in a two subdomain problem is given by :

- Each processor solves the subdomain problem with the same interface condition $\Phi(u) = \lambda$
- processor-1 sends the local $\Psi(u_1)$ to processor-2 at the same time as processor-2 sends $\Psi(u_2)$ to processor-1
- each processor calculates the interface residual $D = \Psi(u_1) - \Psi(u_2)$
- each processor updates λ using the same procedure

Some iterative procedures to obtain $D = 0$ at the interface have been proposed in [101] and [77] where, for the choice of Φ, Roux in [101] uses the Neumann condition whereas Lai in [77] uses the Dirichlet condition.

It is clear that, in this setting, all subdomain solves can be run in parallel. For the multisubdomain splitting, there is an ease of parallelisation in comparison with the Dirichlet-Neumann algorithm, where the ordering of data sending is not necessary. Communication between processors can even be made more handy if the role of a server and clients has been defined in the distributed system. When this is available, each processor only needs to exchange information with the server and therefore does not require the knowledge of the neighbour of its associated subdomain. In addition, this can centralise both the computation of interface residuals and the updating of the interface variable λ in the server, thus giving comfort in embedding an interface solver without any further interaction with other processors. Nevertheless, note that the server still needs the location of all interface nodes from both subdomains.

This scheme seems encouraging for the computation associated with the component meshing and gluing, or other computations which are more convenient to handle by decoupling them into disjoint subproblems. Subdomain computation in each processor can be made more independent where any existing numerical treatment only needs adjustment for the interface boundary condition, in case it is different from the default setting.

1.4 Research Objectives and Proposed Methods

The aim of work in this thesis is to examine the iterative framework of a class of nonoverlapping domain decomposition methods which allow autonomous subdomain computations and coupling flexibilities at the interface. These methods have the combination of capabilities such as coupling problems of different physics, accommodating different numerical treatments, and enabling component meshing and gluing.

A framework of defect correction schemes is proposed where the schemes possess those capabilities and also the advantage of the parallelism mentioned in the previous section, where the subdomain solves become autonomous while a flexible interface solver can also be implemented without having to know the details of subdomains. One of the highlights of the work here is the coupling of subdomain computations in a computational code where the information of computational procedure is limited.

These schemes are used to couple the subdomain solutions in order to cancel the interface residual. Given that they need a solver to update the interface solution, a nonlinear solver is at the centre of the schemes. The research is also concerned with iterative schemes which would be able to cut down the computational costs at the interface solver by avoiding the computation of the Jacobian matrix as well as its inverse. An adaptive-α solver is used because it simply needs a scalar as the approximation of the Jacobian matrix. For the same purpose, a Newton-GMRES and two nonlinear conjugate gradient methods are also employed in the scheme for comparison purposes. Another scheme is also implemented where it uses a Newton method as the nonlinear solver and the Jacobian matrix is computed using the finite difference approximation.

The schemes are applied to three nonlinear problems. The first problem is a heat conduction in a multichip module where there the domain is assembled from many components of different conductivities and physical sizes. The second problem is a 2D single-phase fluid flow with heat transfer. The final problem is a multiphase heat and moisture transfer in a porous textile, where a mixed phase is treated in two layers of fabric with different average permeability. Here, the PHOENICS CFD package is used to solve the system of partial differential equations in each layer where the coupling of the two subproblems is taken care of by the defect correction schemes.

1.5 Thesis Outline

This section outlines the thesis contents, starting from the following chapter. Chapter 2 presents the defect correction technique as the general framework proposed in this thesis. The concept of defect equation is explained before proceeding to the algorithm of some nonlinear solvers. Here the adaptive-α method is proposed and a section of this chapter lists some propositions related to the local convergence of scalar Jacobian approximation in certain nonlinear equations. Some descriptions of the other nonlinear solvers are also given.

Chapter 3 presents the foundation of nonoverlapping domain decomposition methods as preconditioners of a linear system. It demonstrates how the methods such as Dirichlet-Neumann, Neumann-Neumann and the defect correction method are related. The last part of this chapter shows the setting of the defect correction scheme in the iterative framework of mortar element methods.

The first numerical example is given in Chapter 4 where a heat conduction problem in an multichip electronic module occurs. For this nonlinear problem, the Picard linearisation technique is applied to the nonlinear partial differential equation and the preconditioned conjugate gradient method is employed to solve the linearised equation. Results of some numerical experiments are given afterwards, where the comparison of defect equations and the performance of different nonlinear solver are set out.

The description of the second and the third nonlinear case is detailed in Chapter 5, where a 2D single-phase problem and a multiphase porous medium problem in clothing material are discussed. The multiphase problem is presented in a macro scale model. The interphase processes of the transported variables are presented here before some numerical results obtained from the implementation in the PHOENICS code are demonstrated. The computational results of the second problem and the third problem (for one and a two fabric layers case) in this chapter become the reference solution for the comparison in Chapter 6 where the domain decomposition is implemented to couple PHOENICS' subdomain computations. Finally, Chapter 7 gives the conclusion of the research and possible future directions.

Chapter 2

A Defect Correction Scheme

In this chapter, a nonoverlapping domain decomposition scheme which can accommodate the capabilities mentioned in the previous chapter is presented. Just like other nonoverlapping domain decomposition methods [95] [107] [117], the so called defect correction method also reduces the global problem into an interfacial problem defined on the common interface of two non-overlapping neighbouring subdomains.

An early concept of the method is presented in [23] [78] where each subdomain, after the nonoverlapping decomposition of the whole domain, represents an individual component which can be independently created, meshed and solved. It uses the framework of a defect equation [78] in its domain decomposition algorithm.

While many domain decomposition approaches are motivated by the idea that subdomain solutions are treated as a preconditioner [107] to be used in Krylov subspaces [102] such as conjugate gradient methods, the defect correction scheme is not aimed at a preconditioner scheme. However in the next chapter it will also be shown that for linear elliptic problems, this scheme is equivalent to a left preconditioner to the linear algebraic system of the corresponding substructuring problem. Above that, the main aim of the present method concerns the viability of building a framework allowing flexible autonomy of subdomain computations while maintaining the coupling of subdomains. With this in mind, the general idea of the coupling framework can be directly applied to nonlinear problems.

The novel works in this chapter include:

- Proving a local convergence of the α-method as a quasi-Newton solver

- Introducing some general forms of defect equations in order to improve the previously used equation

- Demonstrating the similarity of some forms of defect equations in the Poisson problem

2.1 A Nonoverlapping Scheme

Consider the problem :

$$\begin{aligned} Lu &= f \quad \text{in } \Omega \\ \gamma u &= c \quad \text{on } \partial\Omega \end{aligned} \tag{2.1}$$

where L is a second order elliptic partial differential operator, f and g are given data, $\partial\Omega$ is the boundary of the domain Ω, and γ is a boundary operator (either Dirichlet, Neumann, Robin or mixed condition) imposed on $\partial\Omega$.

As illustrated in the previous chapter, the construction of a nonoverlapping domain decomposition method begins with the division of the domain into several nonoverlapping subdomains before numerical schemes are carried out independently in each subdomain. Such decomposition generates interface lines or areas acting as the boundaries between neighbouring subdomains [107]. An obvious problem arising in the implementation is the need of extra conditions at the interfaces Γ between any 2 adjacent subdomains since these interfaces represent the new boundaries for each subdomain problem. The interface condition of one subdomain problem can be chosen differently from that of its adjoining neighbour, but any condition will function as a coupling of solutions between the two subdomains.

The following describes the nonoverlapping scheme for the splitting version of (2.1) in the defect correction framework. Assume that Ω is partitioned into two nonoverlapping subdomains Ω_1 and Ω_2, and denote the interface by $\Gamma := \overline{\Omega_1} \cap \overline{\Omega_2}$. Now let u_i be the restriction of u to Ω_i, the proposed method constructs a new problem in the following form for $i = 1, 2$:

$$\begin{aligned} Lu_i &= f \quad \text{in } \Omega_i \\ \gamma u_i &= c \quad \text{on } \partial\Omega_i \\ u_i &= \lambda \quad \text{on } \Gamma \end{aligned} \tag{2.2}$$

The additional Dirichlet condition is associated with the interface boundary for each subdomain due to the nonoverlapped partitioning. By choosing the same interface Dirichlet condition, it ensures the continuity of the function along interfaces. Nevertheless this raises a new problem since the accuracy of the solution then depends on the setting of λ. An erroneous λ will lead to inaccurate subdomain solutions. The proposed scheme requires that the value of λ satisfies a constraint condition enforced along the interface [76]. This condition couples the solution of u_1 of Ω_1 and u_2 of Ω_2 and is of the form $D(u_1, u_2) = 0$ where D is called the defect function.

In an iterative sense, the initial guess of λ in general produces subdomain solutions which do not obey the zero defect at the interface. The next iteration then attempts to improve the new value of λ such that it reduces $D(u_1, u_2)$, and the correction is continued until an approximate zero defect is obtained. So the name defect correction follows from this approach.

A reasonable updating of λ is one which is adaptive to the recent value of $D(u_1, u_2)$. Without explicit definition of both the defect D and the updating of λ, the iterative procedure of defect correction method in two subdomains can be expressed as follows

```
$k := 0;$ Initial guess :$\lambda^{(0)}$;
do
    Solve $\begin{cases} Lu_1^{(k+1)} = f & \text{in } \Omega_1 \\ \gamma u_1^{(k+1)} = c & \text{on } \partial\Omega_1 \\ u_1^{(k+1)} = \lambda^{(k)} & \text{on } \Gamma \end{cases}$

    Solve $\begin{cases} Lu_2^{(k+1)} = f & \text{in } \Omega_2 \\ \gamma u_2^{(k+1)} = c & \text{on } \partial\Omega_2 \\ u_2^{(k+1)} = \lambda^{(k)} & \text{on } \Gamma \end{cases}$

    Compute $D(\lambda^{(k)})$;
    Update $\lambda^{(k+1)}$ based on $D(\lambda^{(k)})$;
$k := k + 1;$
Until $D <$ tol;
```

2.2 Defect Equations

This section presents some choices of a defect function for the coupling at the interface. Two transmission conditions have been mentioned in (1.5). Given the nonlinear operator (2.1) and interface boundary value λ, the first transmission condition is automatically satisfied through the continuity of the solution across the interface. By nature the subdomain solutions u_1 and u_2 will be implicitly nonlinear in λ. Thus, any constraint chosen to couple u_1 and u_2 at the interface will generate a nonlinear defect function $D(\lambda)$.

The coupling of subdomain solutions u_1 and u_2 itself should reflect the continuity of the second transmission condition across the interface. Suppose $D_i(\lambda)$ is the component of $D(\lambda)$ contributed by subdomain solution u_i in Ω_i which corresponds to the second transmission operator on the interface Γ. One way to enforce this continuity is by taking $D(\lambda) = D_1(\lambda) - D_2(\lambda)$ and finding the root to the defect equation $D(\lambda) = 0$.

Bearing in mind that one of the aims of the proposed framework is to enable autonomous subdomain computations, hence also for the computation of D_i, it is necessary that the unit and the physical interpretation of D_1 and D_2 must be equivalent to each other.

In the remainder of this thesis, the problem is presented in a finite dimensional framework. Here, $\lambda \in \Re^N$ and $\mathbf{D} : \Re^N \to \Re^N$ represent the finite dimensional approximation of the interface variable and the defect function respectively, whereas $u_h = [u_{1,h}\, , \, u_{2,h}]$ is the finite dimensional approximation of the problem.

Some alternatives of defect equations are demonstrated in the following:

- Pointwise finite difference approximations of the normal derivative $\dfrac{\partial u}{\partial n}$

$$D(\lambda) = \partial^+_{n,j}(u_{p,h}(\lambda)|_\Gamma) - \partial^-_{n,j}(u_{q,h}(\lambda)|_\Gamma) = 0 \tag{2.3}$$

  where $\partial^+_{n,j}$ and $\partial^-_{n,j}$ are the $j-$th order forward and backward difference approximation of normal derivative. For two subdomains sharing an interface line, then p is the index of the subdomain located normal to the interface line, whereas q is the index of the other subdomain. With this convention, the definition of D_1 and D_2 (either forward or backward difference) for two coupled subdomain solutions u_1 and u_2 will depend on the position of each subdomain with respect to the interface line. The discrete operator $\partial^+_{n,j}$ and $\partial^-_{n,j}$ can be, for example, of first order ($j = 1$) or second order ($j = 2$) finite difference. This defect function is suggested in [78] and has been used in [25], [6], and [57] to solve some heat conduction problems. It becomes a representation of point-wise continuity of the normal derivatives along the interface [78].

- Integration of the transmission operator Ψ

$$D(\lambda) = \int_{\Gamma_k} \Psi(u_{1,h}(\lambda)) - \Psi(u_{2,h}(\lambda)) dS = 0 \tag{2.4}$$

for each segment $\Gamma_k \in \Gamma$ at the interface, where $\Gamma = \bigcup_k \Gamma_k$ is the union of segments of interface. The transmission operator Ψ has been introduced in (1.5) where it represents the normal derivative operator for the case of Poisson problem. The contribution D_i from computation in Ω_i is given by $D_i(\lambda) = \int_{\Gamma_k} \Psi(u_{i,h}(\lambda))dS$. This improves the idea of a defect equation suggested in [78], i.e. line integrals of the point-wise continuity of the normal derivatives along the interface.

- Discretisation of the residual of governing equation at the interface

$$D(\lambda) = f_h(u_h(\lambda)) - L_h u_h(\lambda) = 0 \tag{2.5}$$

  A numerical approximation of the defect equation can be obtained by applying a discretisation method at the interface between two subdomains, which may involve interface grid points and their immediate neighbours. The definition of $D_1(\lambda)$ and $D_2(\lambda)$ can then be adjusted accordingly. This may be of interest if the interface problem is handled separately (e.g. in a server processor) from subdomain computations (in many client processors), where a server can employ its own discretisation and single-handedly define the contribution of defect D_i from client-i according to the discrete form. This defect equation can be formed without any detail of the transmission operator Ψ. However, it is more suitably applied when the mathematical models are uniform across the interface, as will be shown in the next section. When properties of governing equation vary across the interface, further treatment such as property averaging is needed to define this defect equation.

If (2.1) is a Poisson problem, the transmission condition is commonly known as the continuity of normal derivatives accross the interface. In the next section, it will be shown that for this type of problem, (2.4) can reduce to (2.5) using an equivalent discretisation scheme.

Notice that all alternatives above can be represented by a more general discrete operator:

$$D(\lambda) = X_h(u_{1,h}(\lambda)|) - X_h(u_{2,h}(\lambda)|) = 0 \tag{2.6}$$

where X_h represent any kind of discrete equation which is possibly defined at the interface in any type of discretisation scheme.

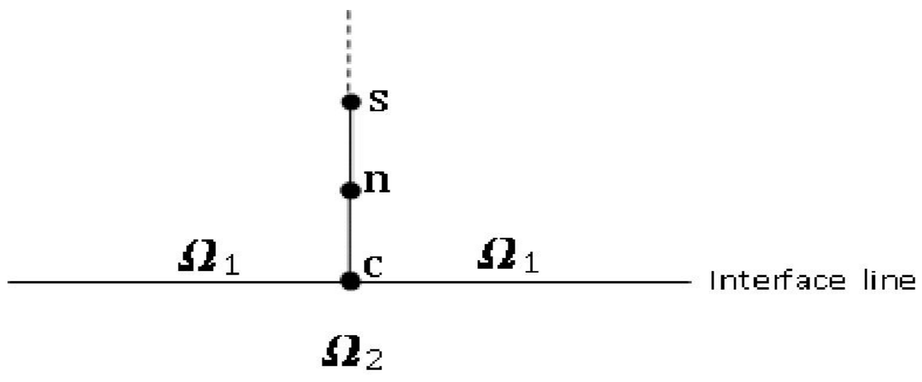

Figure 2.1: Vertical line perpendicular to interface line

2.3 A Normal Derivative Approximation For Defect Equations

Consider a Poisson problem in a 2-dimensional domain Ω :

$$\begin{aligned} \Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{2.7}$$

where $f \in L^2(\Omega)$ is a given function and $\partial\Omega$ is the boundary of Ω. In a nonoverlapping decomposition $\Omega = \Omega_1 \cup \Omega_2$ with the interface $\Gamma := \partial\Omega_1 \cap \partial\Omega_2$, the common transmission conditions accross Γ are the continuity of the subdomain solutions and the continuity of the normal derivatives. In the defect correction scheme, a defect equation associated with the latter transmission condition should be derived to satisfy this continuity, therefore it can be written in continuous fashion : $D = \frac{\partial u_1}{\partial n} - \frac{\partial u_2}{\partial n}$ or $D = \frac{\partial u_1}{\partial n_1} + \frac{\partial u_2}{\partial n_2}$ if the n_i is the outward normal to Ω_i and the convention $n_1 = n$ is taken.

Apparently, the approximation of normal derivative is extremely significant for the accuracy. On the orthogonal grid like that in Fig.2.1 one might be tempted to approximate the normal derivative at the interface point Γ_c using the nodal value along the vertical line perpendicular to the interface, such as with the forward difference

$$\frac{\partial u_1}{\partial n_1}|_{\Gamma_c} \approx \frac{u_n - u_c}{\Delta y}$$

which is exactly the operator $\partial^+_{n,1}$ in (2.3). However this may lead to significant errors. It is second order accurate only in a one-dimensional problem with zero sources. In two-dimensions or if a large source exists, the finite difference approximation might cause an

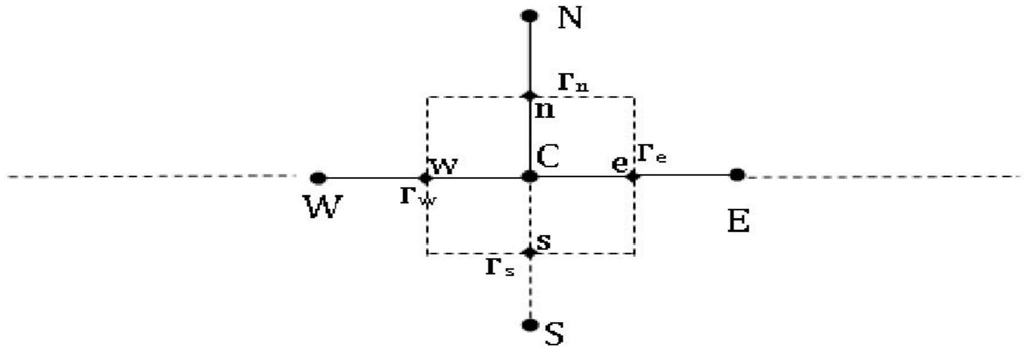

Figure 2.2: Discretisation in orthogonal grid

inconsistency of discretisation with the governing equation [59]. In finite element discretisation, this is very obvious and straightforward. It can also be illustrated in a simple finite volume representation. Now consider the 2D rectangular volume V surrounded by the dashed line $\overline{enws}$ in Fig.2.2. The real nodal points are E, N, W, S while e,n,w,and s are imaginary points to represent the volume.

The finite volume integration in the volume V gives

$$\int_V \Delta u \ dV = \int_{\Gamma_e} \frac{\partial u}{\partial n} dS + \int_{\Gamma_w} \frac{\partial u}{\partial n} dS + \int_{\Gamma_n} \frac{\partial u}{\partial n} dS + \int_{\Gamma_s} \frac{\partial u}{\partial n} dS = \int_V f \ dV$$

Let $\overline{CN} = \overline{CS} = \Delta y$ and $\overline{CW} = \overline{CE} = \Delta x$. Using the central difference for the normal derivative and averaging over each corresponding face, it can be obtained that

$$\frac{u_E - u_C}{\Delta x} \Delta y + \frac{u_W - u_C}{\Delta x} \Delta y + \frac{u_N - u_C}{\Delta y} \Delta x + \frac{u_S - u_C}{\Delta y} \Delta x = f \Delta x \Delta y$$

Now the domain is separated with the interface line cutting through the nodal point E, C, W in Fig.2.3, with $\Gamma_e = \Gamma_{e1} \cup \Gamma_{e2}$ and $\Gamma_w = \Gamma_{w1} \cup \Gamma_{w2}$.

Let u_1 and u_2 be the subdomain solution in Ω_1 and Ω_2. When the normal derivatives of u_1 and u_2 at the interface point C are approximated with the backward $\partial_{n,1}^-$ or forward difference $\partial_{n,1}^+$ using the values at nodal points S, N along the vertical line, the defect is given by

$$D = \frac{\partial u_1}{\partial n}|_{\Gamma_c} - \frac{\partial u_2}{\partial n}|_{\Gamma_c} = \frac{u_N - u_C}{\Delta y} - \frac{u_C - u_S}{\Delta y} = \frac{u_N - 2u_C + u_S}{\Delta y}$$

The zero defect will obey a second order approximation of $\Delta u = 0$ in one-dimensional problem. However, this will not be consistent with the problem (2.7) since it is a two dimensional problem and furthermore it has a source f at the right hand side.

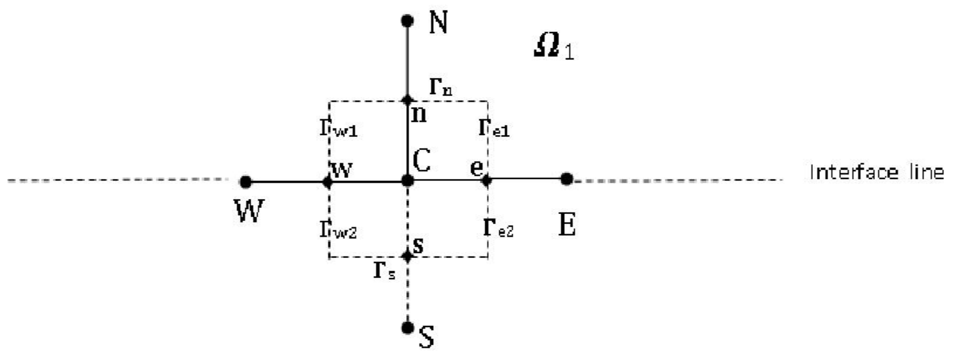

Figure 2.3: Subdivision in orthogonal grid

A better approximation is given in the following. Applying a finite volume integration in the control volume $V_1 = \overline{nwCen}$ in subdomain Ω_1 yields

$$\int_{V_1} \Delta u_1 \; dV = \int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n} dS + \int_{\Gamma_n} \frac{\partial u_1}{\partial n} dS + \int_{\overline{wCe}} \frac{\partial u_1}{\partial n_1} dS = \int_{V_1} f \; dV$$

It follows that

$$\int_{\overline{wCe}} \frac{\partial u_1}{\partial n_1} dS = \int_{V_1} f \; dV - \int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS - \int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n_1} dS - \int_{\Gamma_n} \frac{\partial u_1}{\partial n_1} dS$$

Similarly from the control volume $V_2 = \overline{swCes}$ in subdomain Ω_2,

$$\int_{V_2} \Delta u_2 \; dV = \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n_2} dS + \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS + \int_{\Gamma_s} \frac{\partial u_2}{\partial n_2} dS + \int_{\overline{wCe}} \frac{\partial u_2}{\partial n_2} dS = \int_{V_2} f \; dV$$

hence

$$\int_{\overline{wCe}} \frac{\partial u_2}{\partial n_2} dS = \int_{V_2} f \; dV - \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n} dS - \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS - \int_{\Gamma_s} \frac{\partial u_2}{\partial n_2} dS$$

The outward normal vector n_1 and n_2 at $\overline{wCe}$ have opposite direction, $n_2 = -n_1$. Suppose the normal direction is $n = n_1$, the defect value which is difference of normal derivatives at $\overline{wCe}$ in the n direction can be expressed by the functional

$$D = \int_{\overline{wCe}} \frac{\partial u_1}{\partial n} - \frac{\partial u_2}{\partial n} dS = \int_{\overline{wCe}} \frac{\partial u_1}{\partial n_1} dS + \int_{\overline{wCe}} \frac{\partial u_2}{\partial n_2} dS$$

This defect has the value of

$$\int_{V_1} f \; dV - \int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS - \int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n_1} dS - \int_{\Gamma_n} \frac{\partial u_1}{\partial n_1} dS + \int_{V_2} f \; dV - \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n_2} dS - \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS - \int_{\Gamma_s} \frac{\partial u_2}{\partial n_2} dS$$

They can be rearranged in

$$(\int_{V_1} f\ dV + \int_{V_2} f\ dV) - (\int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n_2} dS) - (\int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS) - \int_{\Gamma_n} \frac{\partial u_1}{\partial n_1} dS - \int_{\Gamma_s} \frac{\partial u_2}{\partial n_2} dS$$

The terms $\int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS$ and $\int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n_2} dS$ must be better approximated using the nodal values at interface points W,C,E than those at the vertical points S and N. Bearing this mind, the approximation of normal derivative by using only the vertical nodal values can then present huge inaccuracy. In fact, in view of the total control volume $V = V_1 \cup V_2$, it is wished that

$$\int_{\Gamma_{w1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{w2}} \frac{\partial u_2}{\partial n_2} dS \approx \int_{\Gamma_w} \frac{\partial u}{\partial n} dS$$

and

$$\int_{\Gamma_{e1}} \frac{\partial u_1}{\partial n_1} dS + \int_{\Gamma_{e2}} \frac{\partial u_2}{\partial n_2} dS \approx \int_{\Gamma_e} \frac{\partial u}{\partial n} dS$$

With these approximations and those of the normal derivatives at Γ_n and Γ_s with central difference, the defect function is roughly

$$D \approx \int_{\overline{wCe}} \frac{\partial u_1}{\partial n} - \frac{\partial u_2}{\partial n} dS = f\Delta x \Delta y - \frac{u_E - u_C}{\Delta x}\Delta y - \frac{u_W - u_C}{\Delta x}\Delta y - \frac{u_N - u_C}{\Delta y}\Delta x - \frac{u_S - u_C}{\Delta y}\Delta x$$

which is the residual of the finite volume discretisation in the cell around the interface point C. Given that there can be a great deal of normal derivative approximation apart from above, it will be sensible if the defect equation is defined as the residual of discretised governing equation at the interface. For the Poisson problem (2.7), this residual is equivalent to a functional of the difference of normal derivatives, which implies the agreement between (2.5) and (2.4).

2.4 Nonlinear Solvers

This section concerns the updating of λ in the attempt to correct the defect function which couples subdomain solutions. Following the form of defect equation in the section 2.2, the mathematical presentation throughout this section is also given in discrete spaces. The defect problem can now be described as the following finite dimensional nonlinear problem: *Let $\lambda \in \Re^s$ be the interface boundary value, find the roots of the nonlinear defect equation* [77]

$$\mathbf{D}(\lambda) = 0, \quad \lambda = (\lambda_1, \ldots, \lambda_s)^T \tag{2.8}$$

subject to (2.2), where $\mathbf{D} : \Re^s \rightarrow \Re^s$ *is the interface problem and its Jacobian matrix* $J(\lambda)$ *may be sparse and diagonal dominant.*

The most well-known method for solving this nonlinear equation is the Newton method [66] which leads to the update:

$$\lambda^{(n+1)} = \lambda^{(n)} - J(\lambda^{(n)})^{-1}\ D(\lambda^{(n)}).$$

where $J = \dfrac{\partial(D_1 ... D_s)}{\partial(\lambda_1 ... \lambda_s)}$ is the Jacobian matrix of the function D.

The method has an excellent local convergence properties [65]. In some cases, it is possible to know the sparsity pattern of the Jacobian matrix. However, the analytical form of the Jacobian matrix in general cannot be calculated. Furthermore, finding a finite difference approximation to the Jacobian matrix is not always encouraging, because the computational work is expensive. An evaluation of D will need a solve of (2.1) in all subdomains. Therefore the interest is in finding nonlinear methods which can circumvent the computation of the Jacobian matrix and its inverse. Three solvers from the class of quasi-Newton and nonlinear conjugate gradient are proposed in this chapter.

2.4.1 Adaptive α method

The first solver, which belongs to the family of quasi-Newton schemes [65], only uses a scalar value [75] as the aproximation of the Jacobian matrix and therefore leads to the update

$$\lambda^{(n+1)} = \lambda^{(n)} - \alpha_n^{-1}\ D(\lambda^{(n)}) \tag{2.9}$$

where α_n^{-1} is an adaptive rate approximating $J(\lambda^{(n)})^{-1}$.

In this work, the following update for the value of α_n [75] is used:

$$\alpha_{n+1} = \alpha_n \frac{\|D(\lambda^{(n+1)}) - D(\lambda^{(n)})\|}{\|D(\lambda^{(n)})\|} \tag{2.10}$$

The adaptive-α method, as a nonlinear solver, has been introduced in [77] and used in [25], [24] for some domain decomposition applications, yet without formal convergence analysis. In [75], Lai used this method to couple two subdomain computations of an advection diffusion process in 1D. The analysis is based on the concept of the shooting method where the convergence is shown specifically just for the two subdomain case, hence not as a nonlinear solver. In the following, a local convergence of the method is demonstrated as a quasi-Newton method for some systems of nonlinear equations.

For a system of nonlinear equations with a diagonal dominant Jacobian matrix with positive diagonal entries, a scalar can be used as a Jacobian approximation which leads a sufficiently good initial condition into convergence. Some propositions which support that will be shown below. They are built from the standard assumptions and a theorem in [65]. The remainder of this section gives the foundation of the local convergence of this α method in solving the system of nonlinear equations :

$$F(x) = 0 \tag{2.11}$$

where $F : R^N \rightarrow R^N$ and $x^* \in R^N$ is a solution to the equations. The ith component of F is denoted by f_i. If the components of F are differentiable at $x \in R^N$, the Jacobian matrix $F'(x)$ is defined by

$$F'(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x).$$

The fundamental theorem of calculus can be expressed as follows

Lemma 1 *Let F be differentiable in an open set $\Omega \subset R^N$ and let $x^* \in \Omega$. Then for all $x \in \Omega$ sufficiently near x^*,*

$$F(x) - F(x^*) = \int_0^1 F^{'}(x^* + t(x - x^*))(x - x^*)dt \tag{2.12}$$

In this section, $\|.\|$ will denote a norm on R^N as well as the induced matrix norm.

Definition 1 *Let $\|.\|$ be a norm on R^N. The induced matrix norm of an $N \times N$ matrix A is defined by*

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

If $\|.\|_p$ is the l^p norm, the norm of a vector x is defined by

$$\|x\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p}$$

There are some assumptions made on F. One of them is the smoothness assumption on $F'(x)$ in order to estimate the error approximation. For this, the notion of Lipschitz continuity is needed.

Definition 2 *Let $\Omega \subset R^N$ and let $G : \Omega \to R^M$. G is Lipschitz continuous on Ω with Lipschitz constant γ if*

$$\|G(x) - G(y)\| \leq \gamma \|x - y\|$$

for all $x, y \in \Omega$

Standard assumptions:

1. Equation (2.11) has a solution x^*
2. $F' : \Omega \to R^{N \times N}$ is Lipschitz continuous with Lipschitz constant γ
3. $F'(x^*)$ is nonsingular

Iterative methods can be classified by their rates of convergence. One of these is the linear convergence.

Definition 3 *Let $x_n \subset R^N$ and $x^* \in R^N$. Then $x_n \to x^*$ q-linearly with q-factor $\sigma \in (0, 1)$ if $\|x_{n+1} - x^*\| \leq \sigma \|x_n - x^*\|$*

A theorem in [65] is used to obtain a local linear convergence of the α method. In what follows, $\mathcal{B}(r)$ denotes the ball of radius r about the solution x^*, and if $x_n \in R^N$ then $e_n = x_n - x^*$ denotes the error.

Theorem 1 *Let the standard assumptions hold. Then there are $K_B > 0, \delta > 0$, and $\delta_1 > 0$ such that if $x_0 \in \mathcal{B}(\delta)$ and the matrix-valued function $B(x)$ satisfies*

$$\left\| I - B(x)F'(x^*) \right\| = \rho(x) \leq \delta_1 \tag{2.13}$$

for all $x \in \mathcal{B}(\delta)$, then the iteration

$$x_{n+1} = x_n - B(x_n)F(x_n) \tag{2.14}$$

converges q-linearly to x^* with

$$\|e_{n+1}\| \leq K_B \left(\rho(x_n) + \|e_n\|\right) \|e_n\|$$

Proof This proof is rewritten from [65] with slight modification, because it will be used in subsequent analyses. First, by (2.13) we have

$$\begin{aligned} \|B(x)\| = & \left\|B(x)F'(x^*)F'(x^*)^{-1}\right\| \leq \left\|B(x)F'(x^*)\right\| \left\|F'(x^*)^{-1}\right\| \\ \leq & \ M_B = (1 + \delta_1) \left\|F'(x^*)^{-1}\right\| \end{aligned} \tag{2.15}$$

Using (2.12), the error of the iteration (2.14) can be written in the following equation:

$$\begin{aligned} e_{n+1} &= e_n - B(x_n)F(x_n) = \int_0^1 (I - B(x_n)F'(x^* + te_n))e_n dt \\ &= (I - B(x_n)F'(x^*))e_n + B(x_n)\int_0^1 (F'(x^*) - F'(x^* + te_n))e_n dt \end{aligned}$$

From the above equation and (2.15), it follows that

$$\|e_{n+1}\| \leq \rho(x_n)\|e_n\| + M_B\gamma\|e_n\|^2/2 \tag{2.16}$$

A local linear convergence is obtained in the neighbourhood $\mathcal{B}(\delta)$ with

$$\delta_1 < 1$$

and

$$\delta < \frac{2(1-\delta_1)}{\gamma M_B} \tag{2.17}$$

since

$$\rho(x_n) + M_B\gamma\delta/2 < \delta_1 + M_B\gamma\delta/2 < 1.$$

Continuing (2.16),

$$\begin{aligned} \|e_{n+1}\| &\leq \rho(x_n)\|e_n\| + M_B\gamma\|e_n\|^2/2 \\ &\leq \rho(x_n)\|e_n\| + M_B\gamma\|e_n\|^2/2 + M_B\gamma\rho(x_n)\|e_n\|/2 + \|e_n\|^2 \\ &\leq K_B(\rho(x_n) + \|e_n\|)\|e_n\| \end{aligned}$$

The proof completes with $K_B = 1 + M_B\gamma/2$. ▮

The closer the Jacobian approximation $B(x)$ is to $F'(x^*)$, the better is the local convergence of the quasi-Newton method. However, the rate of convergence indicates that $\|I - B(x)F'(x^*)\| \geq 1$ will not satisfy the sufficiency of the convergence. The notion of approximate inverse becomes relevant here.

Definition 4 *Let A and B be $N \times N$ matrices. Then B is an approximate inverse of A if $\|I - BA\| < 1$*

Corollary 1 *Let the assumptions of Theorem-1 hold. If $B(x_n)$ is an approximate inverse of $F'(x^*)$ for each n, then the conclusions of Theorem-1 also hold.*

Proof The proof is immediate. Since $B(x_n)$ is an approximate inverse for each n,

$$\delta_1 = \max_n \left\| I - B(x_n)F^{'}(x^*) \right\| < 1$$

Since $1 - \delta_1 > 0$, it follows from (2.17) that $\delta > 0$ can be preserved. ▮

There is a situation where the diagonal elements represent a good approximation of the Jacobian matrix. This happens in a strictly diagonal dominant $F^{'}(x^*)$.

Corollary 2 *Let the standard assumptions hold and let $F^{'}(x^*)$ be strictly diagonal dominant. Then there are $K > 0$ and $\delta > 0$ such that if $x_0 \in \mathcal{B}(\delta)$ and D is the diagonal of $F^{'}(x^*)$, then the iteration*

$$x_{n+1} = x_n - D^{-1}F(x_n)$$

converges q-linearly to x^* and

$$\|e_{n+1}\|_\infty \leq K \|e_n\|_\infty$$

Proof Let δ be small enough that the conclusions of Theorem-1 hold. Let $F^{'}(x^*) = D + E$, where $D = (d_{ii})$ is the diagonal matrix and $E = (e_{ij})$ is the matrix with off-diagonal elements of $F^{'}(x^*)$. It follows that

$$\left\| I - D^{-1}F^{'}(x^*) \right\| = \left\| I - D^{-1}(D + E) \right\| = \left\| D^{-1}E \right\|$$

In the ∞-norm, the property of strictly diagonal dominance gives :

$$\left\| D^{-1}E \right\|_\infty = \max_i \frac{\sum_{j=1, j\neq i}^{n} |e_{ij}|}{d_{ii}} < 1$$

So D is an approximate inverse of $F^{'}(x^*)$. The local convergence, K and δ then follow from Corollary-1 in ∞-norm. ▮

The use of the α method can be viewed as the approximation of the Jacobian by a diagonal matrix with a uniform value. Definitely, this approximation will not be as good as the diagonal of the real Jacobian matrix. Nevertheless, for a more strictly diagonal dominant Jacobian, a uniform diagonal matrix can still obtain a local convergence.

Theorem 2 *Let the standard assumptions hold and let $F^{'}(x^*) = (f_{ij})$ be strictly diagonal dominant with positive diagonal entries. Then there are $\delta > 0, \delta_2 > 0$ and $\alpha > 0$ such that if $x_0 \in \mathcal{B}(\delta)$ and $\max_i \frac{\sum_{j=1,j\neq i}^{n} |f_{ij}|}{f_{ii}} = \rho < \delta_2$, then the iteration*

$$x_{n+1} = x_n - \alpha^{-1}F(x_n)$$

converges q-linearly to x^* in the ∞-norm

Proof Let $\Delta = I - \alpha^{-1}F^{'}(x^*)$ and $F^{'}(x^*) = D + E$ where D and E are the diagonal and the off-diagonal of $F^{'}(x^*)$. Then

$$\begin{aligned} \|\Delta\| = &\ \left\|I - \alpha^{-1}F^{'}(x^*)\right\| \leq \|I - \alpha^{-1}D\| + \|\alpha^{-1}D\| \, \|D^{-1}E\| \\ &\leq \max_i(|1 - f_{ii}|/\alpha) + \frac{\max_i(|f_{ii}|)}{\alpha} \left\|D^{-1}E\right\| \end{aligned}$$

Let $m_1 = \min_i(|f_{ii}|)$ and $m_2 = \max_i(|f_{ii}|)$ and let $\rho = \|D^{-1}E\|$. The first term in the right hand side of the above inequality is subject to the position of α with respect to m_1 and m_2. There are two possible cases :

- If $\alpha \in [\frac{m_1 + m_2}{2}, \infty)$, the norm of Δ can be expressed as

$$\begin{aligned} \|\Delta\| \leq &\ \frac{1}{\alpha}(\alpha - m_1) + \frac{\rho m_2}{\alpha} \\ \leq &\ \frac{1}{\alpha}(\alpha - m_1 + \rho m_2) = 1 - \frac{m_1 - \rho m_2}{\alpha} \end{aligned} \tag{2.18}$$

  If $\rho < m_1/m_2$, then $\|\Delta\| < 1$ will be obtained in this range of α.

- If $\alpha \in (0, \frac{m_1 + m_2}{2}]$, the inequality can be written in

$$\begin{aligned} \|\Delta\| \leq &\ \frac{1}{\alpha}(m_2 - \alpha) + \frac{\rho m_2}{\alpha} \\ \leq &\ \frac{1}{\alpha}(m_2 - \alpha + \rho m_2) = \frac{(1+\rho)m_2}{\alpha} - 1 \end{aligned} \tag{2.19}$$

  It is easy to verify that $\rho < m_1/m_2$ must be preserved in the inequality. Furthermore, given $\rho \in [0, m_1/m_2)$, setting $(\rho m_2 + m_2)/2 < \alpha \leq (m_1 + m_2)/2$ satisfies $\|\Delta\| < 1$.

Combining both, it is contained that $\alpha > (m_2 + \rho m_2)/2$, with $\rho = \|D^{-1}E\| < m_1/m_2$ is the sufficient condition for $\left\|I - \alpha^{-1}F^{'}(x^*)\right\| < 1$ under the corresponding assumptions. In

∞-norm, there holds

$$\left\|D^{-1}E\right\|_{\infty} = \max_i \frac{\sum\limits_{j=1, j\neq i}^{n} |f_{ij}|}{f_{ii}} < m_1/m_2.$$

This completes the proof and the local linear convergence factor and δ follow from Corollary-1 in the ∞-norm. ∎

Although there is no upper bound for α in the above Theorem, a larger α will lead to a bigger $\|\Delta\|$ and $x_{n+1} = x_n + \alpha^{-1}F(x_n)$ shows that the iteration will be very slow. It is clear from (2.18) and (2.19) that $\alpha = (m_1 + m_2)/2$ gives the smallest $\|\Delta\|$. On the other hand it is not an easy task to estimate the minimum and the maximum diagonal element of $F^{'}(x^*)$. The following lemma is the combination of two lemmas by Dennis and Schnabel [29] which will be of very much help later.

Lemma 2 *Let $F : \Re^n \to \Re^m$ be continuously differentiable in the open convex set $D \subset \Re^n$,, let $x \in D$, and let $F^{'}$ be Lipschitz continuous in the neighborhood D with the constant γ using a vector norm and the induced matrix operator norm, and assume that $F^{'}(x)^{-1}$ exists. Then there exist $\epsilon > 0$, and $0 < \mu < \beta$, such that*

$$\mu \|v - u\| \leq \|F(v) - F(u)\| \leq \beta \|v - u\|,$$

for all $v, u \in D$ for which $\max(\|v - x\|, \|u - x\|) \leq \epsilon$

Proof With the help of the equation of fundamental calculus (2.12), for u sufficiently close to v,

$$F(v) - F(u) = \int_0^1 F^{'}(u + t(v-u))(v-u)dt$$

Therefore the following can be expressed :

$$\begin{aligned} F(v) - F(u) - F^{'}(x)(v-u) = \; & \int_0^1 F^{'}(u^* + t(v-u))(v-u)dt - \int_0^1 F^{'}(u^* + t(x-u))(v-u)dt \\ & + \int_0^1 F^{'}(u^* + t(x-u))(v-u)dt - \int_0^1 F^{'}(x)(v-u)dt \end{aligned}$$

An inequality in norm can be presented in

$$\begin{aligned}
\left\|F(v)-F(u)-F^{'}(x)(v-u)\right\| \leq\ & \int_0^1 \left\|F^{'}(u+t(v-u))-F^{'}(u+t(x-u))\right\| \|v-u\|\,dt \\
& +\int_0^1 \left\|F^{'}(u+t(x-u))-F^{'}(x)\right\| \|v-u\|\,dt \\
\leq\ & \int_0^1 \gamma \|t((v-u)-(x-u))\| \|v-u\|\,dt \\
& +\int_0^1 \gamma \|u+t(x-u)-x\| \|v-u\|\,dt \\
\leq\ & \gamma \int_0^1 \|t(v-x)\| \|v-u\|\,dt+\gamma \int_0^1 \|(1-t)(u-x)\| \|v-u\|\,dt \\
\leq\ & \gamma \|v-x\| \|v-u\| \int_0^1 tdt+\gamma \|u-x\| \|v-u\| \int_0^1 (1-t)dt \\
\leq\ & \gamma \frac{\|v-x\|+\|u-x\|}{2} \|v-u\|
\end{aligned}$$

Using both the triangle inequality [73] and the above inequality,

$$\begin{aligned}
\|F(v)-F(u)\| &\leq \left\|F^{'}(x)(v-u)\right\|+\left\|F(v)-F(u)-F^{'}(x)(v-u)\right\| \\
&\leq \left[\left\|F^{'}(x)\right\|+\frac{\gamma}{2}(\|v-x\|+\|u-x\|)\right] \|v-u\| \\
&\leq \left[\left\|F^{'}(x)\right\|+\gamma\epsilon\right] \|v-u\|
\end{aligned}$$

which proves the upper bound with $\beta=\left\|F^{'}(x)\right\|+\frac{\gamma}{\epsilon}$.
Similarly,

$$\begin{aligned}
\|F(v)-F(u)\| &\geq \left\|F^{'}(x)(v-u)\right\|-\left\|F(v)-F(u)-F^{'}(x)(v-u)\right\| \\
&\geq \left(\frac{1}{\|F^{'}(x)^{-1}\|}-\frac{\gamma}{2}(\|v-x\|+\|u-x\|)\right) \|v-u\| \\
&\geq \left(\frac{1}{\|F^{'}(x)^{-1}\|}-\gamma\epsilon\right) \|v-u\|
\end{aligned}$$

Thus if $\epsilon < 1/(\left\|F^{'}(x)^{-1}\right\|\gamma)$, the lower bound holds with

$$\mu=\left(\frac{1}{\|F^{'}(x)^{-1}\|}\right)-\gamma\epsilon > 0.$$

∎

Theorem 3 *Let the standard assumptions hold, let $F^{'}(x^{*}) = (f_{ij})$ be strictly diagonal dominant with positive diagonal entries, and let $\|.\|$ be the ∞-norm. Then there are $\delta > 0$, $\delta_2 > 0$, and $\delta_3 > 0$ such that if $\max\limits_{i} \frac{\sum\limits_{j=1, j\neq i}^{n} |f_{ij}|}{f_{ii}} = \rho < \delta_2$, $\|F'(x^*)^{-1}\| < \delta_3$, and $x_0, x_{-1} \in \mathcal{B}(\delta)$ with $x_0 \neq x_{-1}$, then the iteration*

$$x_{n+1} = x_n - \alpha_n^{-1} F(x_n)$$

with

$$\alpha_n = \alpha_{n-1} \frac{\|F(x_n) - F(x_{n-1})\|}{\|F(x_{n-1})\|} \tag{2.20}$$

converges q-linearly to x^* in ∞-norm.

Proof This theorem could be regarded as a corollary to Theorem-2 and Lemma-2. From

$$x_n - x_{n-1} = \alpha_{n-1}^{-1} F(x_{n-1})$$

it follows that

$$\|x_n - x_{n-1}\| = \alpha_{n-1}^{-1} \|F(x_{n-1})\|,$$

Therefore

$$\alpha_n = \alpha_{n-1} \frac{\|F(x_n) - F(x_{n-1})\|}{\|F(x_{n-1})\|} = \frac{\|F(x_n) - F(x_{n-1})\|}{\|x_n - x_{n-1}\|}.$$

Let δ be small enough that $\mathcal{B}(\delta) \subset D$ where D is the open convex set in Lemma-1. Assume that $x_{n-1}, x_n \in \mathcal{B}(\delta)$ and $x_{n-1} \neq x_n$. The lower bound of Lemma-2 then gives

$$\alpha_n \geq \mu \geq \frac{1}{\|F(x^*)^{-1}\|} - \gamma\delta$$

Now choose $\alpha \in \Re$ such that $\alpha > \frac{1+\rho}{2} m_2$. If $\|F'(x^*)^{-1}\| < 1/\alpha$, then $\theta = \frac{1}{\gamma}\left(\frac{1}{\|F(x^*)^{-1}\|} - \alpha\right) > 0$. Reduce δ if necessary so that $\delta \leq \theta$. Then

$$\alpha_n \geq \frac{1}{\|F'(x^*)^{-1}\|} - \gamma\delta \geq \alpha > \frac{1+\rho}{2} m_2.$$

Notice that a general norm is still valid until this point. From the proof of Theorem-2, it can be obtained that

$$\left\| I - \alpha_n^{-1} F^{'}(x^*) \right\|_{\infty} \leq \frac{(1+\rho)m_2}{\alpha_n} - 1 \leq \frac{(1+\rho)m_2}{\alpha} - 1 < 1$$

because in ∞-norm,

$$\rho = \left\|D^{-1}E\right\|_{\infty} = \max_i \frac{\sum\limits_{j=1, j\neq i}^{n} |f_{ij}|}{f_{ii}} < m_1/m_2.$$

The local linear convergence of (2.20) and the final δ then follows from Theorem-2 in ∞-norm. ∎

Notice that the norm in the computation of α_n can be different from the norm of the local convergence analysis. In the local convergence, ∞-norm is used whereas any other norm can be employed in obtaining α_n. From this can arise two different Lipschitz constants, each for the correscponding norm. Suppose α_n is computed in a p-norm, $1 \leq p < \infty$, with the associated Lipschitz constant γ_p whereas the Lipschitz constant associated with the ∞-norm is denoted by γ, then the local neighborhood $\mathcal{B}(\delta)$ which satisfies Theorem-3 is given by

$$\delta = min\left\{\frac{2(1-\delta_1)}{\gamma_p(1+\delta_1)\left\|F'(x^*)^{-1}\right\|_p}, \frac{1}{\gamma}\left(\frac{1}{\left\|F(x^*)^{-1}\right\|_{\infty}} - \alpha\right)\right\}$$

where α and δ_1 can be concluded from the proof of Theorem-3 as $\delta_1 = \dfrac{(1+\rho)m_2}{\alpha_n} - 1$ and $\alpha > \dfrac{1+\rho}{2}m_2$. Despite the local convergence, it is worth noting that the application of Theorem-3 is restricted to a relatively smaller neighbourhood than many other quasi-Newton methods. It also requires the condition of Jacobian matrix which is not only diagonal dominant, but also the restriction of the strength of the diagonal dominance in each row of the matrix. Another necesseary condition is the limitation of the condition number of the Jacobian matrix. Given that $\|F'(x^*)^{-1}\| < 1/\alpha$ and $m_2 < \|F'(x^*)\|_{\infty} < m_1 + m_2$, the upper bound of the condition number is limited by

$$\kappa_{\infty}(F(x^*) = \left\|F'(x^*)^{-1}\right\|_{\infty}\|F'(x^*)\|_{\infty} < \frac{m_1+m_2}{\alpha} < \frac{2(m_1+m_2)}{(1+\rho)m_2} = \frac{2}{1+\rho}(1+m_1/m_2)$$

The strongest diagonal dominant matrix will give $\rho = 0$, for which the condition number is $\kappa_{\infty}(F(x^*)) < 2(1+m_1/m_2)$.

2.4.2 A Newton Method with Finite Difference Jacobian

The remaining sections in this chapter describes some other alternatives of nonlinear solvers. Consider again the system of nonlinear equations (2.11). The Newton sequence

$$x_{n+1} = x_n - F^{'}(x_n)^{-1}F(x_n),$$

can be written using the following algorithm :

```
1. Initialize $x$, and termination tolerance $\tau_r, \tau_a$
   $r_0 = \|F(x)\|$
2. Do while $\|F(x)\| > \tau_r r_0 + \tau_a$
      (a) Compute $F^{'}(x)$
      (b) Factor $F^{'}(x) = LU$.
      (c) Solve $LUs = -F(x)$
      (d) find a step length $\lambda$
      (e) $x = x + \lambda s$
      (f) Evaluate $F(x)$.
   End while
```

where the factorisation of the Jacobian is carried out by using the LU Decomposition. In the step $2a$, one way to compute the Jacobian $F^{'}(x)$ is by applying a Finite Difference scheme where the jth column of the approximation is

$$(\nabla_h F)(x)_j = \begin{cases} \dfrac{F(x + h\sigma_j e_j) - F(x)}{\sigma_j h}, & x_j \neq 0 \\ \\ \dfrac{F(x + he_j) - F(x)}{h}, & x_j = 0 \end{cases} \tag{2.21}$$

e_j in the above approximation is the unit vector in the jth coordinate direction. The term $h\sigma_j$ (or just h for $x_j = 0$) used to perturb the function will be referred to as the small perturbation in Chapter-4 and Chapter-6. Each column of $(\nabla_h F)$ requires one new function evaluation, therefore the cost of a finite difference Jacobian is N function evaluations. [66] suggests that the difference increment h should be no smaller than the square root of the inaccuracy in the function, whereas the scaling in the j-th column uses

$$\sigma_j = \max(|(x)_j|, 1) sgn((x)_j) \tag{2.22}$$

with

$$sgn(z) = \begin{cases} z/|z| & \text{if } z \neq 0, \\ 1 & \text{if } z = 0. \end{cases} \tag{2.23}$$

If the initial iterate is too far from the root, choosing $\lambda = 1$ in the step $2d$ may not result in a better updating. Line search methods like the Armijo rule [5] are used to search for a decrease in $\|F\|$ along the segment $[x, x + s]$. One way to compute λ is by minimizing a polymomial model [66] over a sequence λ_n with $\lambda_0 = 1$ and $1/10 \leq \lambda_{m+1}/\lambda_m \leq 1/2$.

```
Line search algorithm
α = 10^{-4}
λ = 1
while ‖F(x + λd)‖ > (1 − αλ) ‖F(x+)‖ do
    λ ← σλ, where σ ∈ [1/10, 1/2] is computed by minimizing the polynomial
     model of ‖F(x + λd)‖^2 .
end while
x ← x + λd
```

The above approach uses $\lambda_1 = 1/2$. To compute λ_m for $m > 1$, a parabola is fitted to the data $\sigma(0), \sigma(\lambda_m)$, and $\sigma(\lambda_{m-1})$. λ_m is the minimum of this parabola on the interval $[\lambda_{m-1}/10, \lambda_{m-1}/2]$. The reference [65] shows the details of this method as well as other ways to implement a line search.

2.4.3 Newton-GMRES method

This method is a synergistic combination of Newton-type methods for superlinearly convergent solution of nonlinear equations and a Krylov subspace method for solving the Newton correction equations. Denote by J the Jacobian of F, and v by the Newton correction, the following algorithm shows how the Newton-type method and the GMRES method are combined.

```
[Newton-GMRES Algorithm]
  u_0 given, i = −1
  REPEAT
          i = i + 1
    (a).  Solve J(u_i)δ(i) = −F(u_i) by GMRES
    (b).  u_{i+1} = u_i + α_i δ(i)
  UNTIL convergence
```

where α_i is computed by using a line search method.

The link between the two methods is the Jacobian-vector product, which can be approximately computed without forming the Jacobian matrix [71]. The product of the Jacobian with a vector v may be approximated with a finite difference scheme [16] :

$$J(u).v \approx \frac{F(u+\sigma v)-F(u)}{\sigma}.$$

For the scaling σ, [65] suggests :

$$\sigma = \frac{h}{\left|x^T w\right|^2} \max(\left|x^T w\right|, |w|) sgn(x^T w)$$

where h is roughly the square root of the error in F.
With this approach, the algorithm belongs to the class of Inexact Newton methods [65]. The Newton correction can then be found by substituting the step (a) in the Newton-GMRES algorithm with the following algorithm [22] :

[Inexact-GMRES Algorithm]

given $\epsilon > 0$ the tolerance of the stopping criterion and $\delta_0 \in \Re^n$ the initial guess,

```
(Step-1).  $\widetilde{r}_0 = -F - q_1, q_1 = \frac{F(u+\sigma_0\delta_0)-F(u)}{\sigma_0}$
           $\widetilde{\beta} = \|\widetilde{r}_0\|_2, \widetilde{v}_1 = \frac{\widetilde{r}_0}{\widetilde{\beta}}$
           $k = 0$
(Step-2).  REPEAT
           $k = k+1$
              $(a)$  $q_{k+1} = \frac{F(u+\sigma_k\widetilde{v}_k)-F(u)}{\sigma_k}$
                     $\widetilde{w}_{k+1} = q_{k+1} - \sum_{m=1}^{k} \widetilde{h}_{m,k}\widetilde{v}_m$
                        with $\widetilde{h}_{m,k} = (q_{k+1}, \widetilde{v}_m)$        $(m = 1, ..., k)$
                     $\widetilde{h}_{k+1,k} = \|\widetilde{w}_{k+1}\|_2$
                     $\widetilde{v}_{k+1} = \frac{\widetilde{w}_{k+1}}{\widetilde{h}_{k+1,k}}$
              $(b)$  compute $\widetilde{\rho}_k = \min_{y\in\Re^k} \left\|\widetilde{\beta}e_1 - \widetilde{\overline{H}}_k y\right\|_2$
           UNTIL $\widetilde{\rho}_k \leq \epsilon$
(Step-3).  $K = k$
           compute $\widetilde{y}_K, \widetilde{z}_K = \widetilde{V}_K\widetilde{y}_K$ and $\widetilde{\delta}_K = \delta_0 + \widetilde{z}_K$
```

where

$$(\widetilde{\overline{H}}_k)_{m,l} = \begin{cases} \widetilde{h}_{m,l} & 1 \leq mk+1, ml \leq m \leq k \\ 0 & \text{otherwise} \end{cases}$$

The restart version of the above algorithm can be easily arranged following the restart-GMRES procedure in solving linear equations. The details are also given in [22].

2.4.4 Nonlinear Conjugate Gradient

This method stems from the Conjugate Gradient (CG) class. It is different from the linear CG [106] [102] in the way it performs the line search and search direction. The derivation of this nonlinear version is given in [106]. Let λ be the interface variable and $D(\lambda)$ be the nonlinear defect equation (2.8) whose root is being sought . The iteration of the nonlinear CG method is demonstrated as follows :

$$d_{(0)} = r_{(0)} = -D(\lambda_{(0)})$$

$$\varphi_{(i)} = -\overline{\nu}\frac{D(\lambda_{(i)})^T d_{(i)}}{D(\lambda_{(i)} + \overline{\nu} d_{(i)})^T d_{(i)} - D(\lambda_{(i)})^T d_{(i)}} \tag{2.24}$$

$$\lambda_{(i+1)} = \lambda_{(i)} + \varphi_{(i)} d_{(i)}$$

$$r_{(i+1)} = -D(\lambda_{(i+1)})$$

$$\beta^{FR}_{(i+1)} = \frac{r^T_{(i+1)} r_{(i+1)}}{r^T_{(i)} r_{(i)}} \qquad \text{or} \qquad \beta^{PR}_{(i+1)} = \frac{r^T_{(i+1)}\left(r_{(i+1)} - r_{(i)}\right)}{r^T_{(i)} r_{(i)}}$$

$$d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)} d_{(i)} \tag{2.25}$$

A secant approach is employed in the line search (2.24) by performing an extra computation of (2.2) using a slightly perturbed iterate $(\lambda_{(i)} + \overline{\nu} d_{(i)})$, where $\overline{\nu}$ is chosen very small and $d_{(i)}$ is the new direction. For the search direction (2.25), a choice between Fletcher-Reeves (FR) [43] and Polak-Ribiere (PR) [70] formula can be used, where the first is faster but less stable compared to the latter. Instead of computing a Jacobian matrix, it uses two residuals, one being the latest residual $D(\lambda_{(i)})$, and the other is $D(\lambda_{(i)} + \overline{\nu} d_{(i)})$ which is associated with the small perturbation to $\lambda_{(i)}$. Since an evaluation of defect D needs a domain decomposition iteration, each nonlinear conjugate gradient iteration-i will need two domain decomposition iterations.

Chapter 3

Nonoverlapping Domain Decomposition Methods

This chapter outlines some nonoverlapping domain decomposition methods which are commonly used in iterative procedures of numerical methods for solving ellliptical partial differential equations. It starts with the Schur complement problem as the general framework of nonoverlapping domain decomposition method in linear algebraic systems. It is then followed by the algebraic derivation of some classical nonoverlapping schemes such as Dirichlet-Neumann and Neumann-Neumann. After that, a section is presented to show that the defect correction scheme introduced in Chapter-2 is a preconditioner which is beneficial for parallel implementations. The last part of this chapter describes the basic formulation of nonconforming methods when dealing with nonmatching grids at subdomain interfaces.

The novel feature in this chapter aims to show the setting of iterative procedures of the defect correction scheme in the mortar element framework.

3.1 Schur Complements

Consider the equation resulting from the discretisation of process for two subdomains problem [95] :

$$\begin{bmatrix} A_{\Omega_1} & 0 & A_{1\Gamma} \\ 0 & A_{\Omega_2} & A_{2\Gamma} \\ A_{\Gamma_1} & A_{\Gamma_2} & A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_\Gamma \end{bmatrix}$$

This reflects the algebraic equation relating to a partition of an entire domain Ω into two subdomains Ω_1 and Ω_2 with one nonoverlapping interface. The first and the second row represent the discretisation in the interior part of the subdomain-1 and subdomain-2, while the last row corresponds to that at the interface boundary [93].

The block matrices A_{Ω_1} and A_{Ω_2} correspond to the interior nodal points in Ω_1 and Ω_2 respectively whereas $A_{1\Gamma}$ and $A_{2\Gamma}$ are the block matrices corresponding to the interface nodal points associated with the discretisation in Ω_1 and Ω_2.

The third row signifies the discretisation at the interface boundary with $A_{\Gamma_1}, A_{\Gamma_2}$ and $A^{(1)}_{\Gamma\Gamma} + A_{\Gamma\Gamma}$ denoting the link to the nodal points in the interior of subdomain Ω_1, Ω_2 and interface Γ respectively. Note that the the right hand side f_Γ can be split into

$$f_\Gamma = f^{(1)}_\Gamma + f^{(2)}_\Gamma,$$

with $f^{(i)}_\Gamma = A_{\Gamma_i} u_i + A^{(i)}_{\Gamma\Gamma} u_\Gamma$.

Block factorisation of A is given by

$$A = \begin{bmatrix} A_{\Omega_1} & 0 & 0 \\ 0 & A_{\Omega_2} & 0 \\ A_{\Gamma_1} & A_{\Gamma_2} & I \end{bmatrix} \begin{bmatrix} I & 0 & A^{-1}_{\Omega_1} A_{1\Gamma} \\ 0 & I & A^{-1}_{\Omega_2} A_{2\Gamma} \\ 0 & 0 & S^{(1)} + S^{(2)} \end{bmatrix}$$

where $S^{(i)} = A^{(i)}_{\Gamma\Gamma} - A_{\Gamma_i} A^{-1}_{\Omega_i} A_{i\Gamma}$ is the Schur complements [54] of each subdomain.

The factorisation [107] results in a simpler form

$$\begin{bmatrix} I & 0 & A^{-1}_{\Omega_1} A_{1\Gamma} \\ 0 & I & A^{-1}_{\Omega_2} A_{2\Gamma} \\ 0 & 0 & S^{(1)} + S^{(2)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} A^{-1}_{\Omega_1} & 0 & 0 \\ 0 & A^{-1}_{\Omega_2} & 0 \\ -A_{\Gamma_1} A^{-1}_{\Omega_1} & -A_{\Gamma_2} A^{-1}_{\Omega_2} & I \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_\Gamma \end{bmatrix}$$

where the first and the second columns imply the solves of the first and second subdomain given Dirichlet condition while the last column signify an interface equation coupling the solution of the two subdomains [107].

The last row of the matrix equation, which is also known as the Schur complement equation, reads

$$(S^{(1)} + S^{(2)}) u_\Gamma = f^{(1)}_\Gamma + f^{(2)}_\Gamma - A_{\Gamma_1} A^{-1}_{\Omega_1} f_1 - A_{\Gamma_2} A^{-1}_{\Omega_2} f_2 \tag{3.1}$$

Rather than solving the whole domain problem $Au = f$, the nonoverlapping domain decomposition is centered on solving the Schur complement equation. When the Schur complement in (3.1) are explicitly computed, it results in a direct method called the direct substructuring [107]. The following sections present some existing preconditioners for the iterative substructuring problem resulting from this Schur complement problem.

3.2 Classical Iterative Substructuring Methods

This section concerns the Dirichlet-Neumann and the Neumann-Neumann method which are among the earliest nonoverlapping domain decomposition methods. The derivation of the methods have been shown in some basic references such as [107] and [117]. The derivation shown here is combined from such references.

3.2.1 The Dirichlet-Neumann Method

In this method, S in the Schur equation (3.1) is preconditioned by ${S^{(1)}}^{-1}$ or ${S^{(2)}}^{-1}$. Applying ${S^{(1)}}^{-1}$, it follows that

$$ {S^{(1)}}^{-1}(S^{(1)} + S^{(2)})u_\Gamma = {S^{(1)}}^{-1}(f_\Gamma - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Gamma_2}A_{\Omega_2}^{-1}f_2) $$

or

$$ (I + {S^{(1)}}^{-1}S^{(2)})u_\Gamma = {S^{(1)}}^{-1}(f_\Gamma - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Gamma_2}A_{\Omega_2}^{-1}f_2) $$

By splitting technique, the following iterative equation can be formed:

$$ \begin{aligned} u_\Gamma^{n+1} &= -{S^{(1)}}^{-1}S^{(2)}u_\Gamma^n + {S^{(1)}}^{-1}(f_\Gamma - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Gamma_2}A_{\Omega_2}^{-1}f_2) \\ &= {S^{(1)}}^{-1}(-S^{(2)}u_\Gamma^n + (f_\Gamma - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Gamma_2}A_{\Omega_2}^{-1}f_2)) \end{aligned} $$

Substituting $S^{(2)}$ yields

$$ \begin{aligned} u_\Gamma^{n+1} &= {S^{(1)}}^{-1}(-A_{\Gamma\Gamma}^{(2)}u_\Gamma^n + A_{\Gamma_2}A_{\Omega_2}^{-1}A_{2\Gamma}u_\Gamma^n + f_\Gamma - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Gamma_2}A_{\Omega_2}^{-1}f_2) \\ &= {S^{(1)}}^{-1}(-A_{\Gamma\Gamma}^{(2)}u_\Gamma^n - A_{\Gamma_2}A_{\Omega_2}^{-1}(f_2 - A_{2\Gamma}u_\Gamma^n) - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 + f_\Gamma) \end{aligned} $$

The term $A_{\Omega_2}^{-1}(f_2 - A_{2\Gamma}u_\Gamma^n)$ in the bracket implies the solve of the second boundary problem given Dirichlet boundary condition of u_Γ^n of the previous iteration. It follows that

$$ \begin{aligned} u_\Gamma^{n+1} &= {S^{(1)}}^{-1}(-A_{\Gamma\Gamma}^{(2)}u_\Gamma^n - A_{\Gamma_2}u_2^{n+1} - A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 + f_\Gamma) \\ &= {S^{(1)}}^{-1}(-A_{\Gamma\Gamma}^{(2)}u_\Gamma^n - A_{\Gamma_2}u_2^{n+1} + f_\Gamma) - {S^{(1)}}^{-1}A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 \end{aligned} \tag{3.2} $$

This iteration does not need to compute the inverse of $S^{(1)}$. Instead, it will deal with the inverse of matrix A_1 which is the discretisation matrix of subdomain Ω_1 with Neumann condition :

$$A_1 = \begin{bmatrix} A_{\Omega_1} & A_{1\Gamma} \\ A_{\Gamma_1} & A^{(1)}_{\Gamma\Gamma} \end{bmatrix}$$

Let us perform an iterative procedure for u_1 with the information of the recent interface value :

$$\begin{aligned} u_1^{n+1} &= A_{\Omega_1}^{-1}(f_1 - A_{1\Gamma}u_\Gamma^{n+1}) = A_{\Omega_1}^{-1}f_1 - A_{\Omega_1}^{-1}A_{1\Gamma}u_\Gamma^{n+1}) \\ &= A_{\Omega_1}^{-1}f_1 + A_{\Omega_1}^{-1}A_{1\Gamma}S^{(1)^{-1}}(A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - f_\Gamma + A_{\Gamma_2}u_2^{n+1} + A^{(2)}_{\Gamma\Gamma}u_\Gamma^n) \\ &= A_{\Omega_1}^{-1}f_1 + A_{\Omega_1}^{-1}A_{1\Gamma}S^{(1)^{-1}}A_{\Gamma_1}A_{\Omega_1}^{-1}f_1 - A_{\Omega_1}^{-1}A_{1\Gamma}S^{(1)^{-1}}(f_\Gamma - A_{\Gamma_2}u_2^{n+1} - A^{(2)}_{\Gamma\Gamma}u_\Gamma^n) \end{aligned}$$

A_1 can be factorised as

$$A_1 = \begin{bmatrix} I & 0 \\ A_{\Gamma_1}A_{\Omega_1}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{\Omega_1} & 0 \\ 0 & S^{(1)} \end{bmatrix} \begin{bmatrix} I & A_{\Omega_1}^{-1}A_{1\Gamma} \\ 0 & I \end{bmatrix}$$

Therefore the inverse is given by

$$A_1^{-1} = \begin{bmatrix} I & -A_{\Omega_1}^{-1}A_{1\Gamma} \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{\Omega_1}^{-1} & 0 \\ 0 & S^{(1)^{-1}} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{\Gamma_1}A_{\Omega_1}^{-1} & I \end{bmatrix}$$

$$= \begin{bmatrix} A_{\Omega_1}^{-1}(I + A_{1\Gamma}S^{(1)^{-1}}A_{\Gamma_1}A_{\Omega_1}^{-1}) & -A_{\Omega_1}^{-1}A_{1\Gamma}S^{(1)^{-1}} \\ -S^{(1)^{-1}}A_{\Gamma_1}A_{\Omega_1}^{-1} & S^{(1)^{-1}} \end{bmatrix}$$

Combining the iterate of subdomain-1 and interface together gives

$$\begin{bmatrix} u_1 \\ u_\Gamma \end{bmatrix}^{n+1} = A_1^{-1} \begin{bmatrix} f_1 \\ f_\Gamma - A_{\Gamma_2}u_2^{n+1} - A^{(2)}_{\Gamma\Gamma}u_\Gamma^n \end{bmatrix}$$

By expanding the matrix A_1, the above is the solution to the problem :

$$\begin{bmatrix} A_{\Omega_1} & A_{1\Gamma} \\ A_{\Gamma_1} & A^{(1)}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_1 \\ u_\Gamma \end{bmatrix}^{n+1} = \begin{bmatrix} f_1 \\ f_\Gamma - A_{\Gamma_2}u_2^{n+1} - A^{(2)}_{\Gamma\Gamma}u_\Gamma^n \end{bmatrix}$$

The iterative procedure can be put in the following algorithm

```
**Do** $n = 1, 2, .....$
    Solve $A_{\Omega_2} u_2^{n+1} = f_2 - A_{2\Gamma} u_\Gamma^n$
    Solve $\begin{bmatrix} A_{\Omega_1} & A_{1\Gamma} \\ A_{\Gamma_1} & A_{\Gamma\Gamma}^{(1)} \end{bmatrix} \begin{bmatrix} u_1 \\ u_\Gamma \end{bmatrix}^{n+1} = \begin{bmatrix} f_1 \\ f_\Gamma - A_{\Gamma_2} u_2^{n+1} - A_{\Gamma\Gamma}^{(2)} u_\Gamma^n \end{bmatrix}$
**Until converge**
```

If the governing equation is that of a Poisson problem, it can be shown that a functional of normal derivative approximation at the interface can be given by : $f_\Gamma^{(i)} - A_{\Gamma_i} u_i - A_{\Gamma\Gamma}^{(i)} u_\Gamma$.

This implies that the second equation in the algebraic is equivalent to the equality of normal derivative :

$$\frac{\partial u_1^{n+1}}{\partial n_1} = -\frac{\partial u_2^{n+1}}{\partial n_2}$$

In the Poisson problem, the iterative method in the continuous system can be written as follows

```
$k := 0;$ Initial guess :$\lambda^{(0)}$;
do
    Solve $\begin{cases} \Delta u_2^{(k+1)} = f & \text{in } \Omega_2 \\ u_2^{(k+1)} = g & \text{on } \partial\Omega_2 \cap \partial\Omega \\ u_2^{(k+1)} = \lambda^{(k)} & \text{on } \Gamma \end{cases}$
    Solve $\begin{cases} \Delta u_1^{(k+1)} = f & \text{in } \Omega_1 \\ u_1^{(k+1)} = g & \text{on } \partial\Omega_1 \cap \partial\Omega \\ \dfrac{\partial u_1^{(k+1)}}{\partial n} = \dfrac{\partial u_2^{(k+1)}}{\partial n} & \text{on } \Gamma \end{cases}$
    $\lambda^{(k+1)} := u_1^{(k+1)}|_\Gamma;$
    $k := k + 1;$
Until converged;
```

The standard Dirichlet-Neumann which uses the control parameter θ in determining the Dirichlet boundary value :

$$\lambda^{(k+1)} := \theta u_2^{(k+1)}|_\Gamma + (1 - \theta)\lambda^{(k)};$$

can be obtained by applying the left preconditioner $\theta S^{(1)^{-1}}$ to the Schur equation. This is achieved by a relevant splitting method.

As opposed to the left preconditioner, one can also use $S^{(1)^{-1}}$ or $S^{(2)^{-1}}$ as the right preconditioner. This is commonly used for the preconditioner of Krylov subspaces methods.

3.2.2 The Neumann-Neumann Method

Another widely used preconditioner for the Schur equation (3.1) is the Neumann-Neumann preconditioner : $\theta_1 S^{(1)^{-1}} + \theta_2 S^{(2)^{-1}}$.

Recall the Schur equation :

$$(S^{(1)} + S^{(2)})u_\Gamma = f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2$$

Applying the Neumann-Neumann preconditioner :

$$(\theta_1 S^{(1)^{-1}} + \theta_2 S^{(2)^{-1}})(S^{(1)} + S^{(2)})u_\Gamma = (\theta_1 S^{(1)^{-1}} + \theta_2 S^{(2)^{-1}})(f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2)$$

A splitting method results in :

$$\begin{aligned}
u_\Gamma^{n+1} = \;& u_\Gamma^n - (\theta_1 S^{(1)^{-1}} + \theta_2 S^{(2)^{-1}})(S^{(1)} + S^{(2)})u_\Gamma^n + (\theta_1 S^{(1)^{-1}} + \theta_2 S^{(2)^{-1}})(f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) \\
= \;& u_\Gamma^n - \theta_1 S^{(1)^{-1}} S^{(2)} u_\Gamma^n + \theta_1 S^{(1)^{-1}} (f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) - \theta_1 S^{(1)^{-1}} S^{(1)} u_\Gamma^n \\
& -\theta_2 S^{(2)^{-1}} S^{(1)} u_\Gamma^n + \theta_2 S^{(2)^{-1}} (f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) - \theta_2 S^{(1)^{-1}} S^{(2)} u_\Gamma^n \\
= \;& u_\Gamma^n - \theta_1 S^{(1)^{-1}} S^{(2)} u_\Gamma^n + \theta_1 S^{(1)^{-1}} (f_\Gamma - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) - \theta_1 S^{(1)^{-1}} S^{(1)} u_\Gamma^n + \theta_1 S^{(1)^{-1}} (-A_{\Gamma_1} A_{\Omega_1}^{-1} f_1) \\
& -\theta_2 S^{(2)^{-1}} S^{(1)}) u_\Gamma^n + \theta_2 S^{(2)^{-1}} (f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1) - \theta_2 S^{(1)^{-1}} S^{(2)} u_\Gamma^n + \theta_2 S^{(2)^{-1}} (-A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) \\
= \;& u_\Gamma^n + \theta_1 S^{(1)^{-1}} (-S^{(2)} u_\Gamma^n + f_\Gamma - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2) + \theta_1 S^{(1)^{-1}} (-S^{(1)} u_\Gamma^n - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1) \\
& +\theta_2 S^{(2)^{-1}} (-S^{(1)} u_\Gamma^n + f_\Gamma - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1) + \theta_2 S^{(2)^{-1}} (-S^{(2)} u_\Gamma^n - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2)
\end{aligned}$$

Continuing the the simplification

$$\begin{aligned}
u_\Gamma^{n+1} = \;& u_\Gamma^n + \theta_1 S^{(1)^{-1}} (-A_{\Gamma\Gamma}^{(2)} u_\Gamma^n - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2 + A_{\Gamma_2} A_{\Omega_2}^{-1} A_{2\Gamma} u_\Gamma^n + f_\Gamma) \\
& +\theta_1 S^{(1)^{-1}} (-A_{\Gamma\Gamma}^{(1)} u_\Gamma^n - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 + A_{\Gamma_1} A_{\Omega_1}^{-1} A_{1\Gamma} u_\Gamma^n) \\
& +\theta_2 S^{(2)^{-1}} (-A_{\Gamma\Gamma}^{(1)} u_\Gamma^n - A_{\Gamma_1} A_{\Omega_1}^{-1} f_1 + A_{\Gamma_1} A_{\Omega_1}^{-1} A_{1\Gamma} u_\Gamma^n + f_\Gamma) \\
& +\theta_2 S^{(2)^{-1}} (-A_{\Gamma\Gamma}^{(2)} u_\Gamma^n - A_{\Gamma_2} A_{\Omega_2}^{-1} f_2 + A_{\Gamma_2} A_{\Omega_2}^{-1} A_{2\Gamma} u_\Gamma^n)
\end{aligned}$$

leads to

$$\begin{aligned} u_\Gamma^{n+1} = \; & u_\Gamma^n + \theta_1 S^{(1)^{-1}}(-A_{\Gamma\Gamma}^{(2)} u_\Gamma^n - A_{\Gamma_2} u_2^{n+1} - A_{\Gamma\Gamma}^{(1)} u_\Gamma^n - A_{\Gamma_1} u_1^{n+1} + f_\Gamma) \\ & + \theta_2 S^{(2)^{-1}}(-A_{\Gamma\Gamma}^{(1)} u_\Gamma^n - A_{\Gamma_1} u_1^{n+1} - A_{\Gamma\Gamma}^{(2)} u_\Gamma^n - A_{\Gamma_2} u_2^{n+1} + f_\Gamma) \end{aligned} \tag{3.3}$$

The above equation can now be compared with (3.2). In (3.2), a Dirichlet solves in Ω_2 with source f_2 is followed by a Neumann solves in Ω_1 with the source terms f_1 and $f_\Gamma^{(1)}$ and the normal derivative $f_\Gamma^{(2)} - A_{\Gamma_2} u_2^{n+1} - A_{\Gamma\Gamma}^{(2)} u_\Gamma^n$. With a careful inspection of (3.3), it can be verified that the second term implies two Dirichlet solves, each in Ω_1 and Ω_2 with source f_1 and f_2 respectively which are followed by a Neumann solve in Ω_1 with zero source (($f_1 = 0$) and $f_\Gamma^{(1)} = 0$) and the normal derivative $f_\Gamma^{(1)} - A_{\Gamma_1} u_1^{n+1} - A_{\Gamma\Gamma}^{(1)} u_\Gamma^n + f_\Gamma^{(2)} - A_{\Gamma_2} u_2^{n+1} - A_{\Gamma\Gamma}^{(2)} u_\Gamma^n$.

In a Poisson problem, the approximation of this normal derivative is a functional of $\dfrac{\partial u_1^{n+1}}{\partial n_1} + \dfrac{\partial u_2^{n+1}}{\partial n_2}$.

Similarly, it can be checked from the third term in (3.3), that two Dirichlet solves (exactly the same ones as in the second term) are followed by a Neumann solve in Ω_2 with zero source and the normal derivative of $\dfrac{\partial u_1^{n+1}}{\partial n_1} + \dfrac{\partial u_2^{n+1}}{\partial n_2}$.

This preconditioner in fact gives rise to two Dirichlet solves and followed by two Neumann solves, which can be implemented in two parallel Dirichlet-Neumann computations.

Its algebraic iterative algorithm is presented below

```
**Do** $n = 1, 2, .....$
  $\forall i = 1, 2$: Solve  $A_{\Omega_i} u_i^{n+1} = f_i - A_{i\Gamma} u_\Gamma^n$
  $\forall i = 1, 2$: Solve  $\begin{bmatrix} A_{\Omega_1} & A_{1\Gamma} \\ A_{\Gamma_1} & A_{\Gamma\Gamma}^{(1)} \end{bmatrix} \begin{bmatrix} \psi_i \\ \psi_{\Gamma i} \end{bmatrix}^{n+1} = \begin{bmatrix} 0 \\ f_\Gamma - A_{\Gamma_1} u_1^{n+1} - A_{\Gamma_2} u_2^{n+1} - A_{\Gamma\Gamma}^{(1)} u_\Gamma^n - A_{\Gamma\Gamma}^{(2)} u_\Gamma^n \end{bmatrix}$
  $u_\Gamma^{n+1} = u_\Gamma^n + \theta_1 \psi_{\Gamma 1} + \theta_2 \psi_{\Gamma 2}$
**Until converge**
```

whereas the following demonstrates the procedure in Poisson operator :

```
$k := 0;$ Initial guess :$\lambda^{(0)}$;
do
    Solve $\left\{ \begin{array}{lll} \Delta u_i^{(k+1)} = f & \text{in } \Omega_i \\ u_i^{(k+1)} = g & \text{on } \partial\Omega_i \cap \partial\Omega \\ u_i^{(k+1)} = \lambda^{(k)} & \text{on } \Gamma \end{array} \right\} i = 1, 2$

    Solve $\left\{ \begin{array}{ll} \Delta \psi_i^{(k+1)} = 0 & \text{in } \Omega_i \\ \psi_i^{(k+1)} = 0 & \text{on } \partial\Omega_i \cap \partial\Omega \\ \dfrac{\partial \psi_i^{(k+1)}}{\partial n_i} = \dfrac{\partial u_1^{(k+1)}}{\partial n_1} + \dfrac{\partial u_2^{(k+1)}}{\partial n_2} & \text{on } \Gamma \end{array} \right\} i = 1, 2$

    $\lambda^{(k+1)} := \lambda^{(k)} + \theta_1 \psi_1^{(k+1)}|_\Gamma + \theta_2 \psi_1^{(k+1)}|_\Gamma;$
    $k := k + 1;$
Until converged;
```

3.3 Parallel-Motivated Preconditioners

This section proposes a nonoverlapping domain decomposition preconditioner which can accommodate different numerical techniques for each subdomain problem. Each subdomain can be meshed, discretised, and solved conveniently and independently without any need of knowledge of what occurs in other subdomains. The coupling is only carried out along the interface updating of the Dirichlet condition.

Consider the system of equations to be solved,

$$Au = f$$

Let the algebraic equations is arranged in the following way to reflect the decomposition :

$$\begin{bmatrix} A_{\Omega_1} & 0 & . & 0 & A_{1\Gamma} \\ 0 & A_{\Omega_2} & 0 & . & A_{2\Gamma} \\ . & . & . & . & . \\ 0 & . & 0 & A_{\Omega_N} & A_{N\Gamma} \\ A_{\Gamma_1} & A_{\Gamma_2} & . & A_{\Gamma_N} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ . \\ u_N \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ . \\ f_N \\ f_\Gamma \end{bmatrix}$$

The index $i = 1, ..., N$ refers to the subdomain-i after the nonoverlapping partition of the whole domain Ω into N subdomains. The interface between subdomain is grouped into index block Γ.

Consider an iterative procedure of substructuring computation :

$$\begin{array}{rll}
u_1^{n+1} = & u_1^n + A_{\Omega_1}^{-1}(f_1 - A_{\Omega_1}u_1^n - A_{1\Gamma}u_\Gamma^n) & = u_1^n + A_{\Omega_1}^{-1}(f_1 - A_1u^n) \\
u_2^{n+1} = & u_2^n + A_{\Omega_2}^{-1}(f_2 - A_{\Omega_2}u_2^n - A_{2\Gamma}u_\Gamma^n) & = u_2^n + A_{\Omega_2}^{-1}(f_2 - A_2u^n) \\
. & . & . \\
u_N^{n+1} = & u_N^n + A_{\Omega_N}^{-1}(f_N - A_{\Omega_N}u_N^n - A_{N\Gamma}u_\Gamma^n) & = u_N^n + A_{\Omega_N}^{-1}(f_N - A_Nu^n) \\
u_\Gamma^{n+1} = & u_\Gamma^n + \alpha(f_\Gamma - A_{\Gamma\Gamma}u_\Gamma^n - A_{\Gamma 1}u_1^{n+1} - A_{\Gamma 2}u_2^{n+1} ... - A_{\Gamma N}u_N^{n+1}) &
\end{array}$$

where the matrix α in the last equation is not yet defined.

This procedure runs from 1 to N in the block Jacobi fashion, hence can be performed in parallel. The last equation, where the coupling takes place, uses the latest interior coefficient, with α being an arbitrary matrix. The equation can be modified by substituting the interior coefficient :

$$\begin{aligned}
u_\Gamma^{n+1} = {} & u_\Gamma^n + \alpha(f_\Gamma - A_{\Gamma\Gamma}u_\Gamma^n - A_{\Gamma 1}(u_1^n + A_{\Omega_1}^{-1}(f_1 - A_1u^n)) - A_{\Gamma 2}(u_2^n + A_{\Omega_2}^{-1}(f_2 - A_2u^n)) \\
& - ... - A_{\Gamma N}(u_N^n + A_{\Omega_N}^{-1}(f_N - A_Nu^n))) \\
= {} & u_\Gamma^n + \alpha(f_\Gamma - A_{\Gamma\Gamma}u_\Gamma^n - A_{\Gamma 1}u_1^n - A_{\Gamma 2}u_2^n - ... - A_{\Gamma N}u_N^n) + \alpha(-A_{\Gamma 1}A_{\Omega_1}^{-1}(f_1 - A_1u^n) \\
& - A_{\Gamma 1}A_{\Omega_2}^{-1}(f_2 - A_2u^n) - ... - A_{\Gamma N}A_{\Omega_N}^{-1}(f_N - A_Nu^n)) \\
= {} & u_\Gamma^n + \alpha(f_\Gamma - A_\Gamma u^n) - \alpha A_{\Gamma 1}A_{\Omega_1}^{-1}(f_1 - A_1u^n) - \alpha A_{\Gamma 1}A_{\Omega_2}^{-1}(f_2 - A_2u^n) \\
& - ... - \alpha A_{\Gamma N}A_{\Omega_N}^{-1}(f_N - A_Nu^n)
\end{aligned}$$

where $A_i = [0...0 \; A_{\Omega_i} \; 0...0 \; A_{i\Gamma}]$ is the i-th row block of matrix A. Along with the other N iterative equations, the entire procedure can be brought as follows:

$$\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ . \\ u_N^{n+1} \\ u_\Gamma^{n+1} \end{bmatrix} = \begin{bmatrix} u_1^n \\ u_2^n \\ . \\ u_N^n \\ u_\Gamma^n \end{bmatrix} + \begin{bmatrix} A_{\Omega_1}^{-1} & 0 & . & . & 0 \\ 0 & A_{\Omega_2}^{-1} & 0 & . & 0 \\ . & . & . & . & . \\ 0 & . & 0 & A_{\Omega_N}^{-1} & 0 \\ -\alpha A_{\Gamma 1}A_{\Omega_1}^{-1} & -\alpha A_{\Gamma 2}A_{\Omega_2}^{-1} & . & -\alpha A_{\Gamma N}A_{\Omega_N}^{-1} & \alpha \end{bmatrix} \begin{bmatrix} f_1 - A_1u^n \\ f_2 - A_2u^n \\ . \\ f_N - A_Nu^n \\ f_\Gamma - A_\Gamma u^n \end{bmatrix}$$

or in a simple Richardson iteration

$$u^{n+1} = u^n + M(f - Au^n)$$

where the matrix M is given by

$$M = \begin{bmatrix} A_{\Omega_1}^{-1} & 0 & . & . & 0 \\ 0 & A_{\Omega_2}^{-1} & 0 & . & 0 \\ . & . & . & . & . \\ 0 & . & 0 & A_{\Omega_N}^{-1} & 0 \\ -\alpha A_{\Gamma 1} A_{\Omega_1}^{-1} & \alpha A_{\Gamma 2} A_{\Omega_2}^{-1} & . & \alpha A_{\Gamma N} A_{\Omega_N}^{-1} & \alpha \end{bmatrix}$$

With this procedure, the discretisation and the solver in each subdomain and also at the interface can be built on its own without the intervention of other subdomain routine. From the meshing aspect, each subdomain can be meshed separately as well, the interface updating needs the knowledge of the nodal position of the interface neighbours. This iterative procedure will be consequently encouraging for independent parallel implementation.

However, the computational performance is subject to the choice of the matrix M. When M is not a good preconditioner for the whole algebraic system, the result might be daunting. It must be noted that the choice of M is only limited by the setting of matrix α. It is clear that if α is chosen as the identity matrix, the procedure is no more than the iterative substructuring without preconditioner, which is usually not convergent. The option of S_1^{-1} or S_2^{-1} for α will end up being a Dirichlet-Neumann method which is sequential in nature. In order to preserve the parallel and independent characteristic of the iteration, α must not be a function of A_{Ω_i} or $A_{i\Gamma}$.

One natural option is $A_{\Gamma\Gamma}^{-1}$ where this will lead to a mix of block Jacobi (parallel subdomain computation) and block Gauss-Seidel (the updating of the interface Dirichlet value needs the recent iterate of interface neighbour).

From this point of view, it can also be seen that the adaptive-α in the defect correction method uses a scalar value to represent the matrix α. The scalar itself is a variable which is a function of $A_{\Gamma i}$, $A_{\Gamma\Gamma}$ and the recent iterates at the interface and its neighbour. Hence it is a variable preconditioner.

3.4 Nonconforming methods

The classical nonoverlapping domain decomposition methods described in previous sections are derived on matching grids at the interfaces. It is often inconvenient to coordinate the decomposition and reassembly processes so that the constructed subdomain meshes would coincide when assembled. The use of nonconforming methods enables an analysis to be performed independently when nonmatching grids are used at common interfaces. The examples of non-conforming technique are the mortar finite element method [11] [10] and the FETI (Finite Element Tearing and Interconnect) method [40] [39].

The advantages of nonconforming methods have begun to be exploited. Several domain decomposition algorithms, that have been successfully applied to conforming finite element discretisations, have been extended to mortar discretisations of linear problems. Some domain decomposition preconditioners are described in [69]. FETI method has been applied in tackling nonlinear Navier Stokes equation where a domain decomposition algorithm is used to solve the resulting linearized Navier Stokes equation in each Picard iteration step [81]. It is also used in the coupling of linear Stokes equation in the fluid region and Darcy equation in the porous region [45].

Yotov in [132] [131] has used a mortar finite element method to couple different physical and numerical models in a single simulation where the resulting non-linear algebraic system is reduced to a non-linear interface problem which is solved by a multigrid scheme.

One of the recent application of mortar techniques is in multiscale methods. A research in [4] shows that the extension of mortar element method to multiscale frameworks has led to a mathematical formulation which is more flexible than existing multiscale finite element and variational multiscale methods. A new implementation of non-overlapping domain decomposition algorithm in this multiscale mortar finite element formulation is given in [48] where it improves the computational efficiency.

The following formulation shows the basic approach of nonconforming methods which are based on the primal hybrid variational principle [97] where Lagrange multipliers [84] are introduced to enforce compatibility at the interface nodes.

Consider the elliptical problem :

$$\begin{cases} \Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{3.4}$$

The weak formulation [63] of (3.4) reads :

$$\text{find } u \in V : a(u,v) = (f,v) \qquad \forall v \in V, \tag{3.5}$$

where

$$\begin{aligned}
(w, v) &:= \int_\Omega wv \\
a(w, v) &:= (\nabla w, \nabla v) \\
H^1(\Omega) &:= \{v \in L^2(\Omega) \mid D_j v \in L^2(\Omega), j = 1, ..., d\} \\
H_0^1(\Omega) &:= \{v \in H^1(\Omega) \mid v_{|\partial\Omega} = 0\} \\
V &:= H^1(\Omega)
\end{aligned} \tag{3.6}$$

and $L^2(\Omega)$ is the space of square-integrable functions [73] in Ω.

The idea of nonconforming methods is to approximate (3.5) by the following discrete problem (such as shown in [95]) :

$$\text{find } u_\delta \in V_\delta : \sum_{i=1}^{2} \int_{\Omega_i} \nabla u_\delta . \nabla v_\delta = \sum_{i=1}^{2} \int_{\Omega_i} f v_\delta \qquad \forall v_\delta \in V_\delta, \tag{3.7}$$

In the above formulation, $\delta > 0$ is a parameter describing the quality of the discretisation, and V_δ is a finite dimensional space that approximates $H_0^1(\Omega)$ without being contained into $C^0(\Omega)$, i.e. V_δ is a subspace of the following space :

$$Y_\delta := \{v_\delta \in L^2(\Omega) \mid v_{\delta|\Omega_i} \in Y_{i,\delta}, \quad i = 1, 2,$$

where, for each $i = 1, 2$, $Y_{i,\delta}$ is a finite dimensional subspace of the space V_i introduced in (3.6). The space V_δ contains functions in Y_δ that satisfy some kind of matching across Γ. Precisely, if $v_\delta \in V_\delta$ where $v_\delta^{(1)} \in Y_{1,\delta}$ and $v_\delta^{(2)} \in Y_{2,\delta}$ denote its restriction to Ω_1 and Ω_2, respectively, the following integral matching conditions should be satisfied for a certain fixed index i :

$$\int_\Gamma (v_\delta^{(1)} - v_\delta^{(2)}) \mu_\delta^{(i)} = 0 \qquad \forall \mu_\delta^{(i)} \in \Lambda_\delta^{(i)}, \tag{3.8}$$

where $\Lambda_\delta^{(i)}$ denotes the restriction to Γ of the functions of $Y_{i,\delta}$. The space $\Lambda_\delta^{(i)}$ is also called the Lagrange multiplier spaces, the choice of which defines the mortar element and the FETI method. For the mortar element, in case of first order approximation, $\Lambda_\delta^{(i)}$ is the subspace of all functions with zero slope on the elements including the end points [74], whereas the FETI method uses polynomial functions as the basis of $\Lambda_\delta^{(i)}$ [74].

The relation of nonconforming method to the defect correction scheme can be viewed by using the iterative formulation and notations in [95].

Denote by $\{\varphi_{k'}^{(1)}\}, k' = 1, ..., N_1$, the Lagrange functions associated with the interior nodes of Ω_1; since they vanish on Γ, they can be extended by 0 in $\overline{\Omega_2}$. These extended functions are denoted by $\{\widetilde{\varphi}^{(1)}_{k'}\}$, and can be taken as a first basis functions for V_δ. Similarly, the Lagrange functions $\{\widetilde{\varphi}^{(2)}_{k''}\}, k'' = 1, ..., N_2$ associated with the interior nodes of Ω_2 can also be generated. Finally, always supposing that Ω_1 is the master domain and Ω_2 its slave, let $\{\varphi^{(1)}_{m,\Gamma}\}$, $m = 1, ..., N^{(1)}_\Gamma$ be the Lagrange functions associated with the set of master nodes on the interface Γ. Then for every Lagrange functions $\{\varphi^{(1)}_{m,\Gamma}\}$ in $\overline{\Omega_1}$, a basis function $\{\widetilde{\varphi}_{m,\Gamma}\}$ can be obtained as follows

$$\widetilde{\varphi}_{m,\Gamma} := \begin{cases} \varphi^{(1)}_{m,\Gamma} & \text{in } \overline{\Omega_1} \\ \widetilde{\varphi}^{(2)}_{m,\Gamma} & \text{in } \overline{\Omega_2} \end{cases},$$

where

$$\widetilde{\varphi}^{(2)}_{m,\Gamma} = \sum_{j=1}^{N^{(2)}_\Gamma} \xi_j \varphi^{(2)}_{j,\Gamma},$$

$\varphi^{(2)}_{j,\Gamma}$ are the Lagrange functions in $\overline{\Omega_2}$ associated with the slave nodes of the interface Γ, and ξ_j are unknown coefficients that should be determined through the fulfilment of the matching equations (3.8). Precisely, they must satisfy

$$\int_\Gamma \left(\sum_{j=1}^{N^{(2)}_\Gamma} \xi_j \varphi^{(2)}_{j,\Gamma} - \varphi^{(1)}_{m,\Gamma} \right) \varphi^{(2)}_{l,\Gamma} = 0 \qquad \forall l = 1, ..., N^{(2)}_\Gamma.$$

A basis for V_δ is therefore provided by the set of all functions $\{\widetilde{\varphi}^{(1)}_{k'}\}, k' = 1, ..., N_1,, \{\widetilde{\varphi}^{(2)}_{k''}\}, k'' = 1, ..., N_2$ and $\{\widetilde{\varphi}_{m,\Gamma}\}$, $m = 1, ..., N^{(1)}_\Gamma$. An instance of substructuring iterative procedure for this nonconforming formulation can be written in a Dirichlet-Neumann version through the solution of the following subproblems [95]:

- Neumann step in Ω_1

$$\begin{cases} \text{find}(u^{(1)k+1}_\delta) \in Y_{1,\delta} : \\ \displaystyle\int_{\Omega_1} \nabla(u^{(1)}_\delta)^{k+1} . \nabla\varphi^{(1)}_{k'} = \int_{\Omega_1} f\varphi^{(1)}_{k'} \qquad \forall k' = 1, ..., N_1 \\ \displaystyle\int_{\Omega_1} \nabla(u^{(1)}_\delta)^{k+1} . \nabla\varphi^{(1)}_{m,\Gamma} = \int_{\Omega_1} f\varphi^{(1)}_{m,\Gamma} + \int_{\Omega_2} f\widetilde{\varphi}^{(2)}_{m,\Gamma} \\ \qquad\qquad \displaystyle - \int_{\Omega_2} \nabla(u^{(2)k+1}_\delta) . \nabla\widetilde{\varphi}^{(2)}_{m,\Gamma} \qquad \forall m = 1, ..., N^{(1)}_\Gamma \end{cases} \tag{3.9}$$

- Dirichlet step in Ω_2

$$\left\{\begin{array}{l} \text{find}(u_\delta^{(2)k+1}) \in Y_{2,\delta} : \\ \displaystyle\int_{\Omega_2} \nabla(u_\delta^{(2)})^{k+1}.\nabla\varphi_{k''}^{(2)} = \int_{\Omega_2} f\varphi_{k''}^{(2)} \qquad \forall k^{''} = 1,...,N_1 \\ \displaystyle\int_\Gamma [(u_\delta^{(2)})^{k+1} - (u_\delta^{(1)})^{k+1}]\varphi_{j,\Gamma}^{(2)} = 0 \qquad \forall j = 1,...,N_\Gamma^{(2)} \end{array}\right. \tag{3.10}$$

with a possible relaxation on the last set of interface equations.

Notice that the second equation in (3.9) can be employed as a defect function. In this framework, a defect correction scheme can be set in the following subproblems:

- Dirichlet step in Ω_1 and Ω_2

$$\left\{\begin{array}{l} \text{find}(u_\delta^{(1)})^{k+1} \in Y_{1,\delta} : \\ \displaystyle\int_{\Omega_1} \nabla(u_\delta^{(1)})^{k+1}.\nabla\varphi_{k'}^{(1)} = \int_{\Omega_1} f\varphi_{k'}^{(1)} \qquad \forall k^{'} = 1,...,N_1 \\ (u_\delta^{(1)})^{k+1} = \lambda^k \qquad \text{on } \Gamma \end{array}\right. \tag{3.11}$$

$$\left\{\begin{array}{l} \text{find}(u_\delta^{(2)})^{k+1} \in Y_{2,\delta} : \\ \displaystyle\int_{\Omega_2} \nabla(u_\delta^{(2)})^{k+1}.\nabla\varphi_{k''}^{(2)} = \int_{\Omega_2} f\varphi_{k''}^{(2)} \qquad \forall k^{''} = 1,...,N_1 \\ \displaystyle\int_\Gamma [(u_\delta^{(2)})^{k+1} - (u_\delta^{(1)})^{k+1}]\varphi_{j,\Gamma}^{(2)} = 0 \qquad \forall j = 1,...,N_\Gamma^{(2)} \end{array}\right. \tag{3.12}$$

- Setting the defect as a representation of the Neumann step at the interface Γ

$$\begin{aligned} D^{k+1} := -\int_{\Omega_1} \nabla(u_\delta^{(1)})^{k+1}.\nabla\varphi_{m,\Gamma}^{(1)} + \int_{\Omega_1} f\varphi_{m,\Gamma}^{(1)} + \int_{\Omega_2} f\widetilde{\varphi}_{m,\Gamma}^{(2)} \\ -\int_{\Omega_2} \nabla(u_\delta^{(2)k+1}).\nabla\widetilde{\varphi}_{m,\Gamma}^{(2)} \qquad \forall m = 1,...,N_\Gamma^{(1)} \end{aligned} \tag{3.13}$$

- Updating the interface values using the latest defect values

$$\lambda^{k+1} = \lambda^k + \tau(\lambda^k, D^{k+1}) \tag{3.14}$$

where τ is defined by the updating step taken in the operating nonlinear solver.

3.5 Closure

It has been shown that the adaptive-α method in the defect correction scheme is a simple preconditioner in linear algebraic systems. However, the role of a preconditioner in the defect correction scheme is only valid for discretised equations resulting from linear PDE. In general, when dealing with nonlinear problems, variants of classical nonoverlapping domain decomposition methods are employed as preconditioners of a linearised form of the whole system. In essence, the main role of defect correction methods is different where it intends to give spaces for autonomous computation of subproblems, thus also giving a freedom of employing any numerical scheme (including linearisation technique) to any subdomain computation. Consequently, the entire form of the coupling of subdomains is not necessarily a preconditioner.

Chapter 4

A Nonlinear Heat Transfer Problem

This chapter addresses the first physical problem to which the nonoverlapping domain decomposition method is applied. In the first section, a heat conduction process is illustrated in a multisubdomain electronic material where the conductivity is a function of temperature, thus making the process nonlinear. Then the numerical scheme is presented in the section 4.2 the discretisation of the governing equation and its linearisation are described. A linear solver employed to the resulting linear problem is demonstrated in the Section 4.3 and it is followed by numerical results in the Section 4.4. This example shows how a defect correction scheme is implemented as a component meshing and gluing technique, where the meshings and the computational treatment of individual components are performed independently whereas the componentwise solutions are glued or coupled via a certain defect equation.

The novel area of works associated with this chapter are:

- the computation of a nonlinear heat conduction problem in a multichip (with FORTRAN-90) by combining a Picard linearisation technique and a Preconditioned Conjugate Gradient method
- the inclusion of a second order discretisation of flux continuity across the interfaces
- the implementation of the defect correction scheme in the heat conduction computation with two different defect equations
- the integration of the defect correction iterative procedure with the nonlinear corrective procedure of the Newton-GMRES, the Newton method (with finite difference Jacobian), α-method, and two nonlinear conjugate gradient solvers.

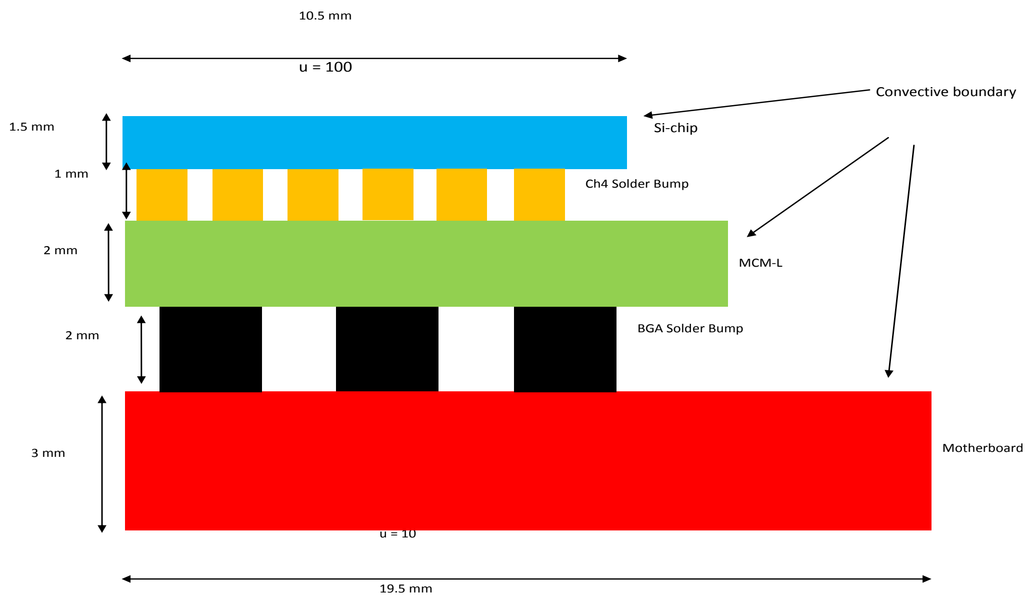

Figure 4.1: Multi-chip module of five different layers

4.1 A multi-chip electronic module

Consider a simple nonlinear steady state heat conduction process in 2D governed by

$$\nabla.\left(k(u)\nabla u\right) = \frac{\partial}{\partial x}\left(k(u)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k(u)\frac{\partial u}{\partial y}\right) = f \tag{4.1}$$

where u denotes the temperature, k is the heat conductivity and f is the source/sink within the process domain.

The process occurs in an electronic device, particularly on a multi-chip module. The module is a multi-layer of components with various temperature effect on conductivity. The physical domain has 5 layers where 2 solder joint layers (BGA and CH4 solder bump) are used to connect 3 board layers (Motherboard, MCM-L, and Si-chip) [25]. The construction of this domain is given in Fig.4.1. There are structural coupling effects due to the sandwich-like construction of the devices featuring multiple layers of specific materials. The complete description of the domain is referenced in [25].

Assume the absence of source and sink, thus $f = 0$. The boundary condition is applied with 100^0C along the bottom boundary and 10^0C along the top boundary. The left sides of the domain have symmetric boundary conditions whereas for all other boundaries, a convective heat boundary condition with ambient temperature of 25^0C and a heat transfer coefficient of 10 W/m^{2}C are assumed. The nonlinearity of the process is reflected by the heat conductivity, which takes

$k_{board}(u) = 0.005 + 0.0013u$, in board layers
$k_{solder}(u) = 280(0.005 + 0.0013u)$, in solder layers

Across the interface between two different layers where there is a jump of conductivity, the perfect contact condition [80] is considered where the the solution and the normal component of its flux are continuous. Then for any interface point (x^*, y^*), the jump conditions are:

$$[u] = u^+(x^*, y^*) - u^-(x^*, y^*) = 0$$
$$[\sigma] = \sigma^+(x^*, y^*) - \sigma^-(x^*, y^*) = 0 \quad ; \quad \sigma = -k\tfrac{\partial u}{\partial n}$$

As illustrated in Fig.4.1, each interface line between two different layers is represented by a horizontal line. Therefore the jump condition takes place across the interface line in the $y-$ direction. The meaning of the negative and positive sign in the above notations are given for any variable z by :

$$z^+(x^*, y^*) = \lim_{\delta\to 0^+} z(x^*, y^* + \delta), \quad z^-(x^*, y^*) = \lim_{\delta\to 0^-} z(x^*, y^* + \delta)$$

4.2 Numerical Schemes

Consider again the heat conduction problem

$$\nabla.\left(k(u)\nabla u\right) = \frac{\partial}{\partial x}\left(k(u)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(k(u)\frac{\partial u}{\partial y}\right) = f$$

The application of a standard central difference scheme [104] at a grid point (x_i, y_j) results in:

$$\begin{aligned}&\frac{1}{\Delta x}\left[k_{i+\frac{1}{2},j}(u)\frac{u_{i+1,j}-u_{i,j}}{\Delta x} - k_{i-\frac{1}{2},j}(u)\frac{u_{i,j}-u_{i-1,j}}{\Delta x}\right] + \\ &\frac{1}{\Delta y}\left[k_{i,j+\frac{1}{2}}(u)\frac{u_{i,j+1}-u_{i,j}}{\Delta y} - k_{i,j-\frac{1}{2}}(u)\frac{u_{i,j}-u_{i,j-1}}{\Delta x}\right] = f_{i,j}\end{aligned} \tag{4.2}$$

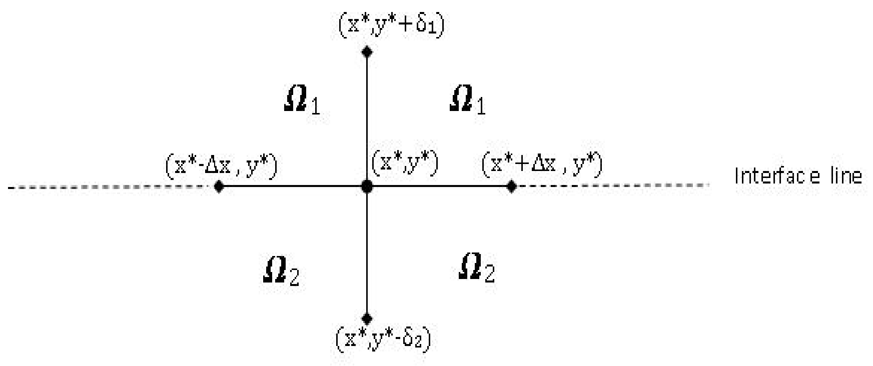

Figure 4.2: Five points stencil at the interface

where the subscript (i, j) refers to the location of a grid point at coordinate (x_i, y_j). Some approximations for the conductivity $k_{i+\frac{1}{2},j}(u)$ between two grid points (x_i, y_j) and (x_{i+1}, y_j) can be considered :

$$k_{i+\frac{1}{2},j}(u) = \frac{k(u_{i,j}) + k(u_{i+1,j})}{2} \tag{4.3}$$

$$k_{i+\frac{1}{2},j}(u) = \frac{2k(u_{i,j})k(u_{i+1,j})}{k(u_{i,j}) + k(u_{i+1,j})} \tag{4.4}$$

If the conductivity k is a continuous function of temperature u, both approximations will lead to second order accurate discretisations [123]. The first approximation is chosen in this numerical scheme for it is preferable in the case where the conductivity $k = k_0\, u^{\alpha}$ is a power function of temperature [104].

Some different notations of discrete variables needs to be addressed in this chapter. Any variable with subscripts such as $u_{i,j}$ refers to non-interface grid points. When dealing with variables at interface point $(x*, y*)$, the naming of variable is simply $u(x*, y*)$.

A special treatment is necessary to discretise the jump condition at the interface. Suppose the mesh size is uniform along the interface line (in the x-direction), whereas that in the y-direction is given by δ_1 and δ_2 as indicated in Fig-4.2. Then a second order flux continuity can be implemented following the method used in [85] by using the following flux approximations:

$$\begin{aligned}\sigma^+(x^*,y^*) = \; & a_+\Bigg[k^+(x^*,y^*)\left(\frac{u(x^*,y^*)-u(x^*,y^*+\delta_1)}{\delta_1}\right)\\ & +\frac{\delta_1}{2}\Bigg(k^+(x^*,y^*)\frac{-u(x^*+\Delta x,y^*)+2u(x^*,y^*)-u(x^*-\Delta x,y^*)}{\Delta y^2}\\ & -\frac{k^+(x^*+\Delta x,y^*)-k^+(x^*-\Delta x,y^*)}{2\Delta x}.\frac{u(x^*+\Delta x,y^*)-u(x^*-\Delta x,y^*)}{2\Delta x}\Bigg)\Bigg]\end{aligned} \tag{4.5}$$

where $a_+ = \dfrac{2k^+(x^*,y^*)}{3k^+(x^*,y^*)-k(x^*,y^*+\delta_1)}$

$$\begin{aligned}\sigma^-(x^*,y^*) = \; & a_-\Bigg[k^-(x^*,y^*)\left(\frac{u(x^*,y^*)-u(x^*,y^*-\delta_2)}{-\delta_2}\right)\\ & -\frac{\delta_2}{2}\Bigg(k^-(x^*,y^*)\frac{-u(x^*+\Delta x,y^*)+2u(x^*,y^*)-u(x^*-\Delta x,y^*)}{\Delta y^2}\\ & -\frac{k^-(x^*+\Delta x,y^*)-k^-(x^*-\Delta x,y^*)}{2\Delta x}.\frac{u(x^*+\Delta x,y^*)-u(x^*-\Delta x,y^*)}{2\Delta x}\Bigg)\Bigg]\end{aligned} \tag{4.6}$$

where $a_- = \dfrac{2k_+(x^*,y^*)}{3k^+(x^*,y^*)-k(x^*,y^*+\delta_2)}$.

It follows that the resulting equation of flux continuity $[\sigma(x^*,y^*)] = \sigma^+(x^*,y^*)-\sigma^+(x^*,y^*) = 0$ will still use the standard five-points stencil while retaining the second order accuracy.

The discretisation over all grid points will yield a system of equations which can be seen in a matrix form :

$$A(u)\; u = f \tag{4.7}$$

The evaluations of the thermal conductivity in (4.2) build the the components of the coefficient matrix $A(u)$, as a result the nonzero components (up to five nonzero components in each row) in the matrix will have the same pattern as in the linear problem, but the values of the components will change with the temperature u. Therefore with this approximation, the scheme (4.2) is nonlinear with respect to u.

4.2.1 Linearisation Method

In order to deal with the nonlinear algebraic form (4.7), a Picard linearisation method [104] is employed in an iterative fashion by freezing the coefficients of the equations, i.e. the coefficients are fixed and computed from the values obtained in previous iteration level.

Denoting u and $\widehat{u}$ as the iterate of current level $u^{(n)}$ and previous level $u^{(n-1)}$ respectively, the scheme (4.2) can now be written in a linear form

$$\begin{aligned}\frac{1}{\Delta x}\left[k_{i+\frac{1}{2},j}(\widehat{u})\frac{u_{i+1,j}-u_{i,j}}{\Delta x}-k_{i-\frac{1}{2},j}(\widehat{u})\frac{u_{i,j}-u_{i-1,j}}{\Delta x}\right]+\\ \frac{1}{\Delta y}\left[k_{i,j+\frac{1}{2}}(\widehat{u})\frac{u_{i,j+1}-u_{i,j}}{\Delta y}-k_{i,j-\frac{1}{2}}(\widehat{u})\frac{u_{i,j}-u_{i,j-1}}{\Delta x}\right]=f_{i,j}\end{aligned} \tag{4.8}$$

where the conductivity function takes the approximation (4.3). The linear scheme (4.8) can be solved by a linear solver at the current iteration level n. The iteration is stopped once the difference between the current iterate u and the previous iterate $\widehat{u}$ achieves a specific tolerance. In view of the form (4.7), the iterative procedure is perfomed by simply make an initial guess, compute the thermal conductivity at each point in space, evaluate the matrix $A(u)$ and solve for the next possible temperature [125].

```
**Picard's Algorithm.**
choose initial   $u^0$
**Do** $m = 0, \mathrm{maxm}$
    compute $A(u^m)$
    solve $A(u^m)u^{m+1} = f$
    test for convergence : $\|u^{m+1} - u^m\|$
**End**
```

In the defect correction scheme, Dirichlet conditions are imposed at the interface boundaries. It can be verified that, the matrix $A(u)$ of any subdomain problem is symmetric positive definite. The symmetricity of the subdomain matrix built by (4.8) simply follows from the evaluation of the conductivity (4.3). Furthermore it can be inspected, through the definition of strongly connected graph and a theorem in [118], that $A(u)$ is also irreducible. The positive definiteness then follows from the fact that an irreducibly diagonal dominant matrix is positive definite [118]. With these two properties of $A(u)$, subdomain computations of the linearized equations can be performed in each iteration with the following linear solver.

4.2.2 Conjugate Gradient Solver

Consider the linear system

$$Ax = b \tag{4.9}$$

which may be built by the discretization (4.8) with x as the temperature variable representing the discrete values of u whereas A is the matrix containing the relevant conductivity values.

One of the most popular iterative methods to solve this linear system is the conjugate gradient method. It is designed for linear problem with symmetric positive definite matrix A to minimize the A norm of the residual vector over a Krylov subspace [102].

The method of Conjugate Gradient is [106] :

$$
\begin{aligned}
d_{(0)} = r_{(0)} &= b - Ax_{(0)}, \\
\alpha_{(i)} &= \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}, \\
x_{(i+1)} &= x_{(i)} + \alpha_{(i)} d_{(i)}, \\
r_{(i+1)} &= r_{(i)} - \alpha_{(i)} A d_{(i)}, \\
\beta_{(i+1)} &= \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}, \\
d_{(i+1)} &= r_{(i+1)} + \beta_{(i+1)} d_{(i)},
\end{aligned}
$$

4.2.3 Preconditioned Conjugate Gradient Method

Preconditioning is a technique for improving the condition number of a matrix. In the linear problem (4.9), A is the matrix of interest which needs to be preconditioned. Suppose that M is a symmetric positive-definite matrix that approximates A but is easier to invert, then (4.9) can be solved indirectly by solving

$$M^{-1}Ax = M^{-1}b \tag{4.10}$$

If $\kappa(M^{-1}A) << \kappa(A)$, or if the eigenvalues of $M^{-1}A$ are better clustered than those of A, the equation (4.10) can be solved iteratively more quickly than the original problem (4.9). It requires that the preconditioner M be symmetric positive definite. Skipping the mathematical details of the matrix transformation like shown in [106], the method of Preconditioned Conjugate Gradient can be derived in the following :

$$
\begin{aligned}
r_{(0)} &= b - Ax_{(0)}, \\
d_{(0)} &= M^{-1} r_{(0)}, \\
\alpha_{(i)} &= \frac{r_{(i)}^T M^{-1} r_{(i)}}{d_{(i)}^T A d_{(i)}}
\end{aligned}
$$

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$$

$$r_{(i+1)} = r_{(i)} - \alpha_{(i)} A d_{(i)}$$

$$\beta_{(i+1)} = \frac{r_{(i+1)}^T M^{-1} r_{(i+1)}}{r_{(i)}^T M^{-1} r_{(i)}}$$

$$d_{(i+1)} = M^{-1} r_{(i+1)} + \beta_{(i+1)} d_{(i)}$$

4.2.4 SSOR Preconditioner

A simple but effective preconditioning can be obtained from the Gauss-Seidel method, despite its slow convergence when it is used as stand-alone iteration [14]. If the original, symmetric, matrix is decomposed as

$$A = D + L + L^T$$

in its diagonal, lower, and upper triangular part, the SSOR matrix is defined as

$$M = (D + L)D^{-1}(D + L)^T,$$

or parametrized [8] by ω

$$M(\omega) = \frac{1}{2-\omega}(\frac{1}{\omega}D + L)((\frac{1}{\omega}D)^{-1}((\frac{1}{\omega}D + L)^T.$$

4.3 Numerical Results

This section demonstrates the results obtained from applying the above methods to the nonlinear heat conduction equation in the multichip. The Picard algorithm is used as the linearisation scheme. The combination of this linearisation and the LU Decomposition solver is used to obtain the reference solution of the nonlinear heat conduction in the multichip by running them under the conventional scheme with simultaneous/sequential computation. Some computations are carried out on uniform meshes of square grid ($\Delta x = \Delta y = h$), by varying the mesh size ($h = 0.5$ mm, 0.25 mm, 0.125 mm, and 0.0625 mm). The solution on the finest mesh ($h =$0.0625 mm) is taken as the reference solution u_{ref}. Let u_h be the solution obtained with mesh size h, the Table-4.1 dispays the error of each solution with respect to u_{ref} in the ∞-norm and a normalized Euclidean norm. In both norms, the errors decrease with the mesh refinement.

The temperature contour of the reference solution is depicted in Fig.4.3. The temperature

Table 4.1: Error with respect to the reference solution

h(mm)	$\|\|u_{ref} - u_h\|\|_\infty$	$\dfrac{\|\|u_{ref} - u_h\|\|}{\|\|u_{ref}\|\|}$
0.500	4.6490	0.0139
0.250	1.9470	0.0058
0.125	0.9370	0.0037

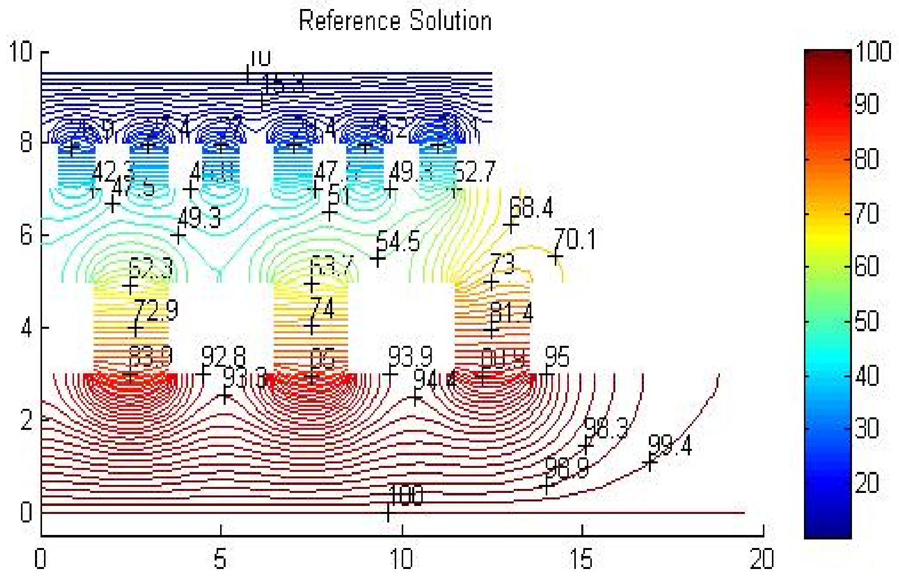

Figure 4.3: Temperature Contour in the Multi-Chip

decreases from the bottom (100^0 C) to the top (10^0 C) of the domain.

The next simulations demonstrate the implementation of the defect correction scheme in the multichip module. The domain is decomposed into 12 subdomains based on the nature of their physical structure, i.e. 3 board subdomains and 9 solder bump subdomains. Accordingly, there are 4 interface lines splitting the subdomains.

In every subdomain, the nonlinear heat conduction equation(4.1) is solved in every nonlinear iteration with Dirichlet interface conditions given by the interface temperature λ obtained from the update process of any nonlinear solver. The pre-processing stage is again performed by employing a uniform square grid through the entire mesh with (h = 0.5 mm, h = 0.25 mm, h= 0.125 mm, and h= 0.0625 mm).

There are two significant factors in the scheme, i.e.the choice of defect equation and the nonlinear solver. The performances of the scheme with the variation of each factor are assessed below.

4.3.1 Comparison of Defect Functions

In this section, the accuracy of two defect functions are compared. The error in the Euclidean norm is calculated with respect to the reference solution, i.e. $\|u - u_{ref}\|$ where the reference solution is the solution obtained from conventional scheme with the finest grid.
The first defect function is denoted by the second order form of (2.3) :

$$D(x^*, y^*) = \partial^+_{n,2} u(x^*, y^*) - \partial^-_{n,2} u(x^*, y^*) = 0 \tag{4.11}$$

for each interface point $(x^*, y^*) \in \Gamma$, where both the forward and backward difference scheme use three grid points:

$$\partial^+_{n,2} u(x^*, y^*) = \frac{-3u(x^*, y^*) + 4u(x^*, y^* + \Delta y) - u(x^*, y^* + 2\Delta y)}{2\Delta y}$$

$$\partial^-_{n,2} u(x^*, y^*) = \frac{3u(x^*, y^*) - 4u(x^*, y^* - \Delta y) + u(x^*, y^* - 2\Delta y)}{2\Delta y}$$

While the first order form of (2.3) is employed in [78], [25], and [6], the defect equation (4.11) has recently been used in [57].

The second defect equation is a finite difference form of (2.5) at each interface point (x^*, y^*):

$$\begin{aligned} D(x^*, y^*) = \; & \frac{1}{\Delta x}\left[\bar{k}(x^* + \Delta x/2, y^*)\frac{u(x^* + \Delta x, y^*) - u(x^*, y^*)}{\Delta x}\right. \\ & \left. -\bar{k}(x^* - \Delta x/2, y^*)\frac{u(x^*, y^*) - u(x^* - \Delta x, y^*)}{\Delta x}\right] + \\ & \frac{1}{\Delta y}\left[k(x^*, y^* + \Delta y/2)\frac{u(x^*, y^* + \Delta y) - u(x^*, y^*)}{\Delta y}\right. \\ & \left. -k(x^*, y^* - \Delta y/2)\frac{u(x^*, y^*) - u(x^*, y^* - \Delta y)}{\Delta y}\right] = 0 \end{aligned} \tag{4.12}$$

The conductivity $\bar{k}$ must be defined more specifically, since it lies at the interface where there is a jump of conductivity. The approximation is taken following a simple averaging scheme in [62] for heat flow along interface lines :

$$\bar{k}(x^* + \Delta x/2, y^*) = \frac{k^+(x^* + \Delta x/2, y^*) + k^-(x^* + \Delta x/2, y^*)}{2}$$

Some numerical computations of defect correction scheme under the two defect equations are run using a sequence of mesh with the size $h = 0.5$ mm, $h = 0.25$mm, $h = 0.125$mm, and $h = 0.0625$mm. Table 4.2 displays the error of both schemes with respect to the reference solution u_{ref}. In the table, $u_{h,1}$ and $u_{h,2}$ denote the solution obtained from the defect correction scheme with the defect (4.11) and (4.12) respectively. The error of the scheme with (4.12) decreases when the mesh size is refined. With the finest mesh, its maximum error is also very small compared to the range of the temperature in the domain. On the other hand, the scheme with (4.11) shows the opposite. The error increases with the mesh refinement and furthermore, the error is very large for each grid size.

Another comparison is given between the solution of the schemes and the conventional solution u_h for each grid size h. Recall that the error of the conventional solution has been given in the previous section. Table4.3 indicates a minor difference between the conventional solution u_h and the solution $u_{h,2}$ given by the scheme with (4.12), whereas the scheme with (4.11) yields a solution $u_{h,2}$ which has a large discrepancy from u_h.

Table 4.2: Comparison of Defect Equation

h(mm)	$\|\|u_{ref} - u_{h,1}\|\|_\infty$	$\frac{\|\|u_{ref}-u_{h,1}\|\|}{\|\|u_{ref}\|\|}$	$\|\|u_{ref} - u_{h,2}\|\|_\infty$	$\frac{\|\|u_{ref}-u_{h,2}\|\|}{\|\|u_{ref}\|\|}$
0.5000	12.702	6.47e-2	4.6120	1.4e-2
0.2500	13.821	6.47e-2	1.7970	5.9e-3
0.1250	14.340	6.67e-2	0.9360	3.9e-3
0.0625	14.790	6.76e-2	0.0860	6.49e-4

Table 4.3: Comparison of Defect Equation

h(mm)	$\|\|u_h - u_{h,1}\|\|_\infty$	$\frac{\|\|u_h-u_{h,1}\|\|}{\|\|u_{ref}\|\|}$	$\|\|u_h - u_{h,2}\|\|_\infty$	$\frac{\|\|u_h-u_{h,2}\|\|}{\|\|u_{ref}\|\|}$
0.5000	12.9010	6.13e-2	0.1540	6.21e-4
0.2500	13.7790	6.28e-2	0.2620	1.62e-3
0.1250	14.5870	6.53e-2	0.0860	3.63e-4
0.0625	14.7900	6.76e-2	0.0860	6.49e-4

The inaccuracy of the scheme with (4.11) is actually not surprising. The discretisation of (2.3), at its best, only satisfy the continuity of normal derivative. In the interface problem of heat conduction process, the physics requires the continuity of flux [80] instead of the continuity of normal derivative. If the flux continuity is chosen as the defect equation by using the approximation (4.5) and (4.6), the solution of the scheme will certainly match the reference solution, because the latter is obtained from the conventional scheme which also uses (4.5) and (4.6). Nevertheless, what the results in this section have shown is that the defect (4.12) is a better option than (4.11).

4.3.2 Comparison of Nonlinear Solvers

This section demonstrates the performance of some nonlinear solvers in the defect correction scheme. The algorithm of the nonlinear solvers have been described in Chapter-2. The performance parameters are the number of nonlinear iterations, the number of domain decomposition iterations and the total elapsing time that the scheme needs to reach a certain residual tolerance. The defect equation (4.12) is chosen for this test.

The same mesh configurations are again used here, i.e. $h = 0.5$ mm, $h = 0.25$mm, $h = 0.125$mm, and $h = 0.0625$mm. The stopping criterion for the computations is a relative residual of 5e-3, where the relative residual is defined as the ratio between the norm of current defect and that of the early defect. Each computation is run with initial guess 20^0C in the entire grid points. For each subdomain computation, the convergence criterion for the Picard algorithm is when the maximum difference between two successive iterates falls below 5e-4. For each Picard iteration, the PCG solution of the resulting linear equation is obtained when the residual norm is reduced to one percent of the early residual norm.

The Newton method with finite difference Jacobian (Newton-FD) is applied in the scheme by using a perturbation of $\delta u = 1^0$ C. This is the optimal value obtained after a series of trials with different perturbation values for this method, and it is also better than the perturbation which uses the scaling (2.22). A scaling parameter of $\sigma = 0.1$ is used for the Newton-GMRES method since this gives less nonlinear iteration than the scaling described in section 2.4.3 after a number of numerical tests. For the line search, both Newton FD and Newton-GMRES use the Armijo's rule. While most computational procedures for this work is written in FORTRAN-90 from scratch, the LU decomposition solver for the Newton-FD and the least square minimizer for the Newton-GMRES use some LAPACK routines [135]. In the implementation of the last two nonlinear solvers, The Nonlinear Conjugate Gradient-Fletcher Reeves (NCG-FR) and The Nonlinear Conjugate Gradient-Pollack Ribiere (NCG-PR) use the parameter $\overline{\nu} = 1e - 5$.

Table 4.4 demonstrates the performance of the above solvers for each mesh size. Apart from the α-method, the number of domain decomposition iterations (DD iteration) is not the same as the number of nonlinear iteration (NL iteration). Both conjugate gradient solvers require two DD iterations to yield a new direction. As described in section 2.4.2, the number of DD iterations to obtain one Newton step in the Newton-FD method is proportional to the number of interface grid points. In the Newton-GMRES method, the number of DD iterations for one Newton step depends on the residual $\widetilde{\rho}_k$ (Step-2 of Inexact-GMRES Algorithm in section 2.4.3). In Table 4.4, the α method, Newton-FD and Newton-GMRES show the same trend where the number of NL iteration, and hence the DD iterations, increases when the mesh is refined. Both conjugate gradient solvers do not show the same indication. There is no specific pattern in the iteration numbers with the mesh refinement. Furthermore, both of them, and particularly NCG-FR, give more satisfying results in terms of the number of outer iteration.

Although the NCG-FR method gives the best result in each performance criterion, the Newton-GMRES method needs some notices. Despite its higher number of DD iterations, it can compete with the nonlinear conjugate gradient solver in terms of the computing time when the mesh is refined. An easy comparison can be made between the Newton-GMRES and the α-method. For each mesh size, the α-method needs less DD iterations, but the computing time of the Newton-GMRES method is much faster. The only reasonable explanation for this is that the average number of inner iterations (Picard iterations and PCG iterations) that the Newton-GMRES needs for one DD iteration is smaller.

4.4 Closure

This chapter has illustrated the implementation of the defect correction scheme in a nonlinear heat conduction process in a multichip module. The Picard linearisation technique and PCG linear solver are used to in each subdomain computation. The scheme with the defect equation (4.12) gives a much better solution than the scheme with (4.11). In general, the number of outer iteration increases with the mesh refinement, but it fluctuates in the case of NCG-FR and NCG-PR. Overall, both nonlinear conjugate gradient solvers give more satisfying performance in terms of the number of outer iterations and computing time.

Table 4.4: Performance of Nonlinear Solvers

Method	h(mm)	DD Iteration	NL Iteration	CPU Time(hours)
α Method	0.50	1622	1622	00:00:20
	0.25	4316	4316	00:04:00
	0.125	10164	10164	01:07:58
	0.0625	24871	24871	95:05:36
Newton FD	0.5	294	4	00:00:04
	0.25	844	7	00:00:40
	0.125	5879	27	00:49:09
	0.0625	> 75000	—	not converge yet
Newton GMRES	0.5	6245	435	00:00:40
	0.25	9382	728	00:02:58
	0.125	11659	783	00:38:48
	0.0625	26652	1563	11:10:19
NCG FR	0.5	945	472	00:00:13
	0.25	978	489	00:01:33
	0.125	1495	747	00:18:50
	0.0625	1269	634	09:49:09
NCG PR	0.5	1023	512	00:00:12
	0.25	2469	1235	00:02:32
	0.125	1844	922	00:15:28
	0.0625	2790	1395	11:37:21

Chapter 5

Some Fluid Flow and Heat Transfer Problems Using the PHOENICS CFD Code

This chapter discusses some applications of fluid flow and heat transfers in the PHOENICS CFD code. Its implementation with defect correction scheme will be demonstrated in Chapter-6. The first section of this chapter shows the solution procedure performed in PHOENICS. Two problems will be given in the context of the Darcy's Law and they are solved using PHOENICS. The first case is a 2D single phase fluid flow and heat transfer whereas the second case is a 1D multiphase fluid flow with heat & moisture transfer.

The novel work related to this application is the inclusion of highly coupled system of differential equations of multiphase problem into the PHOENICS code where various terms in the PHOENICS code need to be adjusted in order to comply with the form of the desired governing equations.

5.1 Computational Procedures in PHOENICS

PHOENICS is a computational tool which simulates processes involving fluid flow, heat or mass transfer, chemical reaction and combustion in nearly every branch of engineering and science in which fluid flow plays a key role. The basic fluid flow problem which PHOENICS solves is governed by the coupled system of continuity and momentum equation [109] [108]:

Continuity equation :

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{u}) = 0 \tag{5.1}$$

Momentum equation :

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla.(\rho \mathbf{u}\mathbf{u}) = -\nabla p + \nabla.(\mu \nabla)\mathbf{u} + \mathbf{F} \tag{5.2}$$

In those equations, ρ denotes the density of the fluid, the velocity field $\mathbf{u}$ is a vector variable with its component in each direction, μ is the dynamic fluid viscosity, and $\mathbf{F}$ represents all other forces applied to the fluid . In addition to the coupled equations (5.1) and (5.2), PHOENICS independently solves other transport variables such as the energy equation :

$$\frac{\partial \rho H}{\partial t} + \nabla.(\rho \mathbf{u} H) = \nabla.(k \nabla T) \tag{5.3}$$

where H is the enthalpy, T is the temperature, k is the thermal conductivity and S is the rate of heat generation. The similarity of form in (5.1)-(5.3) enables the division of terms in the differential equations into the transient, convection, diffusion, and source terms. If the dependent variable is denoted by ϕ, the differential equations which PHOENICS solves can be generalized in the following form:

$$\underbrace{\frac{\partial \rho \phi}{\partial t}}_{\text{transient}} + \underbrace{\nabla.(\rho \mathbf{u} \phi)}_{\text{convection}} = \underbrace{\nabla.(\Gamma \nabla \phi)}_{\text{diffusion}} + \underbrace{S}_{\text{source}} \tag{5.4}$$

where Γ is the diffusion coefficient, and S is the source term. The quantities Γ and S are specific to a particular meaning of ϕ. It is obvious that for the continuity equation ($\phi = 1$), both Γ and S reduce to zero, while for the momentum equation ($\phi = \mathbf{u}$) the associated terms are $\Gamma = \mu$ and $S = -\nabla p + \mathbf{F}$. Other transport equations (concentration, turbulence quantity, etc) are also solved with relevant substitution of those terms.

5.1.1 Finite Volume Discretisation

PHOENICS uses the Finite Volume Method (FVM) to solve the transport equations. To use the FVM, the solution domain must first be divided into non-overlapping polyhedral Control Volumes (CV's). The values of the independent and dependent variables located at the centres of these CV's are assumed to be the average value across the whole CV. The first step in the discretisation of (5.4) using the FVM is to integrate over a control volume and then make appropriate approximations for fluxes across the boundary of each CV. In the following, the volume integration of each term in (5.4) will be examined separately.

- The transient term,

$$\int_V \frac{\partial \rho\phi}{\partial t} dV \approx \frac{\rho_P \phi_P V_P - \rho_P^0 \phi_P^0 V_P^0}{\Delta t}$$

where the subscript P and superscripts 0 refers to the current and variable value at the CV respectively. The volume of the CV is denoted by V_P and the time step size by Δt.

- The convection term

$$\begin{aligned}\int_V \nabla.(\rho\mathbf{u}\phi) dV &= \textstyle\int_S \nabla.(\mathbf{u}.\mathbf{n})\rho\phi dS \\ &\approx \textstyle\sum_f \rho_f (\mathbf{u}.\mathbf{n})_f \phi_f A_f\end{aligned}$$

where $\sum_f$ denotes the summation over the faces of a CV and A_f is the area of each face. The face density, ρ_f, is calculated using the upwinding technique,

$$\begin{aligned}\rho_f = \rho_P \quad &\text{if} \quad (\mathbf{u}.\mathbf{n})_f \geq 0 \\ \rho_f = \rho_A \quad &\text{if} \quad (\mathbf{u}.\mathbf{n})_f < 0\end{aligned}$$

where the subscript A denotes the adjacent element. The face velocity $(\mathbf{u}.\mathbf{n})_f$ can be calculated directly when using a staggered grid which will be described later in this chapter. The value of ϕ_f depends on ϕ_A and ϕ_P such that

$$\phi_f = \alpha_f \phi_P + (1 - \alpha_f)\phi_A$$

The value α_f is determined by the scheme used. When upwinding is used, it has the value of

$$\begin{aligned}\alpha_f = 1 \quad &\text{if} \quad (\mathbf{u}.\mathbf{n})_f > 0 \\ \alpha_f = 0 \quad &\text{if} \quad (\mathbf{u}.\mathbf{n})_f < 0\end{aligned}$$

- The diffusion term

$$\begin{aligned}\int_V \nabla.(\Gamma\nabla\phi) dV &= \textstyle\int_S \Gamma(\nabla\phi.\mathbf{n}) dS \\ &\approx \Gamma_f (\nabla\phi.\mathbf{n})_f A_f\end{aligned}$$

The value of the diffusion coefficient Γ_f is estimated using the harmonic mean

$$\Gamma_f = \frac{\Gamma_A \Gamma_P}{\beta_f \Gamma_A + (1-\beta_f)\Gamma_P}$$

with $\beta_f = \frac{d_{Pf}}{d_{AP}}$ where d_{AP} is the distance between neighbouring cell centres and d_{Pf} is the distance from the cell centre to the face. When the mesh is orthogonal, the estimation for the flux $(\nabla\phi.\mathbf{n})_f$ is calculated by

$$(\nabla\phi.\mathbf{n})_f \approx \frac{\phi_A - \phi_P}{d_{AP}}.$$

- The source term

$$\int_V S\, dV \approx V_P(S_C - S_P\phi_P)$$

  If the source term is a function of dependent variable ϕ itself, the approximation is accounted for in a linear dependence because the discretisation equations will be solved by the techniques for linear algebraic equations.

Combining all the terms above, the complete discretisation for the equation (5.4) can be written for each CV in:

$$\begin{aligned}\frac{\rho_P\phi_P V_P - \rho_P^0\phi_P^0 V_P^0}{\Delta t} &+ \textstyle\sum_f \rho_f(\mathbf{u}.\mathbf{n})_f A_f(\alpha_f\phi_P + (1-\alpha_f)\phi_A) \\ &= \Gamma_f A_f \tfrac{(\phi_A - \phi_P)}{d_{AP}} + V_P(S_C - S_P\phi_P) \qquad (5.5)\end{aligned}$$

Equation (5.5) is now in the form of a linear combination of the CV value and its neighbours' values. If the discretisation is applied to the continuity equation in the CV illustrated in Fig.5.1, the resulting equation can be written with all flux components:

$$\frac{\rho_P\phi_P V_P - \rho_P^0\phi_P^0 V_P^0}{\Delta t} + F_e - F_w + F_n - F_s = 0 \qquad (5.6)$$

where

$$F_e = (\rho u)_e \Delta y$$

is the mass flow rate through the face e. Similarly for other faces,

$$\begin{aligned} F_w &= (\rho u)_w \Delta y, \\ F_n &= (\rho v)_n \Delta x, \\ F_s &= (\rho v)_s \Delta x. \end{aligned}$$

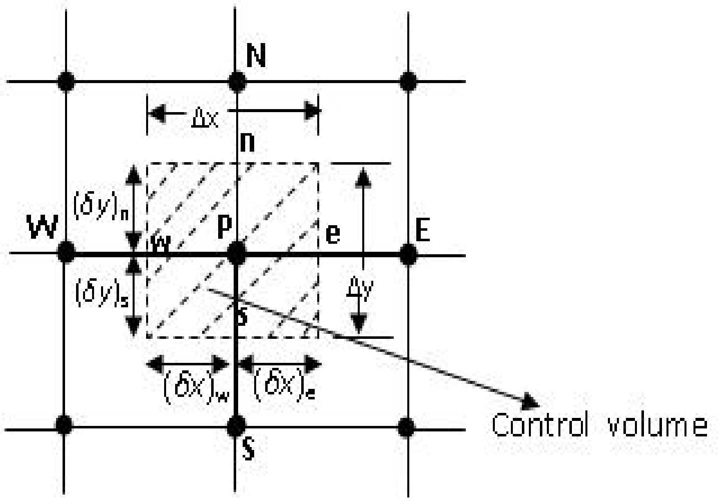

Figure 5.1: Control volume in two-dimensions

In above definitions, u and v denote the velocity components in the x and y directions. With the help of (5.6), Eq.(5.5) can be written for the CV in Fig.5.1 in the following form:

$$\begin{aligned}(\phi_P - \phi_P^0)\frac{\rho_P^0 \Delta x \Delta y}{\Delta t} &+ (D_e + (\alpha_e - 1)F_e)(\phi_P - \phi_E) + (D_w + \alpha_w F_w)(\phi_P - \phi_W) \\ &+ (D_n + (\alpha_n - 1)F_n)(\phi_P - \phi_N) + (D_s + \alpha_s F_s)(\phi_P - \phi_S) \\ &= (S_C + S_P \phi_P)\Delta x \Delta y. \qquad (5.7)\end{aligned}$$

with

$$D_e = \frac{\Gamma_e \Delta y}{(\delta x)_e}, \quad D_w = \frac{\Gamma_w \Delta y}{(\delta x)_w}, \quad D_n = \frac{\Gamma_n \Delta x}{(\delta y)_n}, \quad D_s = \frac{\Gamma_s \Delta x}{(\delta y)_s}$$

If upwinding technique is used as the interpolation method, the equation (5.7) can be cast in a more simple form:

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + a_N \phi_N + a_S \phi_S + b, \qquad (5.8)$$

where

$$\begin{aligned}
a_E &= D_e + max(-F_e, 0), \\
a_W &= D_w + max(F_w, 0), \\
a_N &= D_e + max(-F_n, 0), \\
a_S &= D_s + max(F_s, 0), \\
a_P^0 &= \frac{\rho_P^0 \Delta x \Delta y}{\Delta t}, \\
b &= S_C \Delta x \Delta y + a_P^0 \phi_P^0, \\
a_P &= a_E + a_W + a_N + a_S + a_P^0 + S_P \Delta x \Delta y.
\end{aligned}$$

In general, it is convenient to think of (5.8) as having the form

$$a_P \phi_P = \sum a_{nb} \phi_{nb} + b$$

where the subscript nb denotes a neighbour, and the summation is ot be taken over all the neighbours.

5.1.2 Discretisation of the Momentum Equations

PHOENICS uses a displaced or "staggered" grid for velocity components. In this grid, the velocity components are calculated for the points that lie on the faces of the contol volumes. The x-direction velocity u is calculated at the faces that are normal to the x direction. Similar approaches are also taken for the velocity components v and w with respect to the corresponding faces. Thus the placement of variables will look like Fig.5.2 where scalar variables are stored at cell centres while velocity components are stored at corresponding face centres.

Based on the variable placements in Fig.5.2, a staggered control volume for the x-momentum equation is illustrated in Fig.5.3. The locations for u are shown by short arrows, while the main grid points are shown by dark circles. It can be noticed that, with respect to the main grid points, the u locations are staggered in the x direction. Therefore the u-control volume indicated by the dashed box is centred at point e and its faces normal to the x-direction pass through the main grid points P and E.

For clarity purposes, the descriptions given from here until section 5.1.4 are taken from the reference [91]. The calculation of the diffusion coefficient and the mass flow rate at

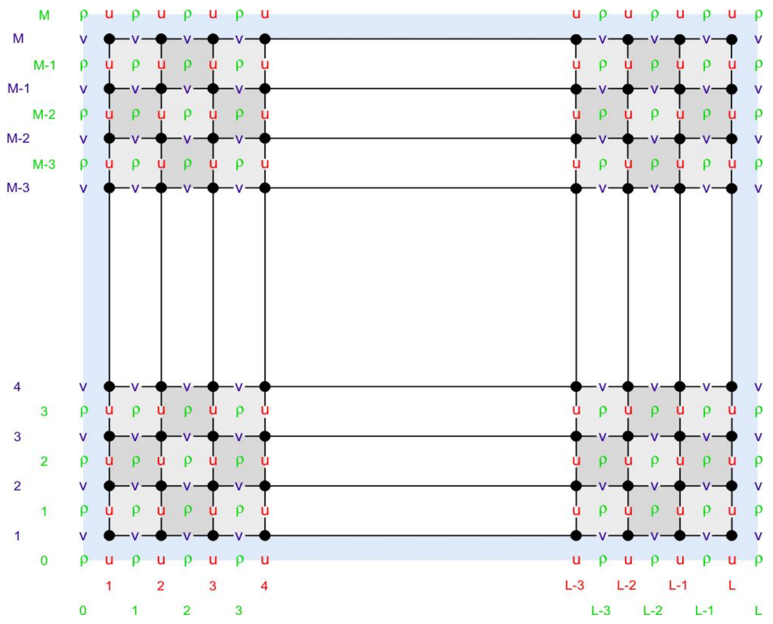

Figure 5.2: Placement of variables in the PHOENICS grid setting

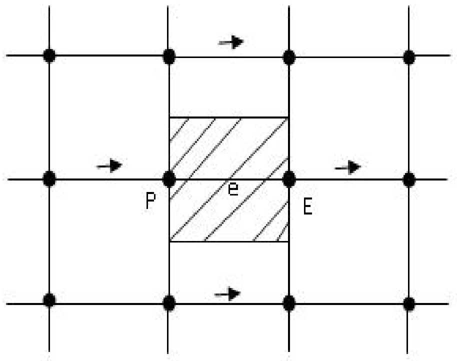

Figure 5.3: Control volume for u

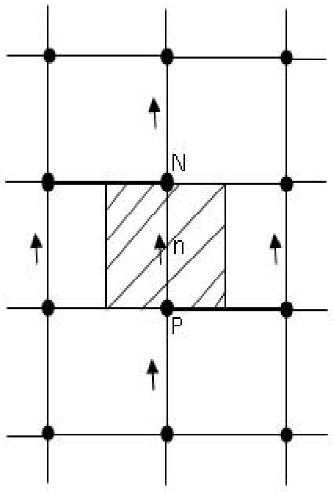

Figure 5.4: Control volume for v

the faces of the u-control volume in Fig.5.3 would require an appropriate interpolation, but essentially the same formulation as described for the scalar transport variables in the previous section would be applicable. The finite volume discretisation equation can be written as

$$a_e u_e = \sum a_{nb} u_{nb} + b + (p_P - p_E) A_e. \tag{5.9}$$

Here the number of neighbour terms will depend on the dimensionality of the problem. For the two-dimensional situation in Fig.5.3, four u neighbours are shown outside the control volume; for a three-dimensional case, six neighbour u's would be included. The neighbour coefficients a_{nb} account for the combined convection-diffusion influence at the control-volume faces. The term b is defined in the same manner as in (5.8), but the pressure gradient is not included in the source term quantities S_C and S_P. The pressure gradient gives rise to the last term in (5.9). The term $(p_P - p_E) A_e$ is the pressure force acting on the u control volume, A_e being the area on which the pressure difference acts. For two dimensions, A_e will be $\Delta y \times 1$, while in the three-dimensional case A_e will stand for $\Delta y \Delta z$. Similar approach can be taken for the momentum equations in the other directions. Figure.5.4 shows the control volume for the y-direction momentum equation; it is staggered in the y direction. The discretisation equation for v_n can be seen to be

$$a_n v_n = \sum a_{nb} v_{nb} + b + (p_P - p_N) A_n, \tag{5.10}$$

where $(p_P - p_N) A_n$ is the appropriate pressure force. For the three-dimensional case, a similar equation for the velocity component w can be written. The momentum equations can be solved only when the pressure field is given or is somehow estimated. Unless the correct pressure field is employed, the resulting velocity field will not satisfy the continuity equation. Such an imperfect velocity field based on a guessed pressure field p^* will be denoted by u^*, v^*, w^*. This "starred" velocity field will result from the solution of the following discretisation equations:

$$a_e u_e^* = \sum a_{nb} u_{nb}^* + b + (p_P^* - p_E^*) A_e, \tag{5.11}$$

$$a_n v_n^* = \sum a_{nb} v_{nb}^* + b + (p_P^* - p_N^*) A_n, \tag{5.12}$$

$$a_t w_t^* = \sum a_{nb} w_{nb}^* + b + (p_P^* - p_T^*) A_t. \tag{5.13}$$

In these equations, the velocity components and pressure have been given the superscript *. The grid point T is the immediate grid point on the top of the grid point P in the z-direction. Hence, the location t lies on the z-direction grid line between the grid points P and T. The aim is now to find a way of improving the guessed pressure p^* such that the redulting starred velocity field will progressively get closer to satisfying the continuity equation. Let the correct pressure p be

$$p = p^* + p', \tag{5.14}$$

where p' will be be called the pressure correction. Next, the responses of velocity components to this change in pressure needs to be known. The corresponding velocity corrections u', v', w' can be introduced in sinilar manner:

$$u = u^* + u' \quad v = v^* + v' \quad w = w^* + w'.$$

Substacting (5.11) from (5.9) yields

$$a_e u_e' = \sum a_{nb} u_{nb}' + (p_P' - p_E') A_e.$$

An approximation is taken by wiping out the term $\sum a_{nb} u_{nb}'$ from the equation. The result is

$$a_e u_e' = (p_P' - p_E') A_e$$

or

$$u_e' = d_e (p_P' - p_E') \tag{5.15}$$

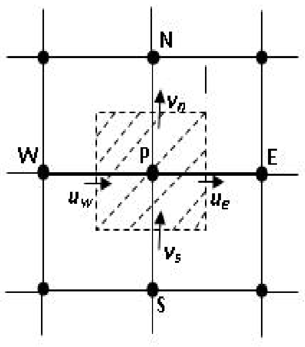

Figure 5.5: Control volume for the continuity equation

where

$$d_e = \frac{A_e}{a_e}.$$

Equation (5.15) will be called the velocity-correction formula, which can also be written as

$$u_e = u_e^* + d_e(p'_P - p'_E). \tag{5.16}$$

This shows how the starred velocity u_e* is to be corrected in response to the pressure corrections to produce u_e. The correction formuilas for the velocity components in the other directions can be written similarly:

$$v_n = v_n^* + d_n(p'_P - p'_N). \tag{5.17}$$

$$w_t = w_t^* + d_t(p'_P - p'_T). \tag{5.18}$$

5.1.3 Pressure Correction Equation

This section describes the way to turn the continuity equation into an equation for the pressure correction. For the purpose of derivation, it is assumed that the density ρ does not directly depend on pressure. The derivation is given here for the three-dimensional situation; the one- and two-dimensional forms can be easily obtained. Recall the continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0. \tag{5.19}$$

This equation will be integrated over the shaded control volume shown in Fig.5.5 where only a two-dimensional view is shown for convenience. For the integration of the term $\frac{\partial \rho}{\partial t}$, it is assumed that the density ρ_P prevails over the control volume. Also, a velocity component such as u_e located on a control-volume face will be supposed to govern the mass flow rate for the whole face. In conformity with the fully implicit practice, the new values of velocity and density (i.e. those at time $t + \Delta t$) will be assumed to prevail over the time step; the old density ρ_P^0 (i.e. the one at time t) will appear only through the term $\frac{\partial \rho}{\partial t}$. The integrated form of Eq.(5.19) becomes

$$\frac{(\rho_P - \rho_P^0)\Delta x\,\Delta y\,\Delta z}{\Delta t} + [(\rho u)_e - (\rho u)_w]\Delta y\,\Delta z + [(\rho v)_n - (\rho v)_s]\Delta z\,\Delta x$$
$$+[(\rho w)_t - (\rho w)_b]\Delta x\,\Delta y = 0. \qquad (5.20)$$

Substituting for all the velocity components the expressions given by the velocity correction formulas [such as Eqs. (5.16)-(5.18)], the following discretisation equation for p' is obtained after rearrangement:

$$a_P p'_P = a_E p'_E + a_W p'_W + a_N p'_N + a_S p'_S + a_T p'_T + a_B p'_B + b, \qquad (5.21)$$

where

$$\begin{aligned}
a_E &= \rho_e d_e \Delta y\,\Delta z,\\
a_W &= \rho_w d_w \Delta y\,\Delta z,\\
a_N &= \rho_n d_n \Delta z\,\Delta x,\\
a_S &= \rho_s d_s \Delta z\,\Delta x,\\
a_E &= \rho_t d_t \Delta x\,\Delta y,\\
a_B &= \rho_b d_b \Delta x\,\Delta y,\\
a_P &= a_E + a_W + a_N + a_S + a_T + a_B,\\
b &= \frac{(\rho_P^0 - \rho_P)\Delta x\,\Delta y\,\Delta z}{\Delta t} + [(\rho u^*)_w - (\rho u^*)_e]\Delta y\,\Delta z + [(\rho v^*)_s\\
&\quad -(\rho v^*)_n]\Delta z\,\Delta x + [(\rho w^*)_b - (\rho w^*)_t]\Delta x\,\Delta y.
\end{aligned}$$

Since the values of the density ρ will normally be available only at the main grid points, the face densitied such as ρ_e may be calculated by any convenient interpolation. It can be seen from the last equation that the term b in the pressure correction equation is essentially the residual of the discretised continuity equation (5.20) evaluated in terms of the starred velocities. If b is zero, it means that the starred velocities, in conjunction with the available value of $\rho_P^0 - \rho_P$, satisfy the continuity equation, and no pressure correction is needed. The term b thus represents a mass source, which the pressure corrections must annihilate.

5.1.4 SIMPLE Algorithm

The correction procedure for the calculation of the flow field is put together as SIMPLE [91], which stands for stands for Semi-Implicit Method for Pressure-Linked Equations. The sequence of operations can be highlighted as follows:

1. Guess the pressure field p^*
2. Solve the momentum equations, such as (5.11)-(5.13), to obtain u^*, v^*, w^*.
3. Solve the p' equation.
4. Calculate p from (5.14) by adding p' to p^*.
5. Calculate u, v, w from their starred values using the velocity-correction formulas (5.16)-(5.18).
6. Solve the discretisation equation for other variables (such as temperature, concentration) if they influence the flow field through fluid properties, source terms, etc. If a particular solved variable does not influence the flow field, it is better to calculate it after a converged solution for the flow field has been obtained.
7. Treat the corrected pressure p as a new guessed pressure p^*, return to step 2, and repeat the whole procedure until a converged solution is obtained.

5.1.5 TDMA

According to [133], for one-dimensional problem, the default linear solver used by PHOENICS is the tri-diagonal matrix algorithm (TDMA). It is a technique developed by Thomas [116] for rapidly solving tri-diagonal systems, hence also called the Thomas algorithm. TDMA is a direct method for one-dimensional situations, but it can be applied iteratively, in a line by line fashion, to solve multi-dimensional problems. In this section, the solution procedure of TDMA is explained according to [121]. Consider a system of equations that has a tri-diagonal form :

$$\left.\begin{array}{llll} \phi_1 & & & = C_1 \\ -\beta_2\phi_1 & +D_2\phi_2 & -\alpha_2\phi_3 & = C_2 \\ & -\beta_3\phi_2 & +D_3\phi_3 - \alpha_3\phi_4 & = C_3 \\ & & -\beta_4\phi_3 + D_4\phi_4 - \alpha_4\phi_5 & = C_4 \\ & & \cdots\cdots\cdots\cdots\cdots\cdots & = . \\ & & -\beta_n\phi_{n-1} + D_n\phi_n - \alpha_n\phi_{n+1} & = C_n \\ & & \phi_{n+1} & = C_{n+1} \end{array}\right\} \tag{5.22}$$

In the above set of equations, ϕ_1 and ϕ_{n+1} are known boundary values. The general form of any single equation is

$$-\beta_j\phi_{j-1} + D_j\phi_j - \alpha_j\phi_{j+1} = C_j \tag{5.23}$$

The set of equations in (5.22) can be rewritten as

$$\phi_2 = \frac{\alpha_2}{D_2}\phi_3 + \frac{\beta_2}{D_2}\phi_1 + \frac{C_2}{D_2} \tag{5.24}$$

$$\phi_3 = \frac{\alpha_3}{D_3}\phi_4 + \frac{\beta_3}{D_3}\phi_2 + \frac{C_3}{D_3} \tag{5.25}$$

$$\phi_4 = \frac{\alpha_4}{D_4}\phi_5 + \frac{\beta_4}{D_4}\phi_3 + \frac{C_4}{D_4} \tag{5.26}$$

.

$$\phi_n = \frac{\alpha_n}{D_n}\phi_{n+1} + \frac{\beta_n}{D_n}\phi_{n-1} + \frac{C_n}{D_n} \tag{5.27}$$

These equations can be solved by forward elimination and back-substitution. The forward elimination process starts by removing ϕ_2 from equation (5.25) by substitution from equation (5.24) to give

$$\phi_3 = \left(\frac{\alpha_3}{D_3 - \beta_3\frac{\alpha_2}{D_2}}\right)\phi_4 + \left(\frac{\beta_3(\frac{\beta_2}{D_2}\phi_1 + \frac{C_2}{D_2}) + C_3}{D_3 - \beta_3\frac{\alpha_2}{D_2}}\right) \tag{5.28}$$

Let

$$A_2 = \frac{\alpha_2}{D_2} \,\text{and}\, C_2' = \frac{\beta_2}{D_2}\phi_1 + \frac{C_2}{D_2}$$

equation (5.28) can be written as

$$\phi_3 = \left(\frac{\alpha_3}{D_3 - \beta_3 A_2}\right)\phi_4 + \left(\frac{\beta_3 C_2' + C_3}{D_3 - \beta_3 A_2}\right) \tag{5.29}$$

Let also

$$A_3 = \frac{\alpha_3}{D_3 - \beta_3 A_2} \quad \text{and} \quad C_3' = \frac{\beta_3 C_2' + C_3}{D_3 - \beta_3 A_2}$$

equation (5.29) can be re-cast as

$$\phi_3 = A_3\phi_4 + C'_3 \tag{5.30}$$

Formula (5.30) can now be used to eliminate ϕ_3 from (5.26) and the procedure can be repeated up to the last equation of the set. This constitutes the forward elimination process. For the back-substitution, the general form of recurrence relationship (5.30) is used:

$$\phi_j = A_j\phi_{j+1} + C'_j \tag{5.31}$$

where

$$A_j = \frac{\alpha_j}{D_j - \beta_j A_{j-1}} \tag{5.32}$$

$$C'_j = \frac{\beta_j C'_{j-1} + C_j}{D_j - \beta_j A_{j-1}} \tag{5.33}$$

The formulae can be made to apply at the boundary points $j = 1$ and $j = n + 1$ by setting the following values for A and C':

$$A_1 = 0 \quad \text{and} \quad C'_1 = \phi_1$$

$$A_{n+1} = 0 \quad \text{and} \quad C'_{n+1} = \phi_{n+1}$$

In order to solve a system of equations it is first arranged in the form of equation (5.23) and α_j, β_j, D_j and C'_j are identified. The values of A_j and C'_j are subsequently calculated starting at $j = 2$ and going up to $j = n$ using (5.32 - 5.33). Since the value of ϕ is known at boundary location $(n + 1)$ the values for ϕ_j can be obtained in reverse order $(\phi_n, \phi_{n-1}, \phi_{n-2}, ..., \phi_2)$ by means of the recurrence formula (5.31).

5.1.6 Stone method

The TDMA algorithm is the default linear solver for one-dimensional problem in PHOENICS. For higher dimensional problem, a Stone-like extension of the TDMA is used, which requires iteration [133]. A brief approach of this method is illustrated in [114]. Consider the system of algebraic equations arising from the use of a discretisation scheme:

$$A\,x = b$$

where A is the relatively sparse matric of known coefficients, x is the column vector of unknowns, and b is a column vector of known quantities. It is well known that if the matrix

A could be factored into the product of upper and lower triangular matrices, the solution for x could proceed in two sweeps, involving only forward and backward substitution. However, a complete factorisation of A may require a very large computational effort. The strongly implicit procedure proposed by Stone [111] is one example of an approximate factorisation strategy which requires less effort than the complete factorisation. The objective of this method is to replace the sparse matrix A by a modified form $A + N$ such that the modified matrix can be decomposed into upper and lower triangular sparse matrices denoted by U and L respectively, then solving for x iteratively. The matrix equation can be seen by splitting the matrix A in :

$$Ax = (M - N)x = b, \text{with } ||M|| >> ||N||$$

The incomplete LU factorization of A is then defined in the iterative fashion:

$$Mx(k+1) = Nx(k) + b$$

Decomposing M into the upper and lower triangular matrices L and U, the iterative procedure can be written as :

```
set a guess
  $k = 0, x(k)$
  $r(k) = b - Ax(k)$
while $(||r(k)|| \geq \varepsilon)$do
  evaluate new right hand side :    $c(k) = Nx(k) + b$
  solve  $Ly(k) = c(k)$ by forward substitution
  solve  $Ux(k+1) = y(k)$ by back substitution
  $k := k + 1$
end while
```

Stone [111] selected N so that L and U have only three nonzero diagonals, the principal diagonal of U being the unity diagonal. Furthermore, the elements of L and U were determined such that the coefficients in the B matrix in the locations of nonzero entries of matrix A were identical with those in A. Two additional nonzero diagonals appear in B. The elements of L, U, and N can be determined from the defining equations established by forming the LU product. The details of this are given in [111].

5.2 A 2D Single-Phase Test Problem

The first test problem which will be implemented in PHOENICS is a 2D fluid flow with heat transfer. Some simplifications are made so that the comparison with the domain decomposition results (in the next chapter) can be easily investigated. The domain in which the fluid flows is a porous medium, thus the Darcy's Law applies. A steady state condition and a constant fluid density are assumed for this problem. The fluid flow and the energy transport are governed by the following differential equations :

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu}\nabla p \tag{5.34}$$

$$\nabla.\mathbf{u} = 0 \tag{5.35}$$

$$\nabla.(\rho\mathbf{u}H) = \nabla.(k\nabla T) \tag{5.36}$$

The first equation, also known as the Darcy's Law, is an expression of conservation of momentum which can be derived from the Navier-Stokes equations via homogenization [115] [86]. As usual, $\mathbf{u}$ is the velocity, ∇p is the pressure gradient and μ is the viscosity of the fluid. The tensor permeability $\mathbf{K}$ in two dimensions is represented by

$$\mathbf{K} = \begin{bmatrix} K_x & 0 \\ 0 & K_y \end{bmatrix}$$

In (5.36), k is the thermal conductivity, while the enthalpy H and the temperature T have a linear relation $H = c_pT$ where C_p is the specific heat capacity.

The domain of the problem is illustrated in Fig.5.10 and the scale is depicted in Fig.5.7. It is enclosed by two horizontal walls with length $L = 1$m, and the two parallel walls are separated by a gap of $W = 0.1$m. The fluid is liquid water with the following properties : thermal conductivity $k = 0.609$ W/mK, heat capacity $C_p = 4181.8\,$kJ/(kgC), and viscosity $\mu = 10^3$ kg/(m.s). The inflow of fluid is driven from the left side by a horizontal velocity of $u = 0.01\,$m/s, with a constant density $\rho = 998.23\,$kg/m^3 and temperature $T = 50^0$C. It exits the domain at the right hand side where the pressure is fixed at the atmospheric pressure. In terms of permeability, the porous domain is divided into two areas where the first half of the domain ($x \in [0, L/2]$) takes the permeability $\mathbf{K_1}$ while the second half ($x \in [L/2, 1]$) takes $\mathbf{K_2}$ with the following values:

$$\mathbf{K_1} = \begin{bmatrix} \frac{1}{3}.10^{-6} & 0 \\ 0 & \frac{1}{3}.10^{-6} \end{bmatrix} \; ; \quad \mathbf{K_2} = \begin{bmatrix} 10^{-6} & 0 \\ 0 & 10^{-6} \end{bmatrix}$$

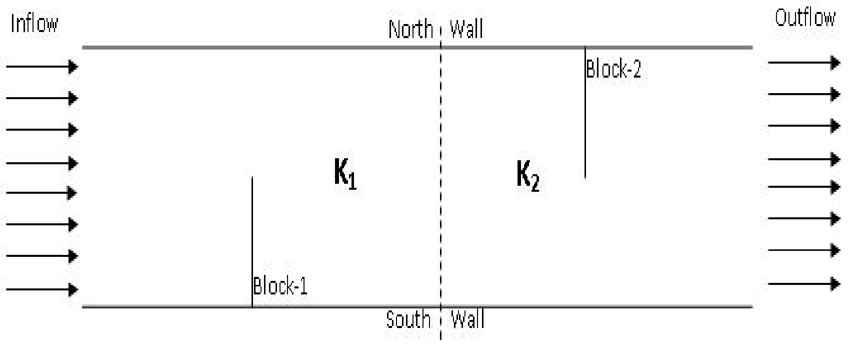

Figure 5.6: Domain of 2D problem with fluid flow and heat transfer

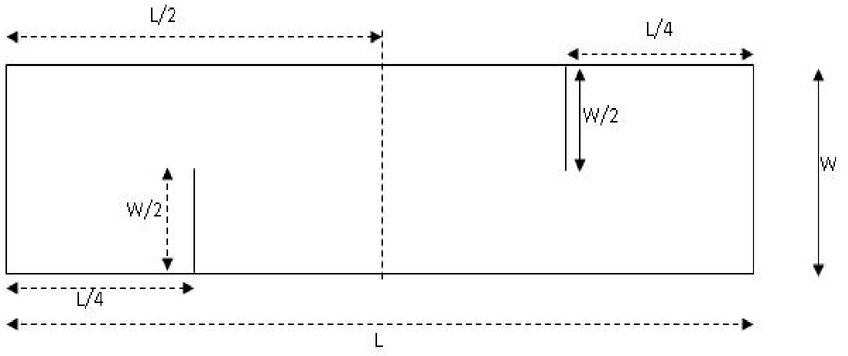

Figure 5.7: Scale of the 2D problem

As can be seen in Fig.5.10, there are two vertical blocks (represented in one-dimension) with the size of $W/2$ in the domain. These blocks can be regarded as impermeable walls where no-slip condition applies. For the horizontal walls, in addition to the no-slip condition, the temperature is kept at 0^0C at the bottom (South Wall) and 100^0C at the top (Northern Wall).

Although the form of Darcy's equation (5.34) is different from that of the momentum equation (5.2), the coupled system of (5.34) and (5.34) can be solved with the pressure correction method. The adjustment is made in PHOENICS by deactivating the transient, convective and diffusive terms of (5.34) and applying a substitution $\mathbf{F} = -\mu\mathbf{K}^{-1}\mathbf{u}$. This results in

$$\cancel{\frac{\partial \rho \mathbf{u}}{\partial t}} + \cancel{\nabla.(\rho \mathbf{u}\mathbf{u})} = -\nabla p + \cancel{\nabla.(\mu\nabla)\mathbf{u}} + -\mu\mathbf{K}^{-1}\mathbf{u}$$

which reduces to (5.34). In terms of discretisation, the finite volume integration of the source term $\mu\mathbf{K}^{-1}\mathbf{u}$ contributes to the coefficient of the left-hand side of equations (5.9) and (5.10) in the SIMPLE correction procedure, where

$$a_e = -(\Delta V_e)\frac{\mu}{K_x}$$

for the x-momentum equation (5.9) with the u-control volume shown in Fig.5.3, and

$$a_n = -(\Delta V_n)\frac{\mu}{K_y}v_n$$

for the y-momentum equation (5.10) with the v-control volume shown in Fig.5.4. The coefficients ΔV_e and ΔV_n above are the volume of the corresponding CV's. Due to the cancellation of the transient, convective and diffusive terms, all coefficients a_{nb} in (5.9) and (5.10) now vanish in this case.

A number of computations are run in PHOENICS to solve the problem (5.36- 5.34). Some uniform rectangular meshes are generated with the following number of cells : 41x8, 101x20, 201x40, 401x80. The number of cells in horizontal direction is set with odd numbers (41, 101, 201, 401) instead of even numbers (40, 100, 200, 400) so that mid-points may lie at the centre of interface cells. In staggered grid arrangement such as the one applied in PHOENICS, scalar variables are stored at cell centres while velocity components at face centres. As will be schown in the next chapter, all interface variables for this problem are scalar variables, therefore the interface line should be a line cutting through cell centres where the interface variables are stored.

The reference solution for this problem is taken as the solution obtained on the finest mesh (401x80). The contour of pressure and temperature of this reference solution are illustrated in Fig.5.8 and Fig.5.9. The pressure difference in the first half (left side) of the domain is larger because it is less permeable than the second half (right side). Table-5.1 demonstrates the error of the coarser solution with respect to the reference solution. The variables with subscript ref denote the reference solution while those with subscript h denote the solution for the given mesh size. In the table, the errors of pressure and enthaply are computed with Euclidean and ∞-norm. The decrease of errors with the mesh refinement shows an indication of convergence to the reference solution.

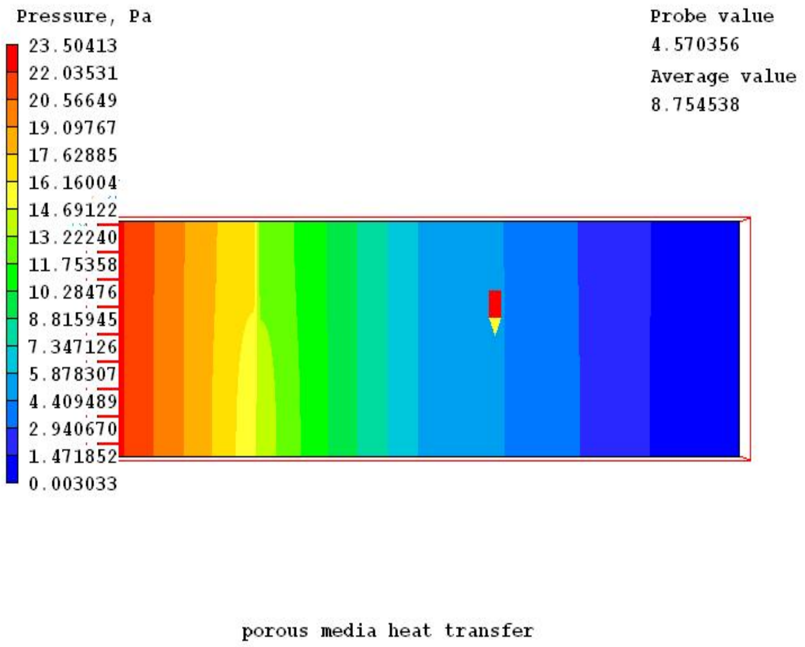

Figure 5.8: Contour of pressure

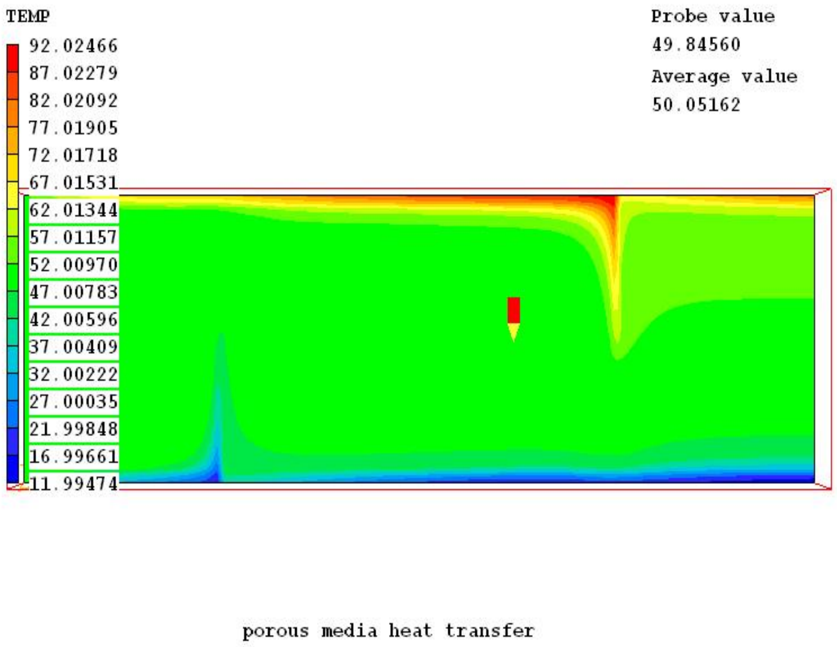

Figure 5.9: Contour of temperature

Table 5.1: Error with respect to the reference solution

ncells	$\|\|p_{ref} - p_h\|\|_\infty$	$\frac{\|\|p_{ref} - p_h\|\|}{\|\|p_{ref}\|\|}$	$\|\|H_{ref} - H_h\|\|_\infty$	$\frac{\|\|H_{ref} - H_h\|\|}{\|\|H_{ref}\|\|}$
41 x 8	0.1656	0.0300	5.9470e+4	0.0512
101 x 20	0.1028	0.0186	5.2483e+4	0.0433
201 x 40	0.0442	0.0080	3.4469e+4	0.0273

5.3 A Multiphase Flow Problem with Heat and Moisture Transfer in Porous Textile Materials

This section is concerned with a multiphase fluid flow with heat and moisture transfer in fibrous textile. The governing equations are presented in a macro scale form. A computational problem is specified with some simplifications and additional constitutive equations. The numerical solutions are obtained using PHOENICS. The domain decomposition implementation of this problem will be discussed in Chapter-6.

5.3.1 Introduction

Textile industries have continuously developed due to demands in many fields. The need of the industries are not only for human basic needs, but also ranges from interests of fashion to highly protective outfits. Fashion industries require the combination of splendour and wearing comfort, sportsmen demand a clothing design which eases the essential physical movement under various physical situations whereas military personnel or fire fighters compel a protective immunity of their outfits against chemical, biological or burning objects [9].

Not solely for humanwear, high concern of textile technology lies also on the product packaging. Numerous equipments and devices make use of layers of fabric with particular properties to support their functions, maintain protection and increase ease of use. In building construction, acoustics design might need some fabric coating to reduce unwanted noise [17]. In general, exposure and exploration of textile properties and capabilities leads to more innovations and advance of other supported industries.

Heat and moisture transfer are key physical processes occuring in fabrics [52]. Conduction, convection, and radiation make up the heat transfer process while advection and diffusion consitute the mass and momentum evolution of vapour within fibrous material.

Phase changes also become a significant factor in the physical process given the presence of moisture in the fabric. Condensation and evaporation constantly happens throughout the process between the vapour and liquid state. In some circumstances, the impact can be appalling. An example is in sportswear and clothing worn in cold climate, where the moisture accumulation resulting from absorption or condensation within the clothing is a serious problem. In order to avert unfavourable effects of this phase change, heat and moisture transfer within textile arrangement need to be better examined.

Computational models have become one of the key development tools in clothing technologies [2]. Understanding the physical processes at the scale of individual fibres and testing the parameter influences can be difficult and expensive to do for a range of parameters. CFD provides the basis for the computational models needed. The outputs of the computational process are the flow velocity, pressure, temperature, and mass or concentration of corresponding material in its computational domain. When fabrics are in contact with moisture, the multiphase structure of the process comprises the solid phase (fibre), liquid phase (water) and gaseous phase (vapour and dry air) [52]. The fibrous system can then be represented by an ideal continuum medium which is divided into volume fractions of liquid, solid, and liquid vapour and air mixture at a particular location. The inhomogeneity scale of the solid, liquid and gas phase mixture is far smaller than the characteristic length over which an appreciable change in moisture content occurs.

It is well known that modelling the macro processes in fibrous material is not simple. Moisture acts as a significant agent which adds to the complexity of the process by taking part in the phase change [51]. The phase change of vapour between gaseous and liquid phase is induced by the local thermal condition of fabric. When evaporation/condensation occurs, there will be heat absorbed/released. Fibres have the ability to adsorp/desorp moisture of either phase, which will also change the thermal condition of the fabric. The diffusion coefficient which governs the sorption is conditioned by temperature and relative humidity. Hence moisture behaviour brings about the coupling of mass and energy equations. Modelling the moisture movement is a significant task in the accuracy of the computation.

An example of a general micro scale multiphase model in a porous medium is described in [37], from which bigger scale models can be derived. Techniques such as averaging and homogenisation methods [92] [7] have been used to extract some macro parameters from their corresponding micro scale models. Yet this research is not aimed at micro scale models nor the extraction techniques. Rather it deals with a macro scale model of a multiphase

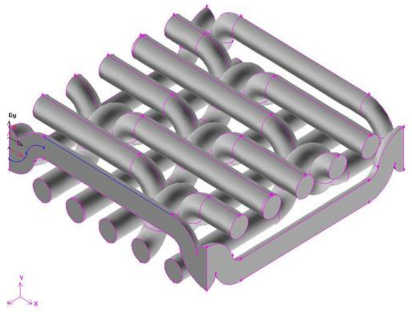

Figure 5.10: A cell of fabric

process.

5.3.2 A Multiphase Process in Fabric

When performing CFD studies at the scale of a full human body, it is not feasible to account for phenomena at the much smaller scales. Therefore, at macro-scale, models should be based on large-scale, ensemble-averaged CFD descriptions of the process, using averaging techniques for modelling the governing equations. A textile material consists of multi-cells, where a simple cell, illustrated in Fig.5.10, represents an area with certain regular porosity. Rather than accounting for the precise geometrical structure of the textile, the textile material will be modelled in a lumped fashion as a porous material with particular physical properties. In the following, continuum hydrodynamics is used to model the macroscopic governing equations, where some physical properties/parameters have to be inferred from either experiment or computation of micro/mesoscale model in a smaller unit length scale.

In the derivation of the balance equations, it is assumed that fibrous material is isotropic, no change of volumes in the solid part, the moisture content at the fibre surface is in sorptive equilibrium [129] with that of the surrounding air. According to the mass balances, the mass governing equations [130] [90] in 1D are:
for whole gas :

$$\frac{\partial(\epsilon_g C_g)}{\partial t} = -\frac{\partial(\epsilon_g V_g C_g)}{\partial x} - \varpi_1 \epsilon_f \frac{\partial C_f}{\partial t} + Q_p \tag{5.37}$$

for vapour only :

$$\frac{\partial(\epsilon_g C_a)}{\partial t} = \frac{\partial}{\partial x}\left[(D_M)\epsilon_g \frac{\partial(C_a)}{\partial x}\right] - \frac{\partial(\epsilon_g V_g C_a)}{\partial x} - \varpi_2 \epsilon_f \frac{\partial C_f}{\partial t} + Q_p \tag{5.38}$$

for liquid water :
$$\frac{\partial(\epsilon_l \rho_l)}{\partial t} = -\frac{\partial(\epsilon_l V_l \rho_l)}{\partial x} - \varpi_2 \epsilon_f \frac{\partial C_f}{\partial t} - Q_p \tag{5.39}$$

In those balances, ϵ_f , ϵ_g , ϵ_l represent the volume fraction of solid, gas and liquid respectively and they are related by

$$\epsilon_f + \epsilon_g + \epsilon_l = 1.$$

The velocities of gas and liquid are denoted by V_g and V_l while C_g , C_f and ρ_l represent the gas, fibre and liquid density respectively. Another concentration in the second equation, C_a is the concentration of vapour which flows within the gas phase [130] and additionally driven by a molecular diffusion coefficient D_M . The evaporation rate of liquid water on the fibres is denoted by Q_p , whereas ϖ_1 and ϖ_2 represent the proportion of the sorption of water vapour and liquid water, respectively, by the fibres. The absorption or desorption rate of moisture by the fibres obey the Fickian diffusion law [82]:

$$\frac{\partial C_f}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(D_f \frac{\partial C_f}{\partial r})$$

where D_f is the diffusion coefficient of water vapour in the fibres, and r is the radial coordinate in a fibre. The boundary condition is determined by the relative humidity of the air surrounding a fibre.

Relationships of pressure drop and velocity of gas and liquid satisfy the Darcy's Law :

$$\epsilon_g V_g = -\frac{K K_{rg}}{\mu_g}\frac{\partial p_g}{\partial x} \tag{5.40}$$

$$\epsilon_l V_l = -\frac{K K_{rl}}{\mu_l}\frac{\partial p_l}{\partial x} \tag{5.41}$$

In those equations, p_g and p_l are gas and liquid pressure respectively, while the dynamic viscosity of gas and liquid are denoted by μ_g and μ_l . Permeabilities of both phases are reflected through three terms K , K_{rg} and K_{rl} where the first is the intrinsical permeability in the fibre, whereas the latter are the relative permeability of gas and liquid respectively. The pressure drop of liquid and gas phase in (5.40) and (5.41) are related by:

$$p_l = p_g - p_c \tag{5.42}$$

with p_c being the capillary pressure. The energy equations are solved for each phase, governed by :

$$\frac{\partial(\epsilon_g C_g H_g)}{\partial t} + \frac{\partial(\epsilon_g C_g V_g H_g)}{\partial x} = \frac{\partial}{\partial x}\left[\epsilon_g k_g \frac{\partial(T_g)}{\partial x}\right] + \lambda_v \varpi_1 \epsilon_f \frac{\partial C_f}{\partial t} + \Delta H_g(Q_p), \text{ in the g-phase} \tag{5.43}$$

$$\frac{\partial(\epsilon_l \rho_l H_l)}{\partial t} + \frac{\partial(\epsilon_l \rho_l V_l H_l)}{\partial x} = \frac{\partial}{\partial x}\left[\epsilon_l k_l \frac{\partial(T_l)}{\partial x}\right] + \lambda_l \varpi_2 \epsilon_f \frac{\partial C_f}{\partial t} + \Delta H_l(Q_p), \text{ in the } l\text{-phase} \tag{5.44}$$

where H_g, H_l, T_g, T_l, k_g, and k_l are the enthalphy of gas and liquid, the temperature of gas and liquid, and the heat conductivity of gas and liquid respectively. λ_l and λ_v represent the sorption latent heat coefficients of the liquid and vapour respectively while $\Delta H_l(Q_p)$ and $\Delta H_g(Q_p)$ signify the rate of enthalpy change of liquid and gas associated with the rate of evaporation (Q_p). The equations exemplified above represent a rather general macro-scale model of multiphase problem in porous mediums. Some of terms still need to be described more explicitly in order to complete the model. These will be shown in a more specific case within the next section.

5.3.3 Problem Description

While the previous section presents the general model of multiphase process in fabric, this section describes a more specific problem by giving further details of the fabric properties along with some simplification of the process and the boundary condition as well. This problem will be solved using PHOENICS and the results will be demonstrated in later sections of this chapter.

The scope of the problem is a one dimensional textile material of 5mm thickness. A steady state condition is assumed to take place. Therefore, the transient terms associated with the storage of variables and sorption in fibres are dropped from the equations (5.37), (5.38), (5.39), (5.43), and (5.44). The governing equations become :

for continuity of gas :

$$\frac{\partial(\epsilon_g V_g C_g)}{\partial x} = Q_p \tag{5.45}$$

for continuity of vapour :

$$\frac{\partial(\epsilon_g V_g C_a)}{\partial x} = \frac{\partial}{\partial x}\left[(D_M)\epsilon_g \frac{\partial(C_a)}{\partial x}\right] + Q_p \tag{5.46}$$

for continuity of liquid water :

$$\frac{\partial(\epsilon_l V_l \rho_l)}{\partial x} = -Q_p \tag{5.47}$$

for enthalpy of gas :

$$\frac{\partial(\epsilon_g C_g V_g H_g)}{\partial x} = \frac{\partial}{\partial x}\left[\epsilon_g k_g \frac{\partial(T_g)}{\partial x}\right] + \Delta H_g(Q_p), \text{ in the g-phase} \tag{5.48}$$

for enthalpy of liquid water :

$$\frac{\partial(\epsilon_l \rho_l V_l H_l)}{\partial x} = \frac{\partial}{\partial x}\left[\epsilon_l k_l \frac{\partial(T_l)}{\partial x}\right] + \Delta H_l(Q_p), \text{ in the } l\text{-phase} \tag{5.49}$$

The diffusion parameter values in (5.46), (5.48), and (5.49) are given by $D_M = 2.5\,10^5\,m^2/s$, $k_g = 0.024$ W/mK and $k_l = 0.609$ W/mK respectively. Taking the reference zero as the base enthalpy and temperature, the phase enthalpies H_g and H_l can be related to phase temperatures H_g and H_l using their respective specific heat capacity, $c_{p_g} = 1004\,\text{kJ/(kgC)}$ and $c_{p_l} = 4181.8\,\text{kJ/(kgC)}$ via

$$H_g = c_{p_g}\,T_g, \quad H_l = c_{p_l}\,T_l.$$

In the liquid transport (5.47) and (5.49),the density of liquid is kept constant at $\rho_l = 998.23\,\text{kg}/m^3$, while the capillary pressure in (5.42) which will be significant for the advection of liquid is expressed [130] by:

$$p_c = \frac{2\sigma\epsilon cos\Theta}{\epsilon_l d_c}$$

where the terms $\sigma = 0.031N/m$, $\Theta = 80^0$, $d_c = 5.7\ 10^{-7}$m, and ϵ denote the surface tension, contact angle, effective pore radius, and the porosity of the fibre respectively . The evaporation rate Q_p which appears in each balance equation is modelled [130] by

$$Q_p = \epsilon_g h_{l\leftrightarrow g} S_v (C_a^*(T_g) - C_a) \tag{5.50}$$

where $S_v = 10^4/m$ denotes the specific volume of fabric, $h_{l\leftrightarrow g} = 0.0137m/s$ denotes mass transfer coefficient, and $C_a^*(T_g)$ is the saturated water vapour concentration which is determined by vapour temperature T_g and defined [28] by:

$$C_a^*(T) = \frac{0.0035}{T} 10^{(8.07131 - \frac{1730.63}{T+233.426})} \tag{5.51}$$

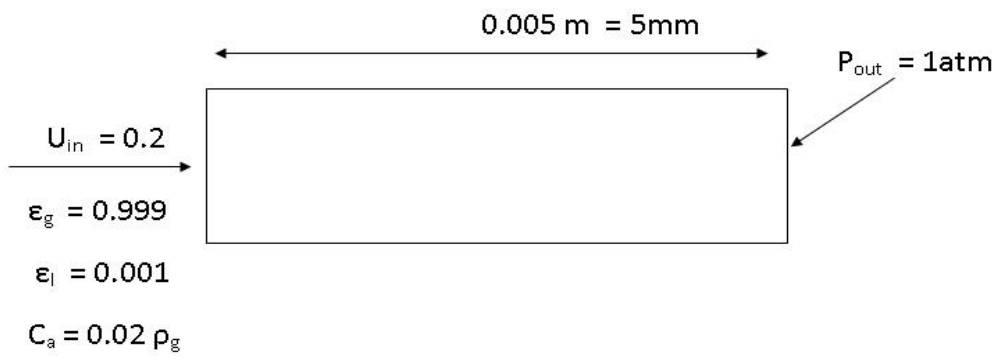

Figure 5.11: Condition at the left and right boundary

The heat associated with the evaporation rate Q_p is given by

$$\Delta H_l(Q_p) = -\max(\lambda Q_p, 0)$$

$$\Delta H_g(Q_p) = \max(-\lambda Q_p, 0)$$

with $\lambda = 2522$ kJ/kg being the latent heat of evaporation.

For the Darcy equations (5.40) and (5.41), the dynamic viscosity of gas and liquid are $\mu_g = 1.8310^5$ kg/(m.s) and $\mu_l = 10^3$ kg/(m.s) respectively. The intrinsical permeability K, as opposed to the local effective permeability resulting from homogenisation theory, is obtained from [130]:

$$K = \frac{3\epsilon sin^2\beta {d_c}^2}{80} \tag{5.52}$$

where ϵ is the porosity of the fibre and $\beta = 20^0$ is the average angle of the capillaries in the fibre. The relative permeability of gas and liquid water are given by

$$K_{rl} = (\epsilon_l/\epsilon)^2,$$

$$K_{rg} = (\epsilon_g/\epsilon)^2$$

To complete the model, the relationship between gas pressure and concentration is given according to the asumption of perfect gas, where the concentration of dry air + vapour can be deduced from :

$$C_g = \frac{p_g}{RT_g}$$

In that relation, the gas constant has the value of $R = 286.4Jkg^{-1}K^{-1}$.

The test problem is depicted in Fig.5.11. The boundary condition imposed at the left hand side (inlet) is a velocity of 0.002 m/s bringing in both phases with the fraction of $\epsilon_g = 0.999$ and $\epsilon_l = 0.001$. Within gas, a 0.02 fraction of vapour flowing into the material. At the right hand side (outlet), the atmospheric pressure of 1 atm applies.

5.3.4 Single layer of fabric

The first test is carried out in a single layer of fabric where throughout the domain, as shown in Fig.5.11, a uniform pore radius applies. A mesh of 100 cells of equal size is employed in this computation. The first simulation is run for a fibrous material with an effective pore radius $d_c = 5.7\ 10^{-6}m$. The linear relationship between the effective pore radius d_c and the intrinsic permeability K in (5.52) implies that the intrinsical permeability of this material is $K \approx 10^{-13}$. Some computations are run in PHOENICS by varying the number of mesh ($ncells = 10, 20, 50, 100$). For the reference solution, the solution with the finest mesh (100 cells) is chosen. Table 5.2-5.3 shows the error of the infinity norm error and the normalized L2-norm of five solved variables (pressure gas, liquid fraction, enthalpy of gas, enthalpy of liquid, vapour density) with respect to the reference solution. The variables with subscript h are those obtained from the computations with corresponding mesh, while variables with subscript ref are the reference solution.

Table 5.2: ∞-norm error

ncells	$\|\|p_{g,ref} - p_{g,h}\|\|_\infty$	$\|\|\epsilon_{l,ref} - \epsilon_{l,h}\|\|_\infty$	$\|\|H_{g,ref} - H_{g,h}\|\|_\infty$	$\|\|H_{l,ref} - H_{l,h}\|\|_\infty$	$\|\|C_{a,ref} - C_{a,h}\|\|_\infty$
10	77.14	2.10 e-3	300.81	634.10	6.76e-4
20	34.27	1.10 e-3	135.70	283.65	3.0e-4
50	8.56	3.06 e-4	34.24	71.23	7.2e-5

Table 5.3: Normalised L^2-norm error

ncells	$\frac{\|\|p_{g,ref}-p_{g,h}\|\|}{\|\|p_{g,ref}\|\|}$	$\frac{\|\|\epsilon_{l,ref}-\epsilon_{l,h}\|\|}{\|\|\epsilon_{l,ref}\|\|}$	$\frac{\|\|H_{g,ref}-H_{g,h}\|\|}{\|\|H_{g,ref}\|\|}$	$\frac{\|\|H_{l,ref}-H_{l,h}\|\|}{\|\|H_{l,ref}\|\|}$	$\frac{\|\|C_{a,ref}-C_{a,h}\|\|}{\|\|C_{a,ref}\|\|}$
10	5.0e-2	1.5e-3	5.8e-3	3.3e-3	8.0e-3
20	2.4e-2	7.3e-4	2.8e-3	1.6e-3	3.9e-3
50	8.6e-3	2.4e-4	9.6e-4	5.6e-4	1.3e-3

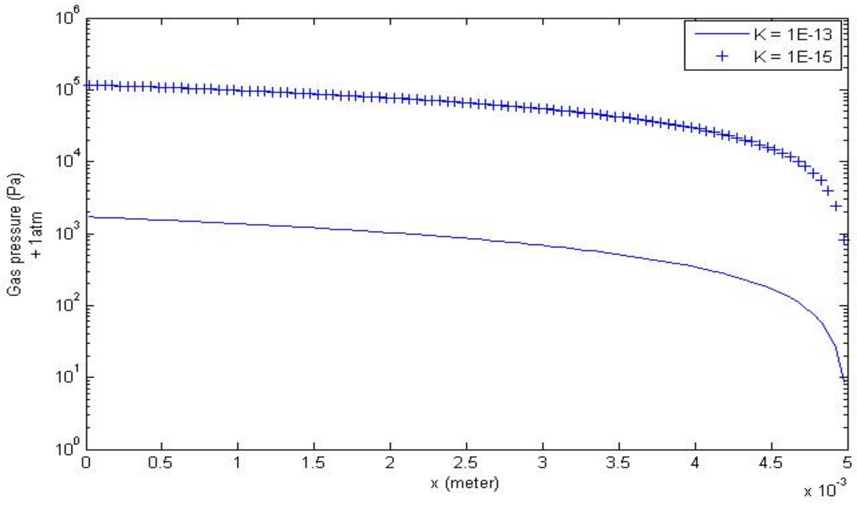

Figure 5.12: Pressure of gas

The aim of running this numerical model is to serve as the reference solutions for comparison with domain decomposition implementation in Chapter-6. Yet, the justification of the obtained results are also given qualitatively, by looking at the relationship among the variables in the underlying model. Along with this, comparisons are also made between solutions with different values of intrinsic permeability K, i,e. $K_1 \approx 10^{-15}$ (given by a fibrous material with an effective pore radius $d_c = 5.7\ 10^{-7}m$), and $K_2 \approx 10^{-13}$ (given by a material with $d_c = 5.7\ 10^{-6}m$).

The graph in Fig.5.12 illustrates the pressure of gas alongside the thickness of the fabric. It is displayed in the logarithmic scale. A higher pressure gradient occurs inside the material with the lower intrinsical permeability ($K_1 \approx 10^{-15}$). This is to compensate the bigger resistance given to the same inflow rate. A better permeability means the ease of penetration of the fluid, thus less pressure is needed to pass the same amount of mass. The pressure gradient is then inversely proportional to the intrinsical permeability K_i. In that graph, the pressure gradient inside the material with intrinsical permeability K_1 is almost 100 times that in the material with the lower intrinsical permeabiliy K_2, which is approximately the ratio K_1/K_2.

The temperature of gas is bounded between 25^0C at the left boundary and 37^0C at the right boundary (between 298K and 310K). From the ideal gas law, the effect of this temperature variation to gas density is far less dominant than the pressure variation. Accordingly

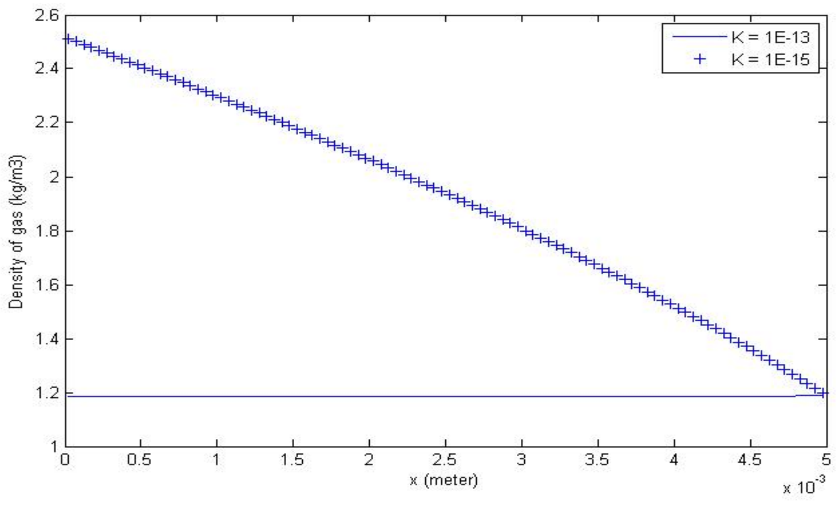

Figure 5.13: Density of gas

the higher pressure in lower intrinsical permeability contribute to the steeper variation of gas density, whereas the density in higher intrinsical permeability material appears almost constant because the pressure difference between the left and right boundary is only about 10^3 Pa which is negligible to atmospheric pressure 1 atm. The profile of gas density is given in Fig.5.13.

A sharp decline of gas density along the fabric thickness means that the gas will occupy more volume for the same amount of mass. This is especially valid when the evaporation/condensation rate is not large, as shown in Fig.5.14. It is then not difficult to appreciate that the volume fraction of gas increase much faster for lower intrinsical permeability K_1 as illustrated in Fig.5.15.

From (5.40), it follows that

$$V_g = -\frac{K}{\epsilon^2 \mu_g} \frac{\partial p_g}{\partial x} \epsilon_g$$

since $K_{rg} = (\epsilon_g/\epsilon)^2$. It is recently discussed that the pressure gradient is inversely proportional to the intrinsical permeability K. This leads to the relation that the gas velocity is proportional to the volume fraction of gas. This may explain the profile of gas velocity in Fig.5.16 which shows similar shape as the gas volume fraction in Fig.5.15.

With the density and velocity of gas obtained in Fig.5.13 and 5.16 respectively, the Peclet number accross the thickness can be plotted. Fig.5.17 shows low Peclet number for both permeabilities which indicate the domination of heat conduction over convection. This agrees

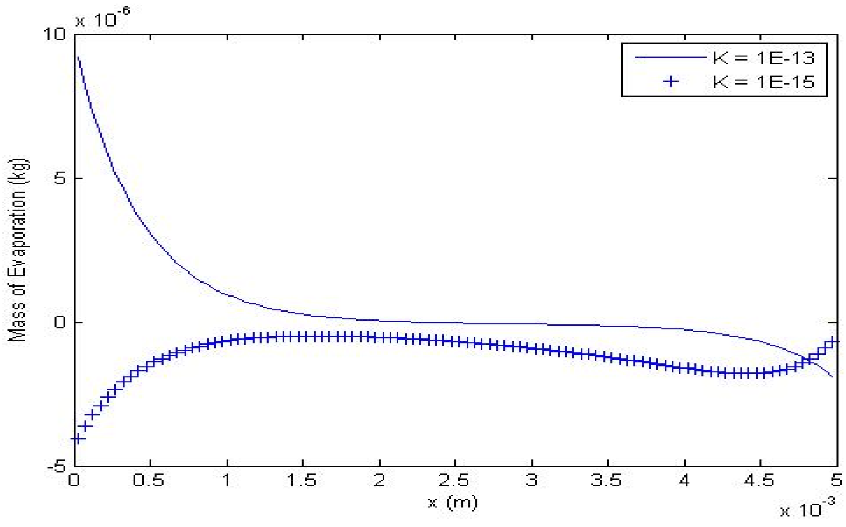

Figure 5.14: Evaporation

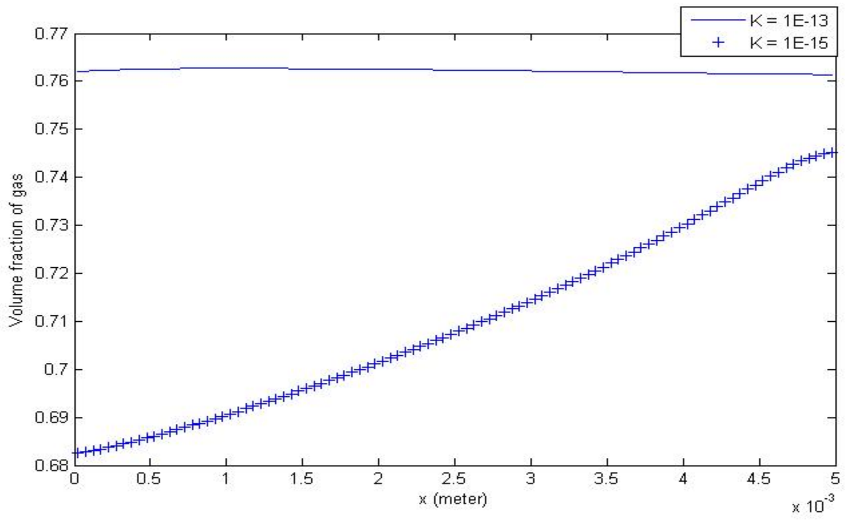

Figure 5.15: Volume fraction of gas

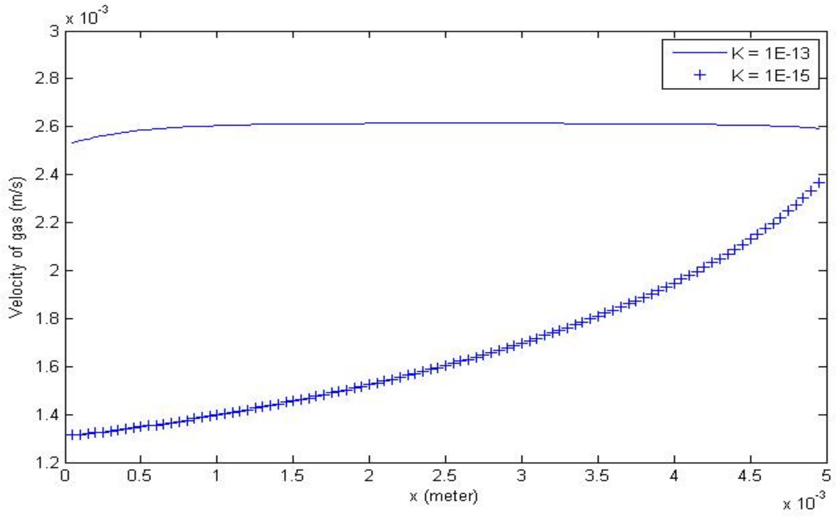

Figure 5.16: Velocity of gas

with the numerical solution of gas temperature in Fig.5.18 which resembles a straight line from the left to the right boundary.

In the mass transport of vapour, with diffusion coefficient $D_M = 2.5\ 10^5\ m^2/s$, the diffusion also dominates advection. The solution obtained for vapour density in Fig.5.19 reflects this condition. The vapour content in Fig.5.20 is simply the ratio between the vapour and gas density.

5.3.5 Two layers of fabric

The second test is a domain with two layers of fabric where the difference of the two layers lie in the value of effective pore radius. The boundary condition imposed at the left and right hand side (inlet) are exactly the same as in the first test. The size of the domain is also the same but now divided into two layers of equal length, as shown in Fig.5.21.

Layer-1 is a textile fabric with effective pore radius d_c= 5.710^{-7} m, whereas the pore radius of Layer-2 is d_c= 5.710^{-6} m. Accordingly, the intrinsic permeabilities of Layer-1 and Layer-2 are $K_1 \approx 10^{-15}$ and $K_2 \approx 10^{-13}$. The mesh setting is made the same as in the single layer problem ($ncell = 10, 20, 50, 100$) and the finest mesh solution ($ncell = 100$) is regarded as the reference solution. Table 5.4- 5.5 list the error of coarser mesh solution relative to the reference solution.

The computational results of the problem are presented in Fig.5.22 - Fig. 5.25. Again,

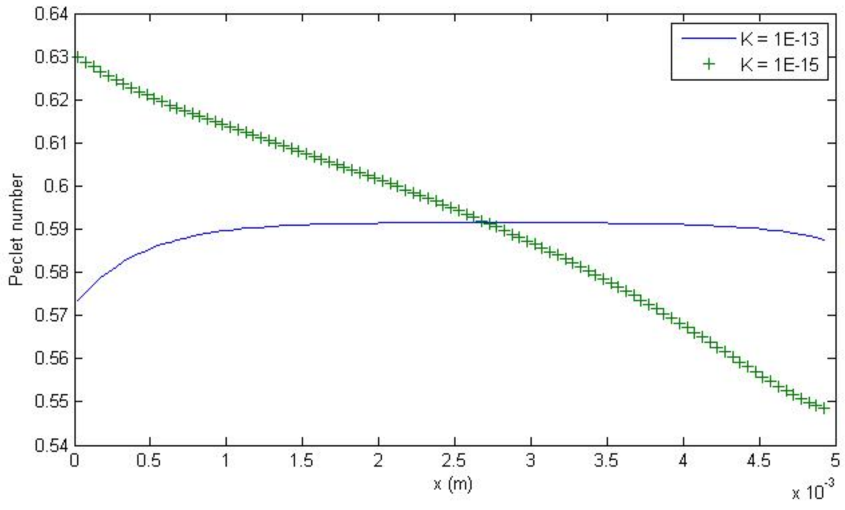

Figure 5.17: Peclet number

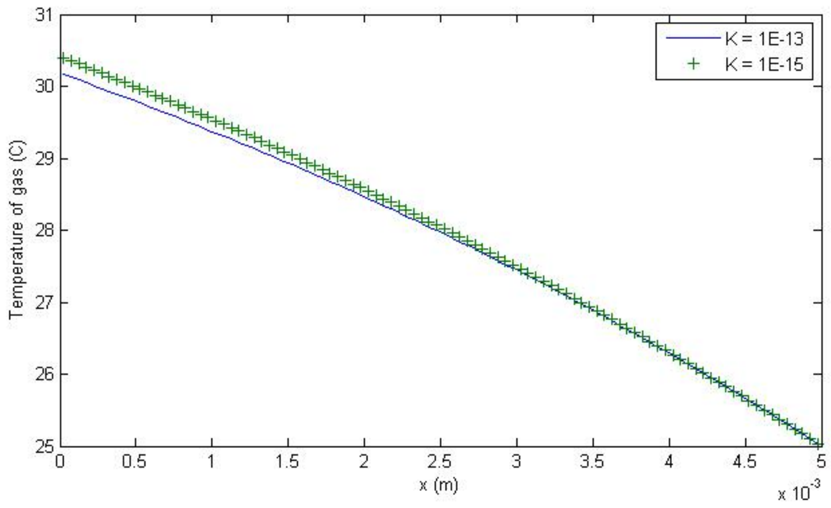

Figure 5.18: Temperature of gas

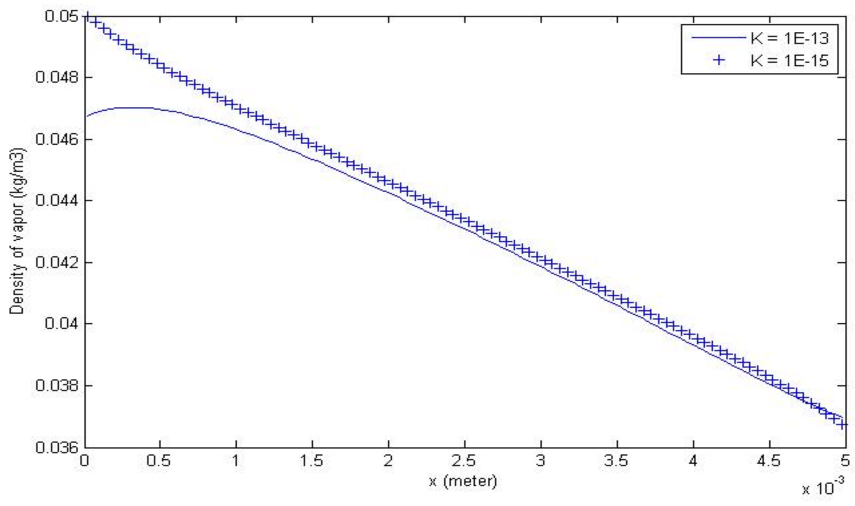

Figure 5.19: Concentration of vapour

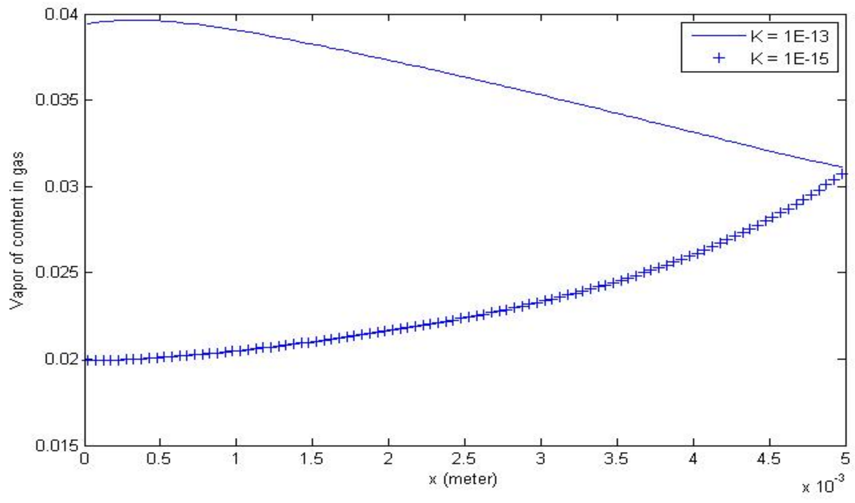

Figure 5.20: vapour content in gas

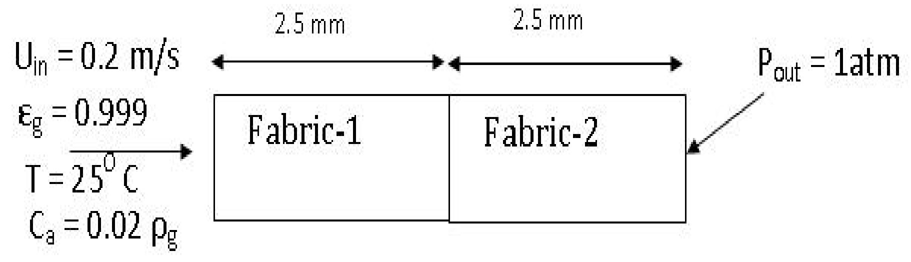

Figure 5.21: Two layers of fabric

Table 5.4: ∞-norm error

ncells	$\|\|p_{g,ref} - p_{g,h}\|\|_\infty$	$\|\|\epsilon_{l,ref} - \epsilon_{l,h}\|\|_\infty$	$\|\|H_{g,ref} - H_{g,h}\|\|_\infty$	$\|\|H_{l,ref} - H_{l,h}\|\|_\infty$	$\|\|C_{a,ref} - C_{a,h}\|\|_\infty$
10	1.7388e+4	0.0147	289.6641	637.0781	0.0028
20	8.3346e+3	0.0075	130.5742	284.8437	0.0017
50	2.2091e+3	0.0020	32.9297	71.9063	0.0005

they will become the reference for comparisons with some domain decomposition computations in Chapter-6. In those figures, the curves undergo a change of shape exactly in the middle ($x = 2.5$ mm) which is the transition of the two different permeabilities.

The logarithmic plot in Fig.5.22 shows that the pressure profile in the first layer ([0,2.5 mm]) has the shape as that in second layer ([2.5mm, 5mm]). In the linear scale, this means the pressure gradient in the first layer is about 100 times that in the second layer. This ratio is approximately the same as that found in the single layer when the pressure profiles of both permeabilities are compared in the entire domain ([0, 5mm]).

If the plots in Fig.5.23, 5.24, 5.25, and 5.26 are compared with Fig.5.16, 5.15, 5.19, and 5.20 respectively, the shapes of the profiles in the two fabric layers are obtained by merging the shape of the profiles of both permeabilties in the single layer.

5.4 Closure

This chapter has described the solution procedure used in the PHOENICS code. A case of 2D fluid flow with heat transfer problem is solved with the code. This chapter has also set out the theory underlying the macroscale equation of multiphase heat and moisture transfer in porous textile. Some 1D steady state problems with evaporation process have been computed

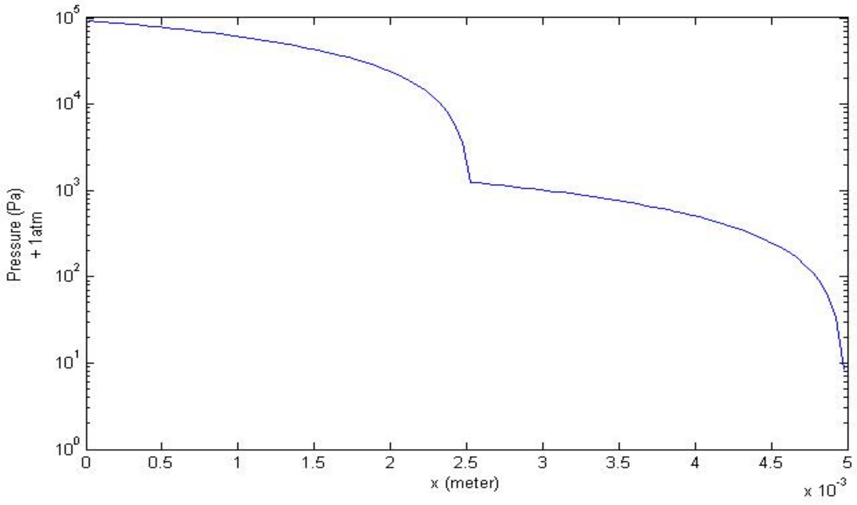

Figure 5.22: Pressure of gas in two layers of fabric

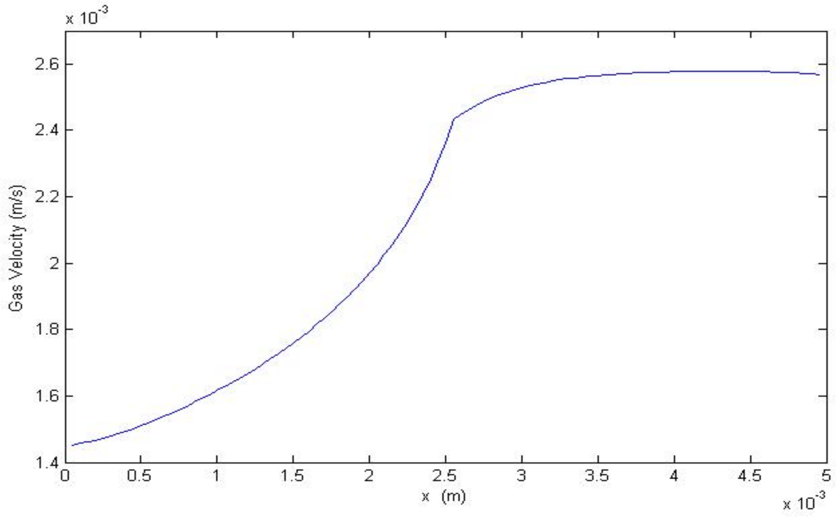

Figure 5.23: Velocity of gas in two layers of fabric

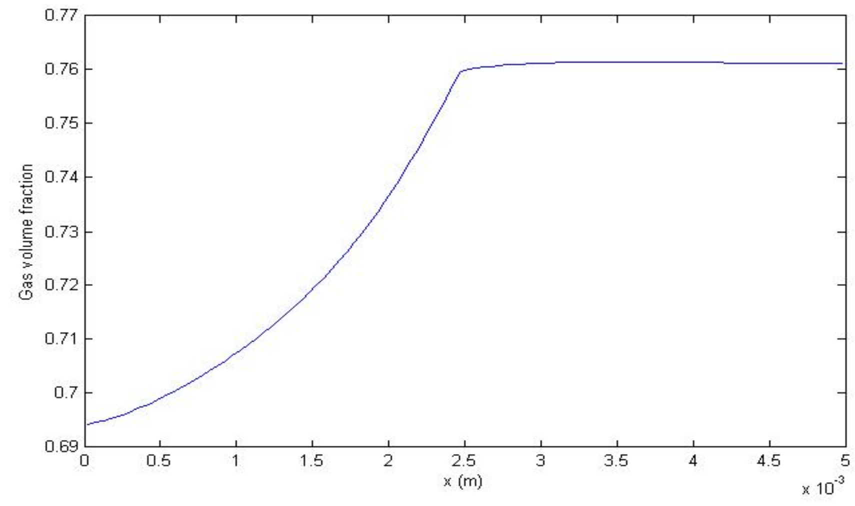

Figure 5.24: Volume fraction of gas in two layers of fabric

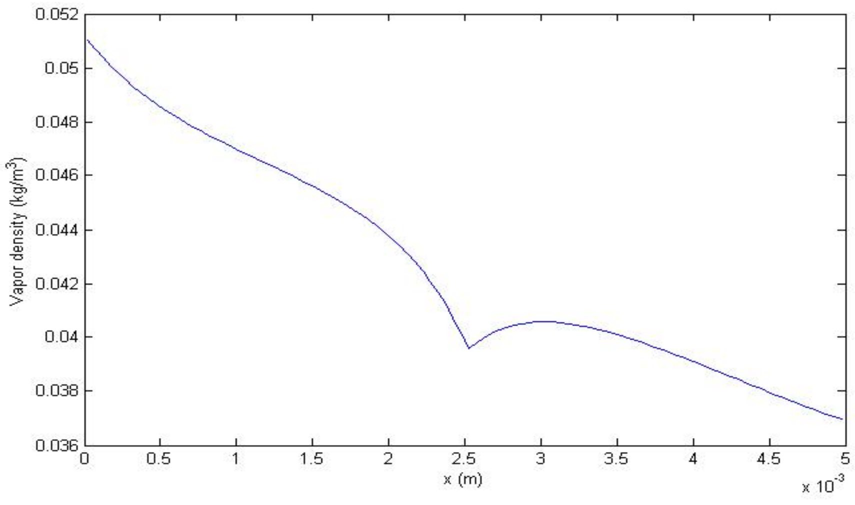

Figure 5.25: Concentration of vapour in two layers of fabric

Table 5.5: Normalised L^2-norm error

ncells	$\frac{\|\|p_{g,ref}-p_{g,h}\|\|}{\|\|p_{g,ref}\|\|}$	$\frac{\|\|\epsilon_{l,ref}-\epsilon_{l,h}\|\|}{\|\|\epsilon_{l,ref}\|\|}$	$\frac{\|\|H_{g,ref}-H_{g,h}\|\|}{\|\|H_{g,ref}\|\|}$	$\frac{\|\|H_{l,ref}-H_{l,h}\|\|}{\|\|H_{l,ref}\|\|}$	$\frac{\|\|C_{a,ref}-C_{a,h}\|\|}{\|\|C_{a,ref}\|\|}$
10	1.3e-1	1.6e-2	5.8e-3	3.2e-3	1.5e-2
20	6.2e-2	8.0e-3	2.8e-3	1.6e-3	8.1e-3
50	1.9e-2	2.5e-3	9.7e-4	5.4e-4	2.6e-3

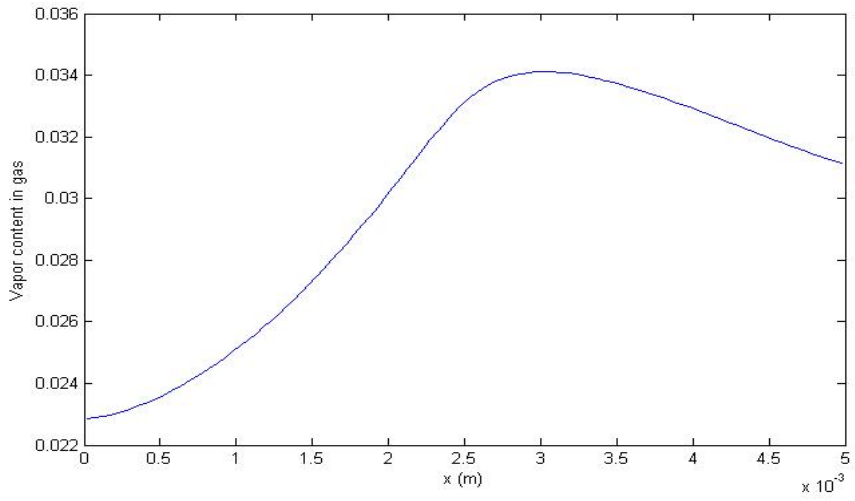

Figure 5.26: vapour content in gas in two layers of fabric

with the PHOENICS code. Two single layer problems and a two layers problem are covered here. The results obtained here become the reference solution in the next chapter where some defect correction schemes are applied to the same problems.

Chapter 6

Defect Correction Scheme in a CFD Code

The previous chapter shows some implementations of fluid flow and transport process in the PHOENICS code. In this chapter, a defect correction scheme is applied to the same problems, i.e. a single phase 2D fluid flow with heat transfer problem and a multiphase 1D fluid flow with heat & moisture transfer problem. In those problems, the defect correction scheme splits one domain computations with different properties into two subdomain computations with uniform properties. For the second problem, the computation in two fabric layers are decoupled into two independent individual computation, each with its own permeability property. This chapter also demonstrates how the scheme is implemented to couple the solutions from the PHOENICS code. Furthermore, the scheme is tested against its suitability for a problem represented by a system of partial differential equations rather than a single partial differential equation as found in Chapter-4.

The novel works towards the implementation of the scheme in this chapter are:

- the application of the defect correction scheme in a coupled system of partial differential equations
- the integration of the defect correction iterative procedure with the nonlinear corrective procedure of the interface solvers (Newton-GMRES, the Newton method with finite difference Jacobian, and a modified α-method) in PHOENICS modules.

6.1 PHOENICS setting

A decomposition of a domain in PHOENICS is not as straightforward as the decomposition of a multichip module demonstrated in Chapter-4. The cell arrangement in PHOENICS for solving a Navier Stokes equation is based on a staggered grid, i.e. velocity is solved and stored at cell faces whereas the pressure and other transported variables are solved and stored at the cell centre. In order to accommodate the defect correction scheme in PHOENICS, the decomposition of the domain must then be adjusted with such staggered grid configuration.

In the proposed defect correction scheme, the interface aligns with the nodes of the solved variables. In these two problems (single-phase and two-phase problems), the most suitable interface variables are scalar variables which are stored at the cell-centre of the staggered mesh. Velocities themselves, despite being stored at cell faces, can be automatically obtained once the pressure is solved due to the Darcy's Law. Therefore, in the case of uniform mesh where the reference solution of the corresponding problems have been obtained in Chapter-5, the most comfortable way to set the interface line is by drawing it through the cell centre.

6.2 2D Single Phase Test Problem

In Chapter-5, the mesh for the single phase problem has been arranged with some cell configurations of (41×8), (101×20),(201×40), and (401×80). Let N_X and N_Y be the number of cells in the x-direction and y-direction respectively, the defect correction scheme is carried out by first decomposing the entire domain Ω into two subdomains Ω_1 and Ω_2, each with a mesh configuration of $(\frac{N_X-1}{2} \times N_Y)$ cells. This is illustrated in Fig.6.1. Notice that, exactly in the middle, there is an interface layer with a mesh configuration of $(1 \times N_Y)$ where the interface line is drawn through the center of each cell.

The interface transport variables in this problem are the pressure p and enthalpy H. There are N_Y cells in the interface layer. Therefore in the discrete framework, each interface cell has two unknowns corresponding to the value of p and H. In total, there are $(2 \times N_Y)$ variables which need to be solved. The associated defect functions are chosen as the volume integration of the residual of governing equations (5.35) and (5.36) in each interface cell, i.e.

$$D_{mass} = \int_{V_{cell}} -\nabla.\mathbf{u}\ dV \tag{6.1}$$

$$D_H = \int_{V_{cell}} -\nabla.(\rho\mathbf{u}H) + \nabla.(k\nabla T)\ dV \tag{6.2}$$

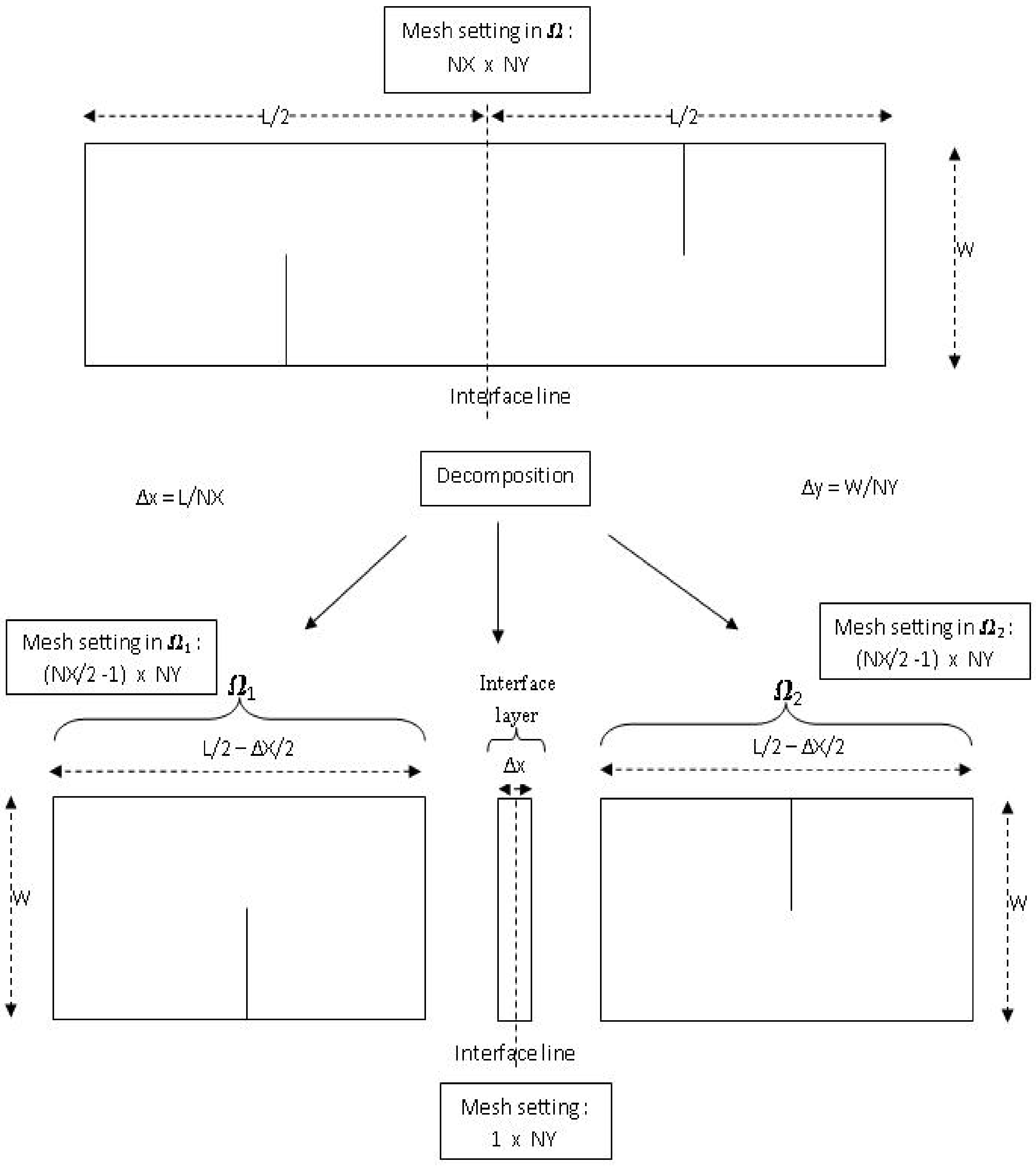

Figure 6.1: Domain and mesh decomposition

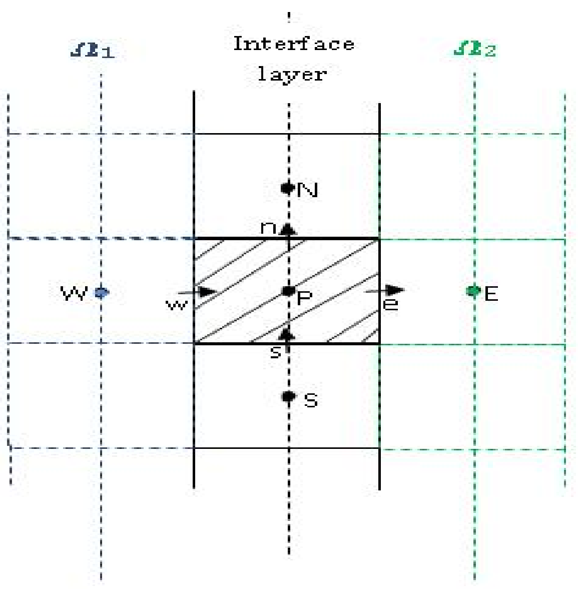

Figure 6.2: A cell in the interface layer

Since the fluid has a constant density, the first defect function correspond to the mass residual in each interface cell. The second function can be easily identified as the enthalpy residual. Using upwinding technique, the volume integration over the interface cell in Fig.6.2 can be approximated by

$$D_{mass} = u_w \Delta y - u_e \Delta y + v_s \Delta x - v_n \Delta x \tag{6.3}$$

$$\begin{aligned} D_H &= \rho\, u_w H_W \Delta y - \rho\, u_e H_P \Delta y + \rho\, v_s H_S \Delta x - \rho\, v_n H_P \Delta x \\ &\quad + \frac{k}{c_p}\left[\frac{H_W - H_P}{\Delta x}\Delta y - \frac{H_P - H_E}{\Delta x}\Delta y + \frac{H_S - H_P}{\Delta y}\Delta x - \frac{H_P - H_N}{\Delta y}\Delta x\right] \end{aligned} \tag{6.4}$$

At the right hand side of both equations, apart from the specific heat c_p, the lower-case subscripts correspond to the values at face centers of the interface cell whereas the upper-case subscripts correspond to those at the center of the cell. By means of the Darcy's equation (5.34), the velocity components in (6.3) and (6.4) can be approximated with the central difference scheme:

$$u_e = \frac{K_{x_e}}{\mu}\frac{p_P - p_E}{\Delta x} \tag{6.5}$$

$$u_w = \frac{K_{x_w}}{\mu}\frac{p_W - p_P}{\Delta x} \tag{6.6}$$

$$v_s = \frac{K_{y_s}}{\mu}\frac{p_S - p_P}{\Delta y} \tag{6.7}$$

$$v_n = \frac{K_{y_n}}{\mu}\frac{p_P - p_N}{\Delta y} \tag{6.8}$$

where the terms $K_{x_e}, K_{x_w}, K_{y_s}$, and K_{y_n} are the permeability K_x and K_y at the corresponding faces of the interface cell. Substituting these terms in (6.3) and (6.4), the equations $D_{mass} = 0$ and $D_H = 0$ form two coupled defect equations which are nonlinear in terms of pressure and enthalpy. Note that the variables which are not associated with the interface layer (p_W, p_E, H_W, H_E) are also functions of the interface variables $(p_P, p_N, p_S, H_P, H_N, H_S)$ due to subdomain computations in Ω_1 and Ω_2.

Now let the interface cell in Fig.6.2 be the i-th interface cell, and denote D_{mass} and D_H in the cell by F_i and $F_{(i+N_Y)}$ respectively, and the corresponding interface variables p_P and H_P at the cell centre by x_i and $x_{(i+N_Y)}$. Let also the neighbouring south and north cells be the $(i-1)$th and $(i+1)$th interface respectively. Then all discrete defect equations in the interface layer can be assembled in a system of nonlinear equations

$$F(x) = 0$$

where $F : R^{(2\times N_Y)} \rightarrow R^{(2\times N_Y)}$ and $x \in R^{(2\times N_Y)}$.

In order to solve these nonlinear equations, the defect correction scheme is applied using the Newton method with a finite difference Jacobian. The Jacobian approximation is computed with a finite difference scheme by using a small perturbation of $\delta H = 100$ kJ and $\delta p = 0.01$ Pa for pressure and enthalpy respectively. These parameters are chosen after a number of numerical experiments and they give better results than those by the scaling method (2.22) suggested in [66].

The aim of the implementation of this defect correction scheme is to test the ability of the scheme in coupling PHOENICS' subdomain computations in order to recover the conventional PHOENICS' numerical result. The word 'conventional' here refers to the PHOENICS' numerical procedures when solving the whole domain problem (without decomposition). Table-6.1 demonstrates the difference between the solutions by defect correction scheme and those by the conventional solution. The pressure p_h and enthalpy H_h are the conventional solution, whereas $p_{h,DD}$ and $H_{h,DD}$ are the solutions obtained by the defect correction scheme. When the mesh is refined, the difference of pressure is getting smaller, yet the enthalpy does not follow the same pattern. The performance of the Newton interface solver is also

Table 6.1: Error with respect to the conventional solution

ncells	$\|\|p_h - p_{h,DD}\|\|_\infty$	$\frac{\|\|p_h - p_{h,DD}\|\|}{\|\|p_h\|\|}$	$\|\|H_h - H_{h,DD}\|\|_\infty$	$\frac{\|\|H_h - H_{h,DD}\|\|}{\|\|H_h\|\|}$
101 x 20	0.1981	1.2e-2	953.4	7.9e-4
201 x 40	0.0995	6.5e-3	1805.6	8.6e-4
401 x 80	0.0499	3.3e-3	3121.0	7.1e-4

examined. It is measured by the number of nonlinear iteration or the number of domain decomposition iteration needed until the normalized residual of mass $\overline{D}_{mass}$ and enthalpy $\overline{D}_H$ in each interface cell are both lower than 10^{-3}. This parameter is also used by PHOENICS as the default stopping criteria of its entire solution procedure. With regards to (6.3) and (6.4), the normalised residual of mass and enthalpy are given by

$$\overline{D}_{mass} = \frac{D_{mass}}{\Sigma_{mass}} \tag{6.9}$$

$$\overline{D}_H = \frac{D_H}{\Sigma_H} \tag{6.10}$$

where Σ_{mass} and Σ_H are the sum of absolute incoming and outcoming flux:

$$\Sigma_{mass} = |u_w \Delta y| + |u_e \Delta y| + |v_s \Delta x| + |v_n \Delta x| \tag{6.11}$$

$$\begin{aligned}\Sigma_H &= |u_w H_W \Delta y| + |\rho\, u_e H_P \Delta y| + |\rho\, v_s H_S \Delta x| + |\rho\, v_n H_P \Delta x| \\ &+ \frac{k}{c_p}\left[\left|\frac{H_W - H_P}{\Delta x}\Delta y\right| + \left|\frac{H_P - H_E}{\Delta x}\Delta y\right| + \left|\frac{H_S - H_P}{\Delta y}\Delta x\right| + \left|\frac{H_P - H_N}{\Delta y}\Delta x\right|\right]\end{aligned} \tag{6.12}$$

The initial guess for interface data is set zero for both pressure and enthalpy in each interface cell. Table-6.2 displays the performance of the scheme for each mesh configuration. Note that due to the finite difference Jacobian, this scheme needs $(2 \times N_Y + 1)$ domain decomposition iterations for for one Newton (nonlinear) iteration. The table shows an increase of nonlinear iteration when the mesh is refined from $N_Y = 40$ to $N_Y = 80$. The defect correction scheme is also implemented with the Newton-GMRES method as the interface solver. Although the residual can be reduced, the solver works very slow. For the mesh configuration (101×20),

Table 6.2: Performance of Newton-finite difference

ncells	NL Iteration	DD Iteration
101 x 20	3	123
201 x 40	3	243
401 x 80	4	644

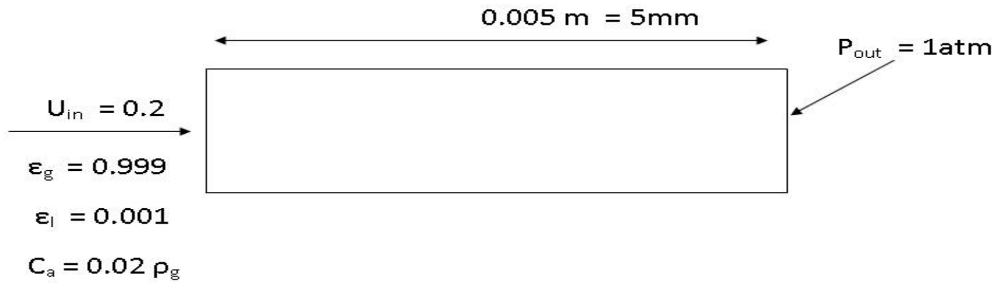

Figure 6.3: Domain of fabric

residuals in some cells are still larger than 10^{-1} after 750 domain decomposition iterations. Various ways have been tried to modify some GMRES parameters, yet it does not show much improvement of the solver's performance.

6.3 Multiphase problem

Just like in the previous single-phase problem, the decomposition of fabric domain in the multiphase problem must also be adjusted with the employed staggered mesh. A simple example of mesh of the domain in Fig.6.3 is illustrated in Fig.6.4 where a uniform grid is employed in x-direction. Note that the vertical width in both figures is imaginary since the problem is 1D in horizontal direction. Rectangular cells are drawn instead of line cells in order to appreciate cell volumes and face areas with regards to volume integration.

Before determining the position of interface, a selection of interface variables whose value will be fixed needs to be done. In PHOENICS, velocity is the only basic variable stored at the cell face. The setting of nonoverlapping decomposition can be simplified with the fact that Darcy's law governs the momentum equation. Velocity is proportional to pressure gradient, therefore the storing of velocity is not needed at the interface since its value near interface can be determined by the interface pressure.

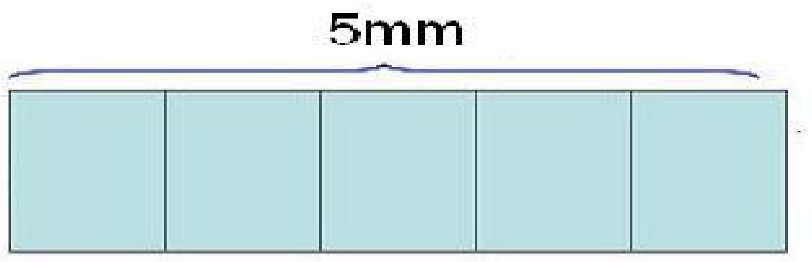

Figure 6.4: A mesh of one single domain in PHOENICS

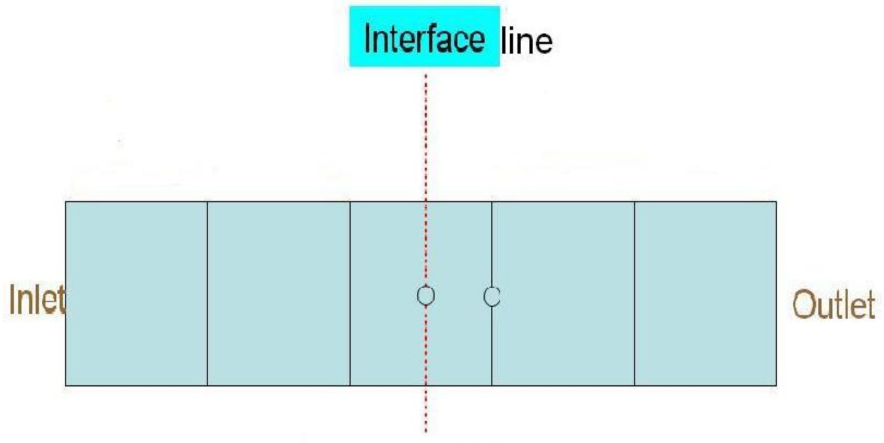

Figure 6.5: Interface line

Now, with all variables to be fixed at the interface having a cell-centred configuration, the interface line needs to be defined at one of the cell centre between two subdomains, such as shown in Fig.6.5. Furthermore, with the PHOENICS requirement that the boundaries of domain/subdomain are some cell faces instead of cell centres, the cell at which centre the interface is set must disappear from the domain subdivision. Fig.6.5 shows the illustration of a subdivision with its interface cell.

In the interface cell, the solved variables pressure p_g, volume fractions ϵ_g, ϵ_l, entalphies H_g, H_l and vapour concentration C_a are fixed. These are associated with the five governing equations : mass and energy of both phases and the transport of vapour. Consequently temperatures T_g, T_l and the gas density $_g$ are also fixed there because their values can be inferred from those main interface variables.

The decomposition into two subdomains requires the setting adjustment of boundary condition for each subdomain. The left boundary condition of the first subdomain and

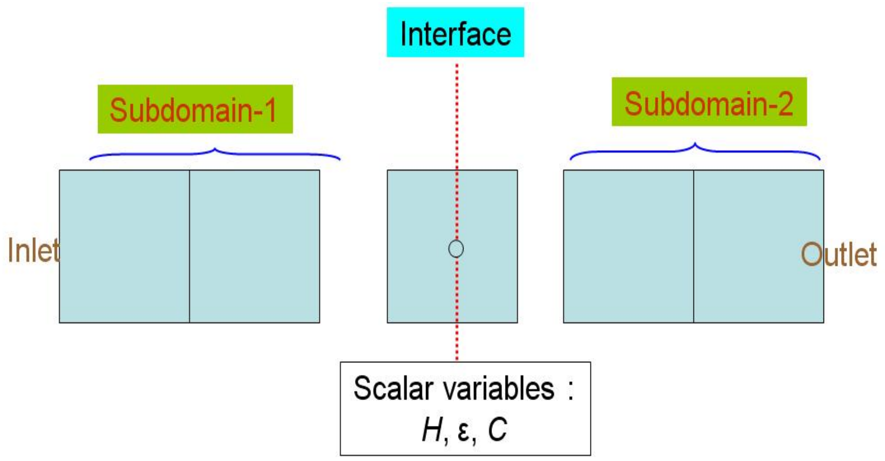

Figure 6.6: Decomposition in multiphase process

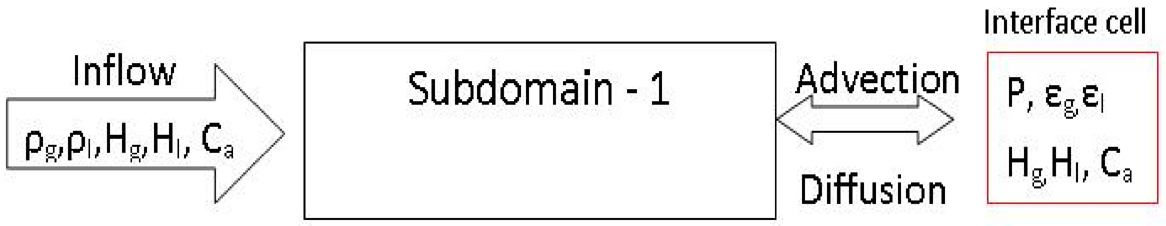

Figure 6.7: Boundary condition of Subdomain-1

the right boundary condition of the second subdomain are the same as those given for the single domain computation. The boundary conditions at the right boundary face of the first subdomain need to be adjusted such that the flow of each transported variable crossing the interface boundary are governed by the advection and diffusion terms of its corresponding governing equation in balance with the interface cell values. The same treatments are given to the left boundary of the second subdomain. These are illustrated in Fig.6.7 and Fig.6.8.

In the current problem, the scheme must acommodate the five different interface variables. Interest is again given to a scheme which evades approximations of full Jacobian matrices because they will need many domain decomposition iterations. Instead of using the adaptive-α method which only employ a scalar α for the whole system of equations, in this current problem a set of scalars α_i are used to represent each interface variable. This will be called as the modified α method.

Given that there are five interface variables to solve with five corresponding governing

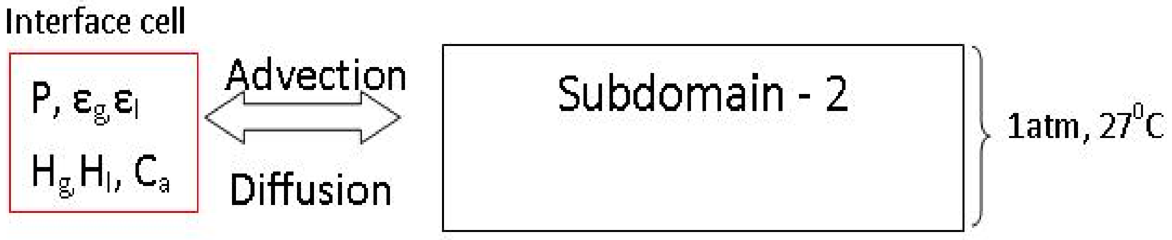

Figure 6.8: Boundary condition of Subdomain-2

equations, the set of five scalars are needed because the range of value of each variable differs significantly. While the pressure can be in the scale of 10^5 Pa, the volume fraction is measured in the unit of percent. For comparisons, some numerical results from another defect correction scheme are also demonstrated in the following sections. It uses the Newton method with finite difference Jacobian approximation as the nonlinear solver.

For the choice of defect equations, the volume intergration of residuals of governing equations will become the criteria of the accuracy of the solution. Therefore there are five defect functions associated with this steady state process :

- for total mass :

$$D_g = \int_V \left(-\frac{\partial(\epsilon_g V_g C_g)}{\partial x} - \frac{\partial(\epsilon_l V_l \rho_l)}{\partial x} \right) \, dV \tag{6.13}$$

- for liquid mass :

$$D_l = \int_V \left(-\frac{\partial(\epsilon_l V_l \rho_l)}{\partial x} - Q_p \right) \, dV \tag{6.14}$$

- for vapour mass :

$$D_a = \int_V \left(\frac{\partial}{\partial x} \left[(D_M)\epsilon_g \frac{\partial(C_a)}{\partial x} \right] - \frac{\partial(\epsilon_g V_g C_a)}{\partial x} + Q_p \right) \, dV \tag{6.15}$$

- for gas enthalpy :

$$D_{H_g} = \int_V \left(-\frac{\partial(\epsilon_g C_g V_g H_g)}{\partial x} + \frac{\partial}{\partial x} \left[\epsilon_g k_g \frac{\partial(T_g)}{\partial x} \right] + \Delta H_g(Q_p) \right) \, dV \tag{6.16}$$

- for liquid water enthalpy :

$$D_{H_l} = \int_V \left(-\frac{\partial(\epsilon_l \rho_l V_l H_l)}{\partial x} + \frac{\partial}{\partial x} \left[\epsilon_l k_l \frac{\partial(T_l)}{\partial x} \right] + \Delta H_l(Q_p) \right) \, dV \tag{6.17}$$

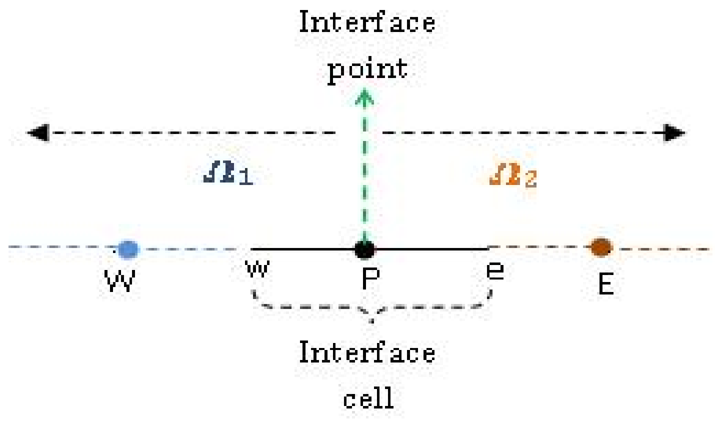

Figure 6.9: Interface Cell

All the notations in above equations have the same meaning as the same notations presented in the previous chapter. The nonlinear discrete approximations of (6.13 -6.17) are performed with the same technique as used in (6.3 - 6.8). Using an upwinding technique, the volume integration over the interface cell in Fig.6.9 can be approximated by

$$D_g = \epsilon_{g,W} V_{g,w} C_{g,W} - \epsilon_{g,P} V_{g,e} C_{g,P} + \epsilon_{l,W} V_{l,w} \rho_l - \epsilon_{l,P} V_{l,e} \rho_l \tag{6.18}$$

$$D_l = \epsilon_{l,W} V_{l,w} \rho_l - \epsilon_{l,P} V_{l,e} \rho_l - Q_{p,P} \Delta x \tag{6.19}$$

$$\begin{aligned} D_{H_g} =& \epsilon_{g,W} V_{g,w} C_{g,W} H_{g,W} - \epsilon_{g,P} V_{g,e} C_{g,P} H_{g,P} + \Delta H_g(Q_{p,P}) \Delta x \\ &+ k_g \left[\frac{\epsilon_{g,E} + \epsilon_{g,P}}{2} \frac{T_{g,E} - T_{g,P}}{\Delta x} - \frac{\epsilon_{g,W} + \epsilon_{g,P}}{2} \frac{T_{g,P} - T_{g,W}}{\Delta x} \right] \end{aligned} \tag{6.20}$$

$$\begin{aligned} D_{H_l} =& \epsilon_{l,W} V_{l,w} \rho_l H_{l,W} - \epsilon_{l,P} V_{l,e} \rho_l H_{l,P} + \Delta H_l(Q_{p,P}) \Delta x \\ &+ k_l \left[\frac{\epsilon_{l,E} + \epsilon_{l,P}}{2} \frac{T_{l,E} - T_{l,P}}{\Delta x} - \frac{\epsilon_{l,W} + \epsilon_{l,P}}{2} \frac{T_{l,P} - T_{l,W}}{\Delta x} \right] \end{aligned} \tag{6.21}$$

$$\begin{aligned} D_a =& \epsilon_{g,W} V_{g,w} C_{a,W} - \epsilon_{g,P} V_{g,e} C_{a,P} + Q_{p,P} \Delta x \\ &+ D_M \left[\frac{\epsilon_{g,E} + \epsilon_{g,P}}{2} \frac{C_{a,E} - C_{a,P}}{\Delta x} - \frac{\epsilon_{g,W} + \epsilon_{g,P}}{2} \frac{C_{a,P} - C_{a,W}}{\Delta x} \right] \end{aligned} \tag{6.22}$$

Some of the variables above has two indices in the subscripts. The first index indicates the phase of the fluid and the second index correspond to the location at the cell. The lower-case in the second index corresponds to a face center of an interface cell whereas the upper-case corresponds to a cell centre. For example, $V_{l,e}$ is the liquid velocity at the face e while $C_{g,P}$ denotes the gas concentration at the cell centre P.

By means of the Darcy's equations (5.40) and (5.41), the gas and liquid velocity components can be approximated with the central difference scheme:

$$V_{g,w} = \frac{\epsilon_{g,W}}{\epsilon^2 \mu_g} K \left[\frac{p_{g,W} - p_{g,P}}{\Delta x} \right] \tag{6.23}$$

$$V_{g,e} = \frac{\epsilon_{g,P}}{\epsilon^2 \mu_g} K \left[\frac{p_{g,P} - p_{g,E}}{\Delta x} \right] \tag{6.24}$$

$$V_{l,w} = \frac{\epsilon_{l,W}}{\epsilon^2 \mu_l} K \left[\frac{p_{g,W} - p_{g,P}}{\Delta x} - \frac{2\sigma\epsilon cos\Theta}{d_c \Delta x} (\frac{1}{\epsilon_{l,W}} - \frac{1}{\epsilon_{l,P}}) \right] \tag{6.25}$$

$$V_{l,e} = \frac{\epsilon_{l,P}}{\epsilon^2 \mu_l} K \left[\frac{p_{g,P} - p_{g,E}}{\Delta x} - \frac{2\sigma\epsilon cos\Theta}{d_c \Delta x} (\frac{1}{\epsilon_{l,P}} - \frac{1}{\epsilon_{l,E}}) \right] \tag{6.26}$$

The evaporation rate at the cell centre P can be approximated using (5.50) :

$$Q_{p,P} = \epsilon_{g,P} h_{l\leftrightarrow g} S_v (C_a^*(T_{g,P}) - C_{a,P})$$

where the saturated concentration C_a^* is given by the function (5.51).

By means of the ideal gas law, the relation $\epsilon_g = 1 - \epsilon_l$, and the substitution of all the velocity terms and the evaporation rate above in (6.18)- (6.22), the zero defects are nonlinear in terms of the pressure p_g, the liquid volume fraction ϵ_l, the enthalpies H_g, H_l and the vapour concentration C_a. Note also that the variables which are not associated with the interface point P are functions of the interface variables $(p_{g,P}; \epsilon_{l,P}; H_{g,P}; H_{l,P}; C_{a,P})$ due to the subdomain computations in Ω_1 and Ω_2. In order to simplify the notations of the interface variables, the second index P is replaced by the subscript $*$. As an example, p_{g*} will be used instead of $p_{g,P}$.

Then a system of five nonlinear defect equations

$$F(x) = 0$$

can be established by setting $x = [p_{g*}, \epsilon_{l*}, H_{g*}, H_{l*}, C_{a*}]^T$ and $F = [D_g, D_l, D_{H_g}, D_{H_l}, D_a]^T$. The Newton solver is implemented in the defect correction scheme where the Jacobian approximation is obtained using the small perturbations:

$$\begin{aligned}
\delta x_1 = &\quad \delta p_{g*} &= 10\,\mathrm{Pa} \\
\delta x_2 = &\quad \delta \epsilon_{l*} &= 0.001 \\
\delta x_3 = &\quad \delta H_{g*} &= 100\,\mathrm{kJ} \\
\delta x_4 = &\quad \delta H_{l*} &= 100\,\mathrm{kJ} \\
\delta x_5 = &\quad \delta C_{a*} &= 0.001\,\mathrm{kg/m^3}
\end{aligned}$$

This set of values is chosen as the optimal values after some numerical experiments.

On the other hand, for the setting of the modified α-method as the interface solver, the defect functions D_g, D_l, D_a, D_{H_g}, and D_{H_l} will correspond to the set of $\alpha_g, \alpha_l, \alpha_a, \alpha_{H_g}$, and α_{H_l} respectively and also to the interface variables $p_{g*}, \epsilon_{l*}, C_{a*}, H_{g*}$, and H_{l*} respectively. The update of interface then satisfies the following iteration :

- $p_{g*}^{n+1} = p_{g*}^n - \alpha_g^n(D_g^n)$
- $\epsilon_l^{n+1} = \epsilon_{l*}^n - \alpha_l^n D_l^n$
- $C_{a*}^{n+1} = C_{a*}^n - \alpha_a^n D_a^n$
- $H_{g*}^{n+1} = H_{g*}^n - \alpha_{H_g}^n D_{H_g}^n$
- $H_{l*}^{n+1} = H_{l*}^n - \alpha_{H_l}^n D_{H_l}^n$

Since the process is in 1D while the decomposition has only one interface, the discretisation will lead to one grid for each variable. Hence the α updating is simply :

- $\alpha_g^{n+1} = \alpha_g^n \frac{\left|D_g^{n+1} - D_g^n\right|}{\left|D_g^n\right|}$
- $\alpha_l^{n+1} = \alpha_l^n \frac{\left|D_l^{n+1} - D_l^n\right|}{\left|D_l^n\right|}$
- $\alpha_a^{n+1} = \alpha_a^n \frac{\left|D_a^{n+1} - D_a^n\right|}{\left|D_a^n\right|}$
- $\alpha_{H_g}^{n+1} = \alpha_{H_g}^n \frac{\left|D_{H_g}^{n+1} - D_{H_g}^n\right|}{\left|D_{H_g}^n\right|}$
- $\alpha_{H_l}^{n+1} = \alpha_{H_l}^n \frac{\left|D_{H_l}^{n+1} - D_{H_l}^n\right|}{\left|D_{H_l}^n\right|}$

The next section presents the numerical results of these two schemes. The problems which are tested on this framework are the same as the ones solved by the conventional scheme shown in the previous chapter. Moreover, the solutions obtained from the conventional scheme become the reference solutions for comparison with domain decomposition results.

6.3.1 A single layer of fabric

The first problem is a uniform layer of fabric characterized by its effective radius of pore, $d_c = 5.10^7 m$, which leads to intrinsical permeability $K \approx 10^{-15}$. While the reference solution is solved on a mesh of 100 equidistant cells, the decomposition into two subdomains is carried out by allocating 50 cells for the subdomain-1 and 49 cells for the subdomain-2. The one extra cell acts as the interface cell.

The reference solution in the interface cell position is given by :

- $p_{g*} = 6.517101\ 10^4$ Pa + 1 atm
- $\epsilon_{l*} = 0.2921035$
- $H_{g*} = 2.812401\ 10^4$ kJ
- $H_{l*} = 1.103564\ 10^5$ kJ
- $C_{a*} = 0.02242636\ C_{g*}$

which is obtained from the profile of solution in previous chapter. The density of gas follows from the ideal gas law $C_{g*} = \dfrac{P_{g*}}{RT_{g*}}$ and $T_{g*} = H_{g*}/c_{p_g}$.

Newton method

This section gives numerical results of the defect correction scheme which use a Newton method as its nonlinear solver whereas the Jacobian matrix is approximated with Finite Difference technique. Despite the Jacobian approximation, the implementation of the scheme in this specific problem is not very heavy because only 5 interface variables are involved. Therefore, one Newton iteration (or one nonlinear iteration) needs 6 domain decomposition iteration, where 5 iterations are the small perturbation steps needed to compute the finite difference approximation of Jacobian matrix, and the other one is the the Newton updating step. The initial interface data for this test are set as follows :

- $p^0_{g*} = 1$ atm
- $\epsilon^0_{l*} = 0.9999$
- $H^0_{g*} = 5.0\ 10^4$ kJ
- $H^0_{l*} = 2.00\ 10^5$ kJ
- $C^0_{a*} = C^0_{g*}$

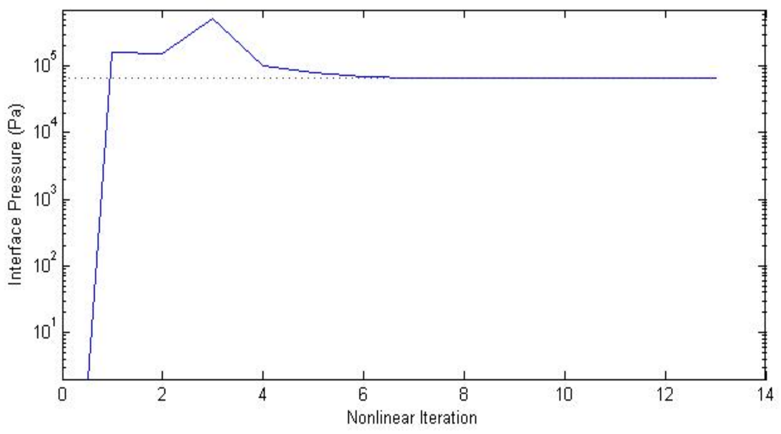

Figure 6.10: Iteration of interface pressure - single layer problem (Newton method)

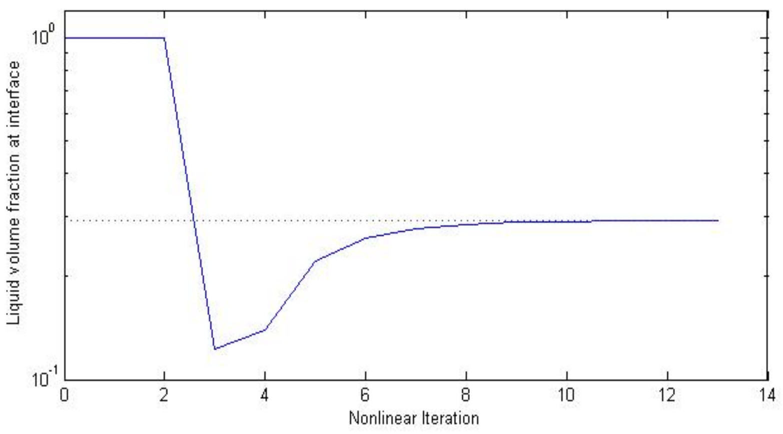

Figure 6.11: Iteration of interface liquid volume fraction - single layer problem (Newton method)

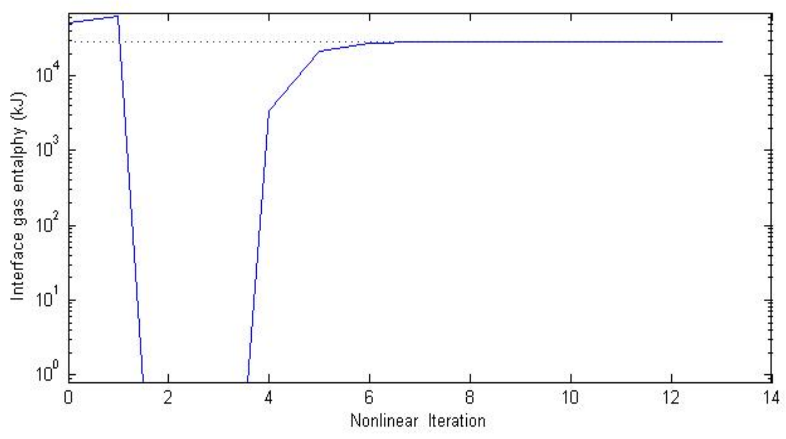

Figure 6.12: Iteration of interface gas enthalpy - single layer problem (Newton method)

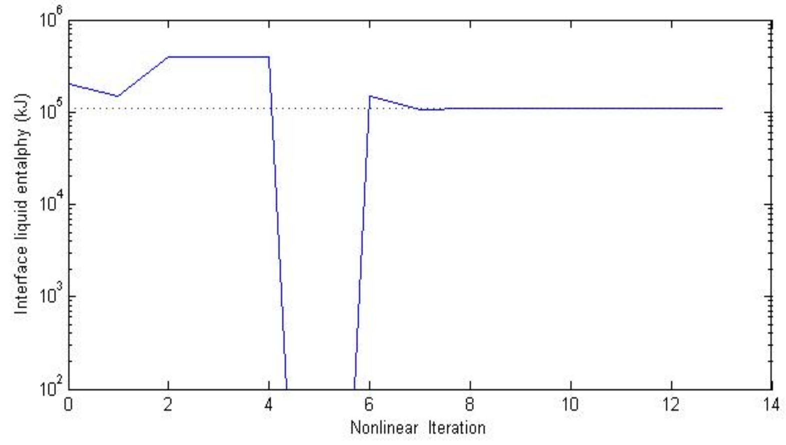

Figure 6.13: Iteration of interface liquid enthalpy - single layer problem (Newton method)

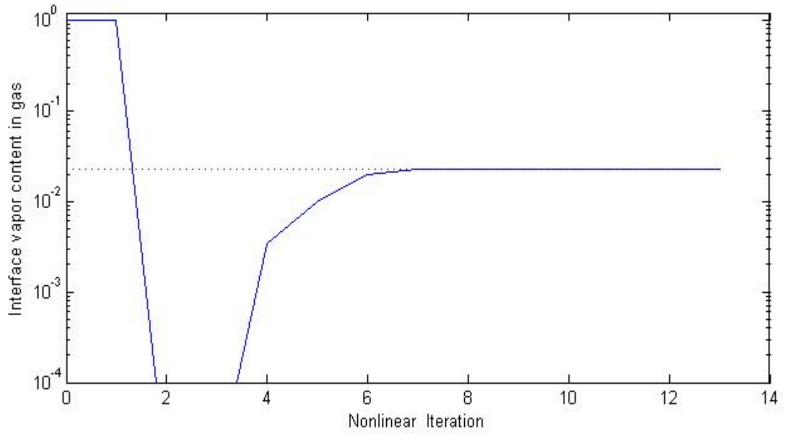

Figure 6.14: Iteration of interface vapor content in gas - single layer problem (Newton method)

where the initial gas density C^0_{g*} folows from the ideal gas law using p^0_{g*} and H^0_{g*}.

Fig.6.10-6.14 demonstrate the nonlinear iterations of each interface variable. The reference solution is displayed as a straight dotted line as an indicator how well the convergence of iterates take place. Recall that each nonlinear iteration needs 5 domain decomposition iterations while each domain decomposition iteration needs subdomain solves of the entire model.

Fig.6.15-6.19 show the residual of total mass, liquid mass, gas enthalpy, liquid enthalpy, and vapour mass which correspond to the iterations of interface variables in Fig.6.10-6.14. The residuals are normalized by the sum of absolute incoming and outcoming flux of the transport variable, just like those defined in equations (6.9) - (6.12).

Fig.6.10-6.14 demonstrate the convergence of all interface variables to a set of solutions very close to the reference. These are confirmed by the decreasing normalized residuals over the Newton (or nonlinear) iteration in Fig.6.15-6.19 where each residual amounts to less than 10^{-3} at latest iterations.

Given the satisfying result of the interface variables and residuals, it is of iterest to compare the entire profile of each variable between the domain decomposition and the reference solution. The subdomain solutions and the interface variable are then merged and the profile comparisons of gas pressure, liquid volume fraction, gas and liquid enthalpy, and vapour content are illustrated in Fig.6.20, 6.22, 6.24, 6.26, and 6.28 respectively. The reference solutions are presented by dash-dotted lines whereas the domain decomposition solutions

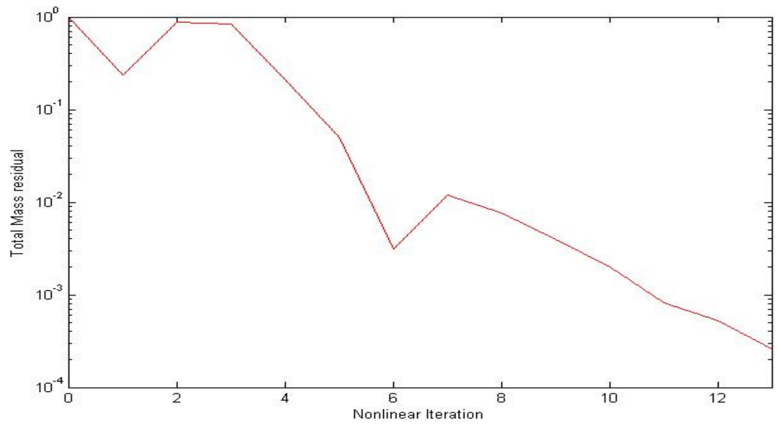

Figure 6.15: Normalized total mass residual at interface - single layer problem (Newton method)

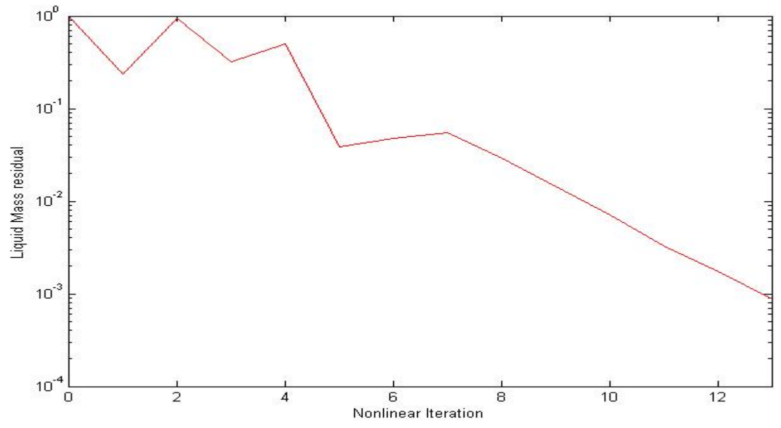

Figure 6.16: Normalized liquid mass residual at interface - single layer problem (Newton method)

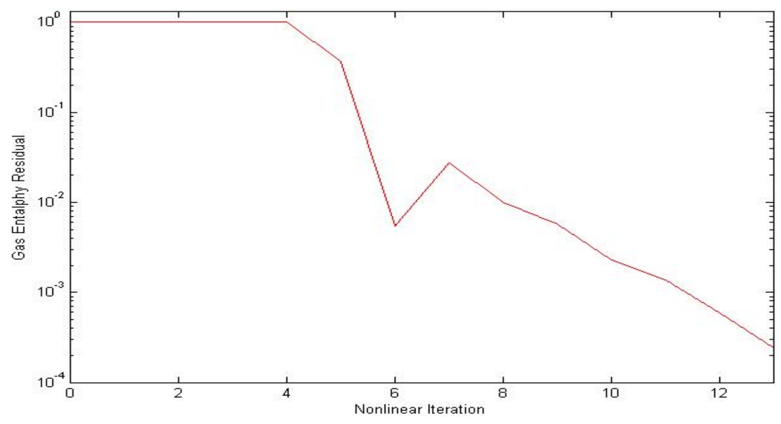

Figure 6.17: Normalized gas enthalpy residual at interface - single layer problem (Newton method)

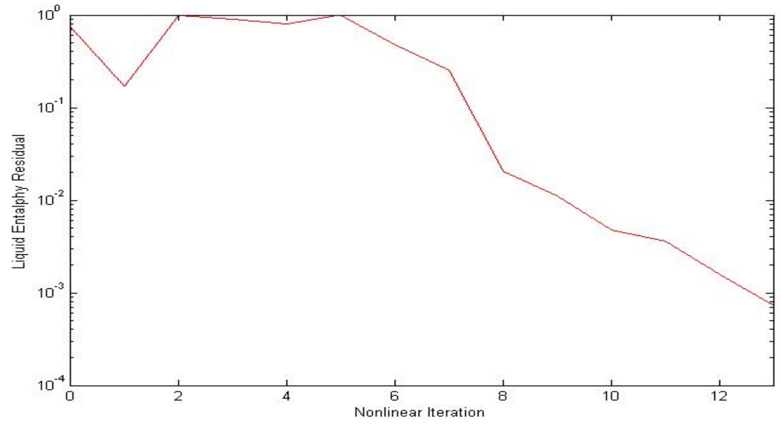

Figure 6.18: Normalized liquid enthalpy residual at interface - single layer problem (Newton method)

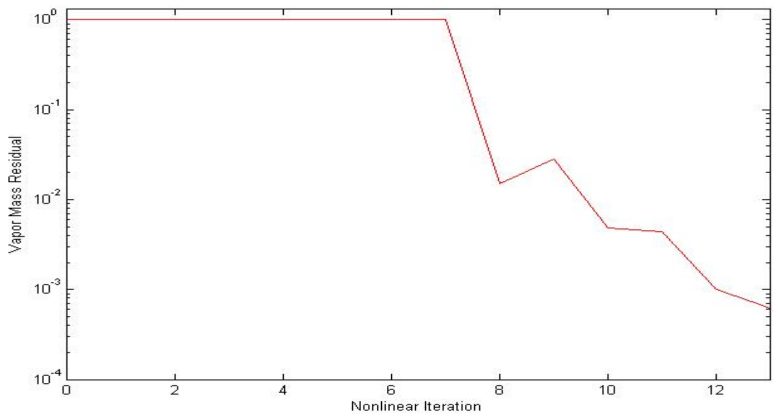

Figure 6.19: Normalized vapor mass residual at interface - single layer problem (Newton method)

by the lines with plus marker. They agree with the reference solution along the thickness quite well. For each variable, the profile of domain decompostion solution agrees with the reference solution to certain discrepancies which are measured in in Fig. 6.21, 6.23, 6.25, 6.27, and 6.29. Each plot simply represents the quantity $X - X^{DD}$ for each variable along the domain thickness where X is reference solution and X^{DD} is the domain decomposition solution. Compared to the range of values in the profile of each variable, these last 5 plots show minor discrepancies between the two solutions.

At the first Newton iteration, when the initial interface condition is far from the reference solution, the computed Jacobian approximation is

$$\begin{bmatrix} 4.106\,10^{-8} & 1.248\,10^{-2} & 1.540\,10^{-8} & 0.0000 & 1.474\,10^{-1} \\ -4.174\,10^{-8} & -1.619\,10^{-2} & 7.220\,10^{-7} & 0.0000 & 1.604\,10^{-1} \\ 1.075\,10^{-2} & 8.496\,10^{+2} & -9.989\,10^{-2} & 0.0000 & -1.538\,10^{+3} \\ -1.111\,10^{-1} & -1.123\,10^{+3} & 2.023\,10^{-1} & 0.0371 & 6.367\,10^{+4} \\ 8.281\,10^{-8} & 2.868\,10^{-2} & -7.066\,10^{-7} & 0.0000 & 3.096\,10^{-1} \end{bmatrix}$$

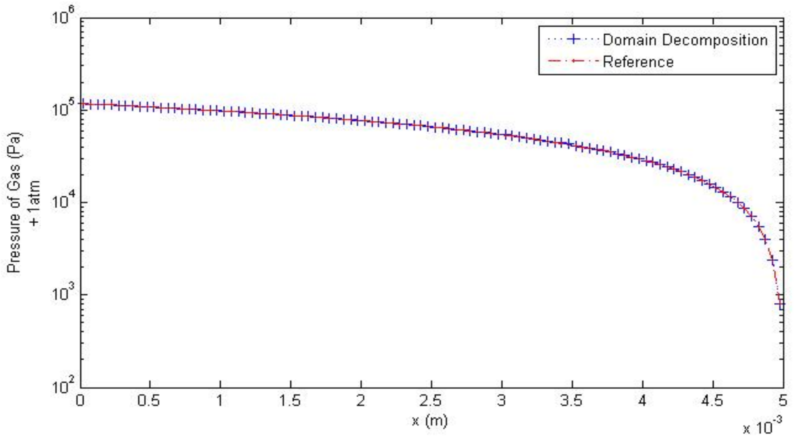

Figure 6.20: Comparison of pressure between domain decomposition and reference solution

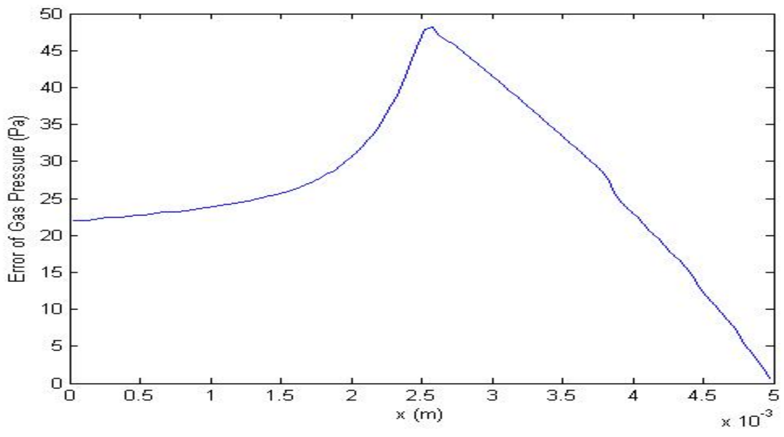

Figure 6.21: Difference of pressure between domain decomposition and reference solution

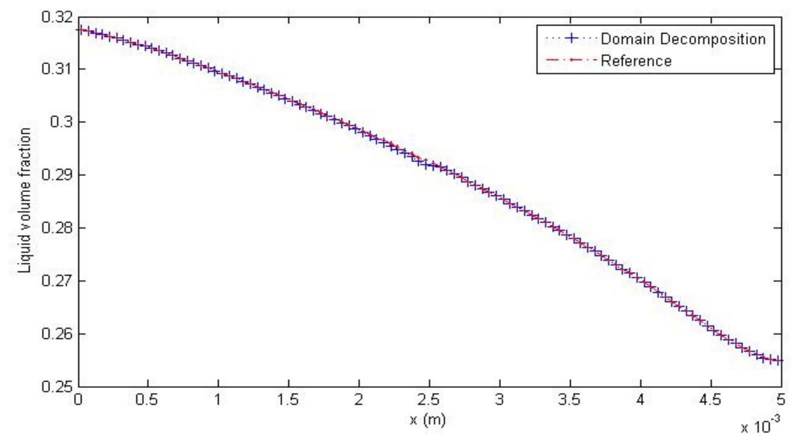

Figure 6.22: Comparison of liquid volume fraction between domain decomposition and reference solution

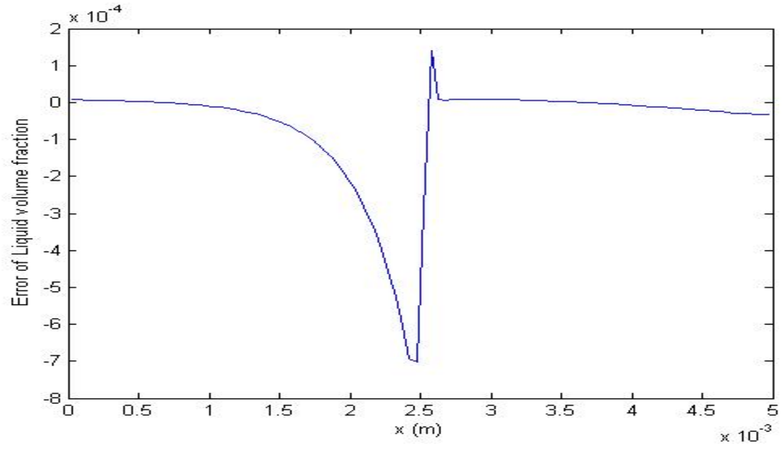

Figure 6.23: Difference of liquid volume fraction between domain decomposition and reference solution

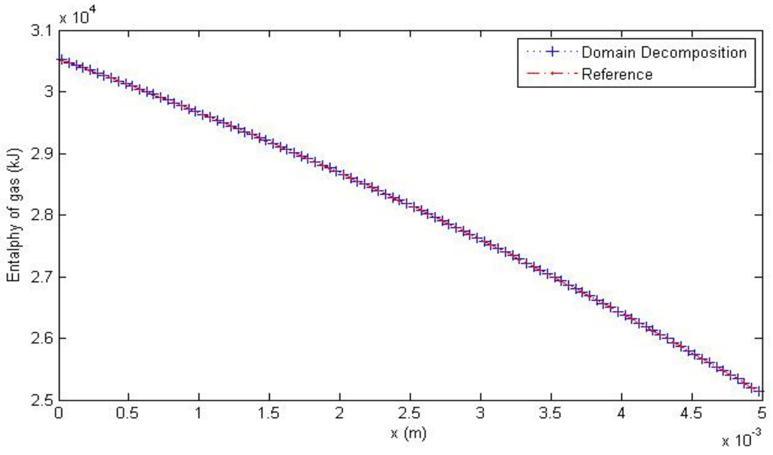

Figure 6.24: Comparison of gas enthalpy between domain decomposition and reference solution

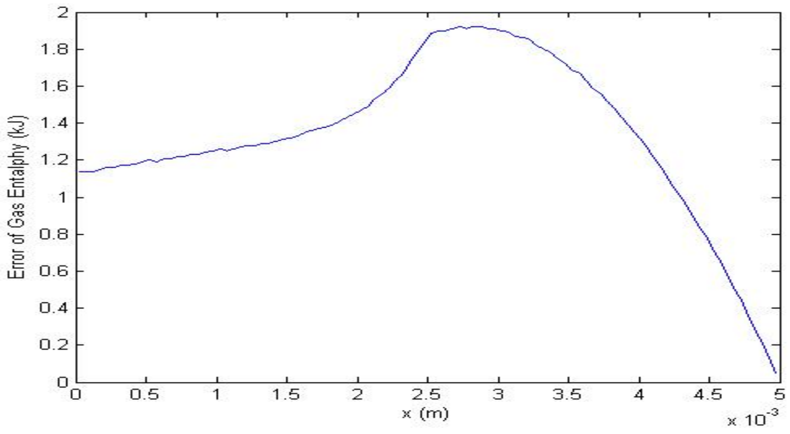

Figure 6.25: Difference of gas enthalpy between domain decomposition and reference solution

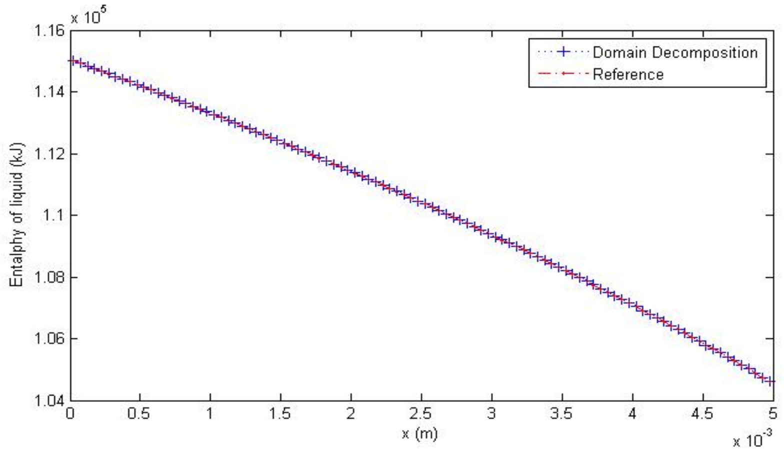

Figure 6.26: Comparison of liquid enthalpy between domain decomposition and reference solution

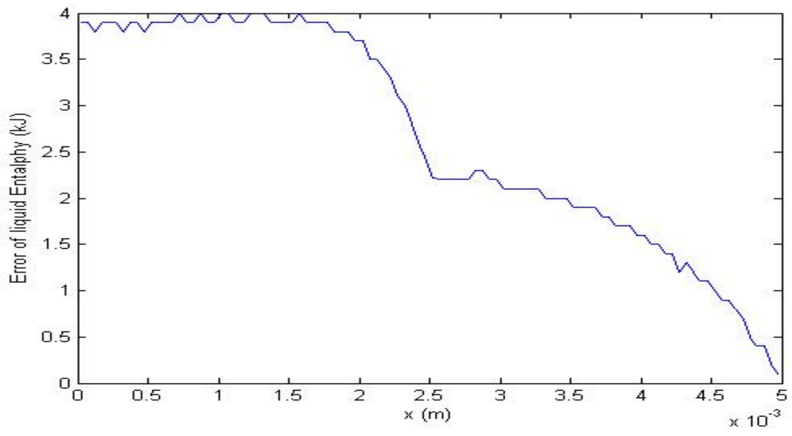

Figure 6.27: Difference of liquid enthalpy between domain decomposition and reference solution

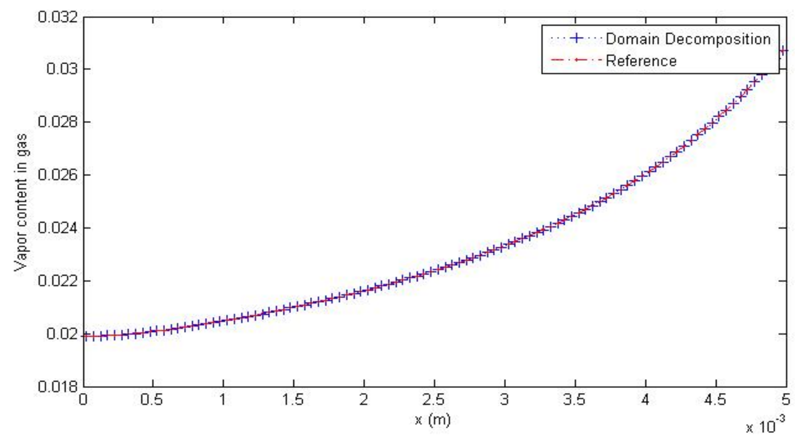

Figure 6.28: Comparison of vapor content (in gas) between domain decomposition and reference solution

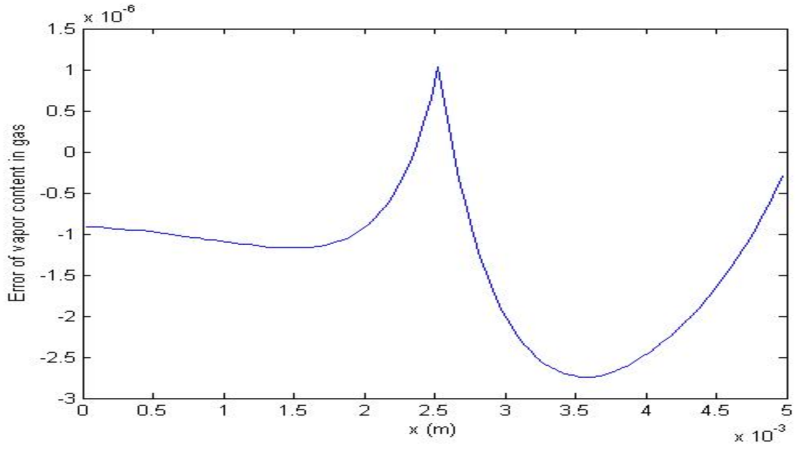

Figure 6.29: Difference of vapor content (in gas) between domain decomposition and reference solution

whereas at the last Newton iteration, which is roughly the Jacobian at the solution, is given by

$$\begin{bmatrix} 7.414\,10^{-8} & 8.708\,10^{-3} & -2.311\,10^{-8} & 0.0000 & 1.490\,10^{-2} \\ 1.518\,10^{-8} & 1.434\,10^{-2} & 1.292\,10^{-7} & 0.0000 & -1.035\,10^{-1} \\ 1.593\,10^{-3} & -1.810\,10^{+2} & 4.757\,10^{-3} & 0.0000 & 3.327\,10^{+3} \\ 1.606\,10^{-3} & 1.589\,10^{+3} & 1.344\,10^{-2} & 0.0169 & -1.076\,10^{+4} \\ 1.930\,10^{-8} & -8.248\,10^{-6} & -1.756\,10^{-7} & 0.0000 & 1.519\,10^{-1} \end{bmatrix}$$

The above two matrices indicate that the system, which arises from implementation of the defect correction scheme in this one layer problem, is not diagonal dominant.

Modified α method

The domain decomposition test for this method is conducted when the initial interface data are set closer to the reference solution in comparison with the intial data in the previous section. The notion 'closer' here refers to the distance $\|X_i - X_i^0|$ between the initial and reference data for each variable X_i. The initial data for this test are the following :

- $p_{g*}^0 = 10^4$ Pa + $1atm$
- $\epsilon_{l*}^0 = 0.30$
- $H_{g*}^0 = 4.0\ 10^4$ kJ
- $H_{l*}^0 = 2.00\ 10^5$ kJ
- $C_{a*}^0 = 0.10\ C_{g*}^0$

The numerical results are given in the plot of iterations of interface variables and their normalized residuals. As illustrated in Fig.6.30-6.34, the method fails to bring the initial interface values close to the reference solution for any variable. Consequently, the residuals in Fig.6.35-6.39 remain large.

In Fig.6.33, the enthalpy of liquid advances closer to the reference solution (shown by dotted line) since iteration-85, however the other variables are still distant from the reference. Although the residuals in Fig.6.35-6.38 after that iteration are reasonably low, the vapor mass residual in Fig.6.39 shows no improvement. It is expected that a good approximation of the reference solution will only be achieved when all residuals are low at the same iteration. It can also be seen from Fig.6.34 that the vapor content in gas remains 100% which is still far

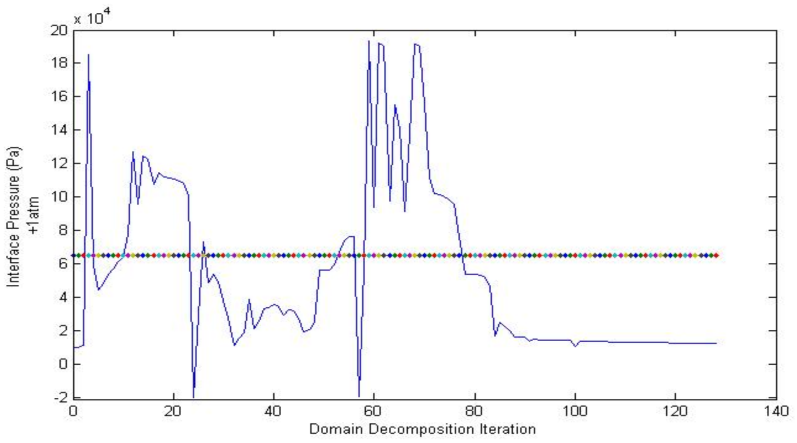

Figure 6.30: Iteration of interface pressure - single layer problem (Modified α method)

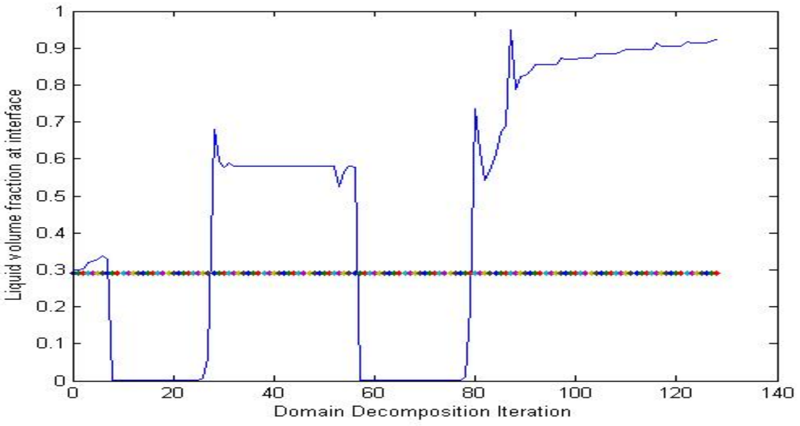

Figure 6.31: Iteration of liquid volume fraction at interface - single layer problem (Modified α method)

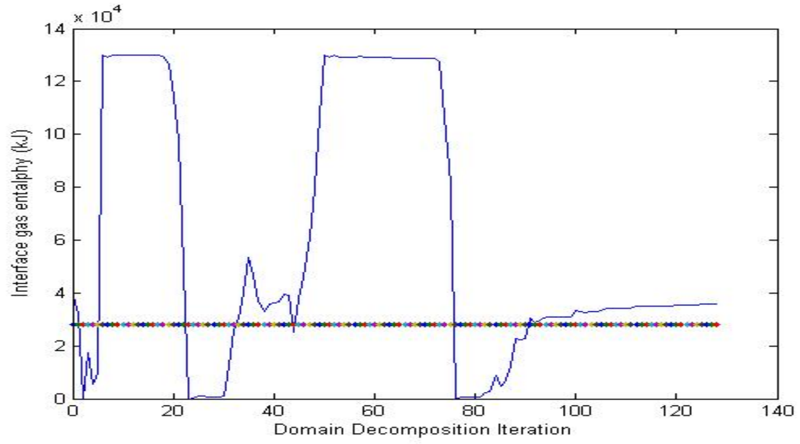

Figure 6.32: Iteration of interface gas enthalpy - single layer problem (Modified α method)

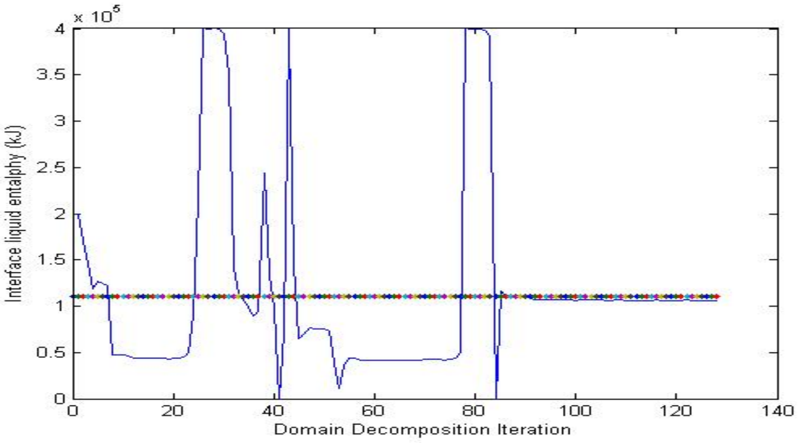

Figure 6.33: Iteration of interface liquid enthalpy - single layer problem (Modified α method)

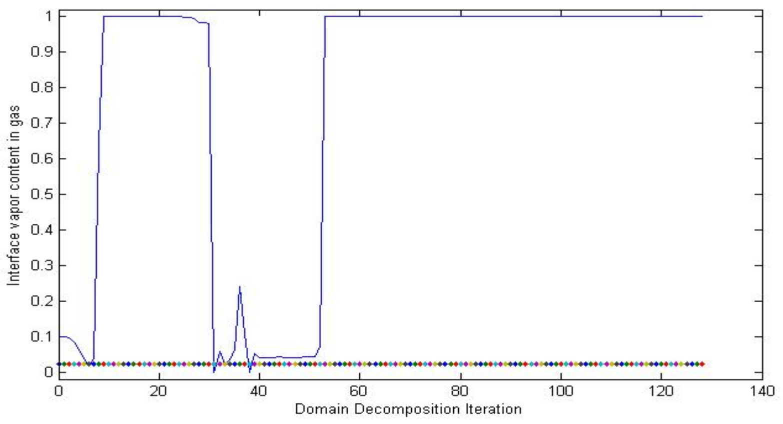

Figure 6.34: Iteration of interface vapor content in gas - single layer problem (Modified α method)

from the reference. The modified α-method cannot find a correction direction which lower the vapour content. There are also some earlier iterations $(11 - 30)$ when the vapor content also stays at 100% and the method is still able to lower the vapour content at iteration-31. However, the correction procedure of the vapour content is affected by the correction of other interface variables. In the case of iteration-85, the vapour content is trapped and the modified α method cannot recover it.

There are some explanations behind these failures. The modified α-method is a quasi-Newton method which seeks to approximate only the diagonal elements of the Jacobian. However, from Corollary-2 in Chapter-2, such quasi-Newton method only works well for a diagonal dominant system. In contrast to this, it has been shown in the previous section that the finite difference approximation of the Jacobian indicates that the problem is not diagonal-dominant.

6.3.2 Two layers of fabric

The second problem is a two layers of fabric where the effective radius of pore of the first and the second layer are 5.0 10^7 m and 5.0 10^6 m respectively, which induce material permeabilities $K_1 \approx 10^{-15}$ and $K_2 \approx 10^{-13}$. The decomposition into two subdomains is carried out by allocating 50 cells for the subdomain-1 and 49 cells for the subdomain-2. One extra cell between the two subdomains acts as the interface cell. The reference solution in the

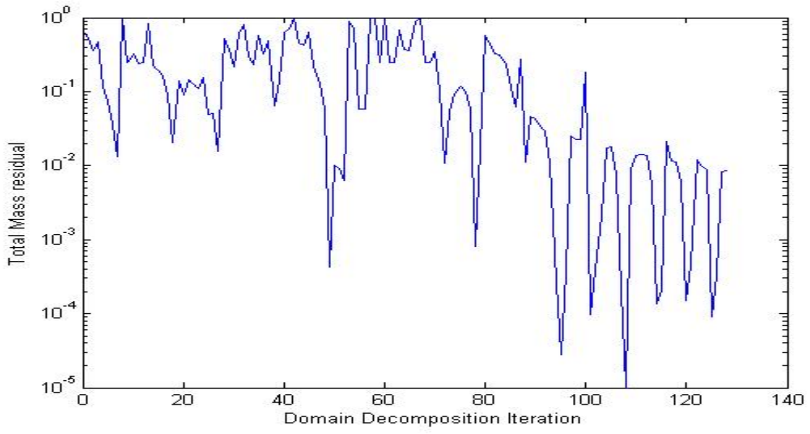

Figure 6.35: Normalized total mass residual at interface - single layer problem (Modified α method)

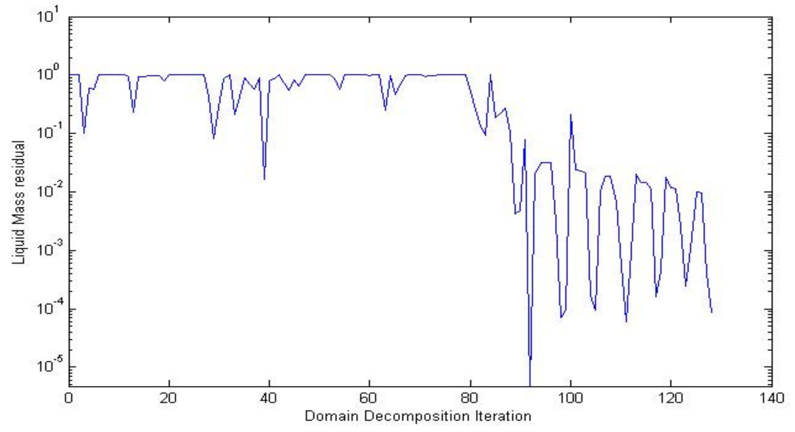

Figure 6.36: Normalized liquid mass residual at interface - single layer problem (Modified α method)

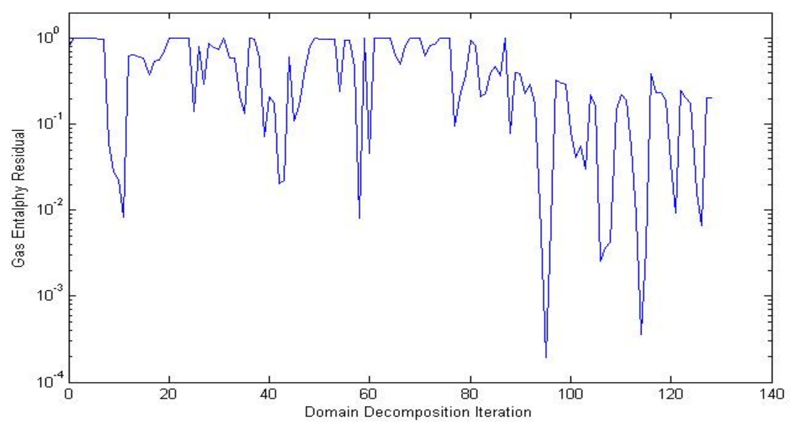

Figure 6.37: Normalized gas enthalpy residual at interface - single layer problem (Modified α method)

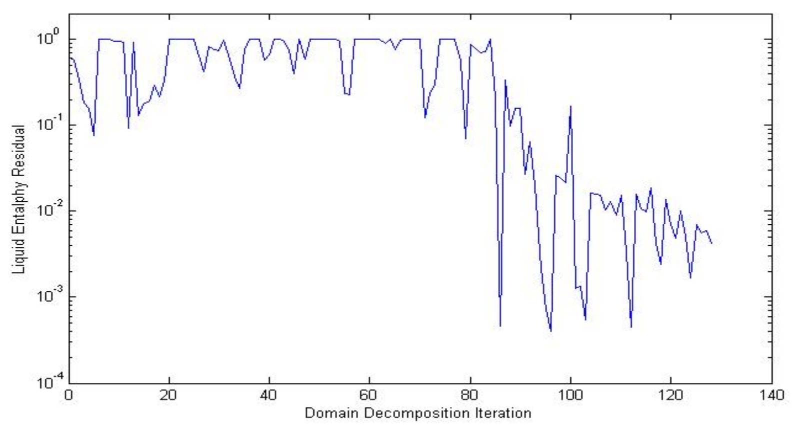

Figure 6.38: Normalized liquid enthalpy residual at interface - single layer problem (Modified α method)

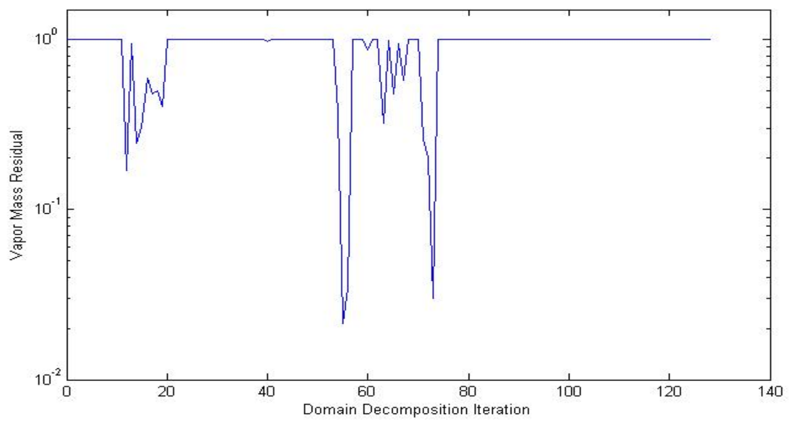

Figure 6.39: Normalized vapor mass residual at interface - single layer problem (Modified α method)

interface cell position is given by :

- $p_{g*} = 1.240719\ 10^3$ Pa + 1 atm
- $\epsilon_{l*} = 0.2398237$
- $H_{g*} = 2.795682\ 10^4$ kJ
- $H_{l*} = 1.111802\ 10^5$ kJ
- $C_{a*} = 0.03324072\ C_{g*}$

which is obtained from the solution of running the model throughout the entire domain as presented in the previous chapter.

Modified α Method

The first domain decomposition test for this 2-layer problem is implemented for the scheme using modified α method, which is started from the following initial interface data :

- $p^0_{g*} = 10^3$ Pa + 1 atm
- $\epsilon^0_{l*} = 0.20$
- $H^0_{g*} = 5.0\ 10^4$ kJ

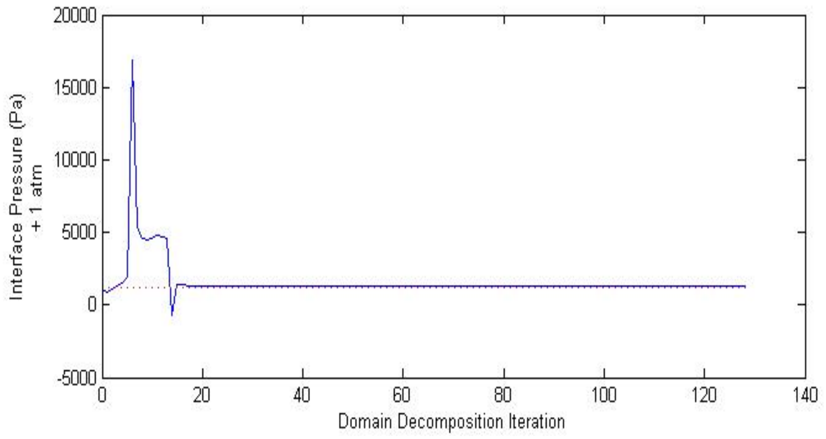

Figure 6.40: Iteration of interface pressure - two layers problem (modified α method)

- $H^0_{l*} = 4.00\ 10^5$ kJ
- $C^0_{a*} = 0.030\ C^0_{g*}$

The initial data P^0_{g*},ϵ^0_{l*}, and C^0_{a*} are chosen reasonably close to the reference solution. The numerical results are presented in the same format as in the single layer problem. However, just as in the single layer problem, the modified α method fails to do any significant correction to the interface variables. Fig.6.40 and Fig.6.41 are sufficient to show this.

Newton method

This section describes the numerical results of the defect correction scheme which use a Newton method for the two layers problem. As in the single layer problem, here the finite difference approach is also employed to approximate the Jacobian matrix. The initial interface data for this domain decomposition test are given below:

- $p^0_{g*} = 1$ atm
- $\epsilon^0_{l*} = 0.9999$
- $H^0_{g*} = 5.0\ 10^4$ kJ
- $H^0_{l*} = 2.00\ 10^5$ kJ
- $C^0_{a*} = C^0_{g*}$

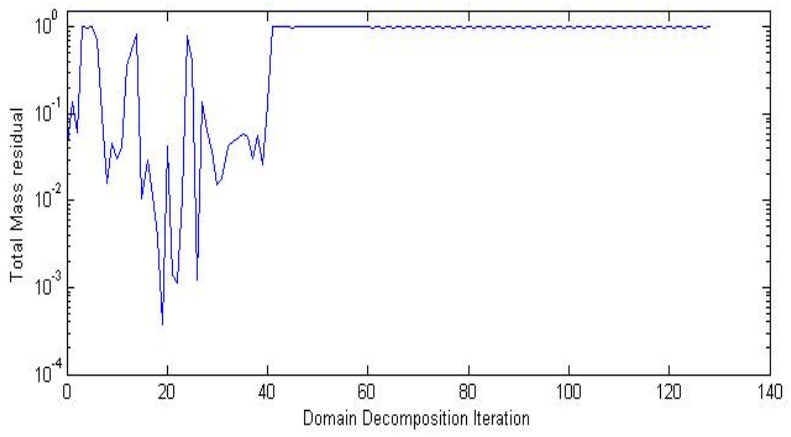

Figure 6.41: Normalized total mass residual at interface - two layers problem (modified α method)

Compared with the initial data for the modified α method, the values of P^0_{g*}, ϵ^0_{l*}, and C^0_{a*} in these initial data are further in Euclidean distance from the reference solution. Despite that, while the modified α method does not deliver, this Newton method manages to lead such initial data into a convergent solution which is very close to the reference. Fig.6.42-6.46 illustrate this for each interface variable.

All the normalized residuals in Fig.6.47-6.47 reach below 10^{-4} after 150 domain decompostion iterations, or equivalent to 25 Newton iterations. The domain decomposition solution profile along the fabric thickness also show good matches with the reference solution as demonstrated in Fig.6.52, 6.54, 6.56, 6.58,and 6.60. There are only minor discrepancies between the two solutions as illustrated in the associated plots in Fig.6.53, 6.55, 6.57, 6.59, and 6.61 respectively.

In the first Newton iteration, the Jacobian matrix computed by this method is delivered by

$$\begin{bmatrix} 3.442\,10^{-6} & 9.802\,10^{-3} & 1.365\,10^{-8} & 0.0000 & -7.125\,10^{-3} \\ 1.529\,10^{-6} & 1.552\,10^{-2} & 2.008\,10^{-7} & 0.0000 & -1.021\,10^{-1} \\ 5.092\,10^{-2} & -1.760\,10^{+2} & 4.854\,10^{-3} & 0.0000 & 2.233\,10^{+3} \\ 1.654\,10^{-1} & 2.020\,10^{+3} & 2.044\,10^{-2} & 0.0152 & -1.016\,10^{+4} \\ 8.873\,10^{-8} & 1.563\,10^{-4} & -1.922\,10^{-7} & 0.0000 & 1.101\,10^{-1} \end{bmatrix}$$

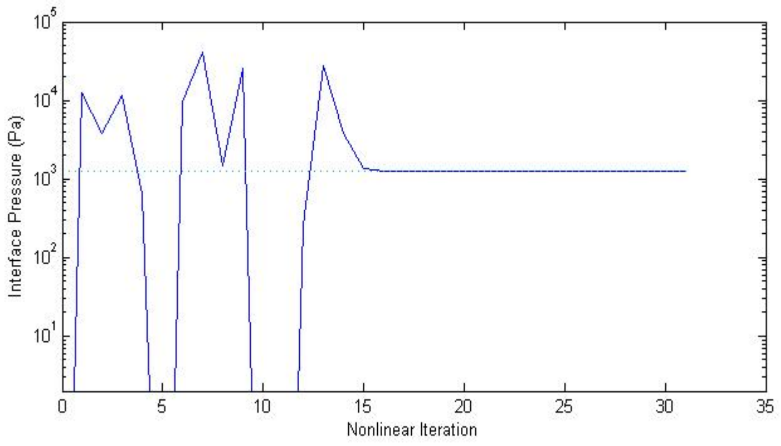

Figure 6.42: Iteration of interface pressure - two layers problem (Newton method)

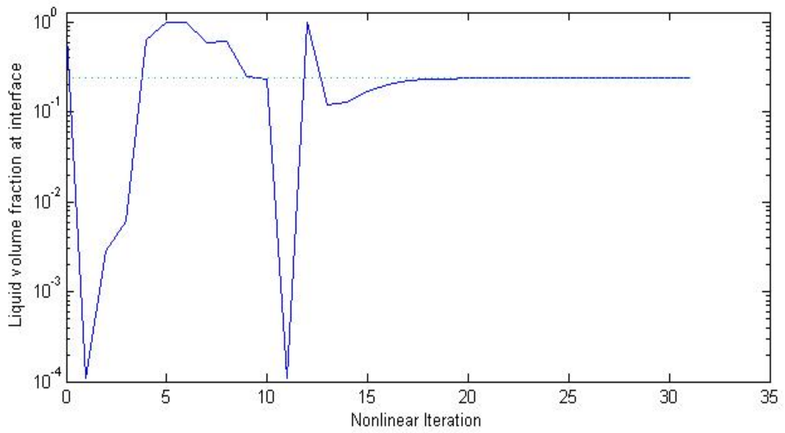

Figure 6.43: Iteration of interface liquid volume fraction - two layers problem (Newton method)

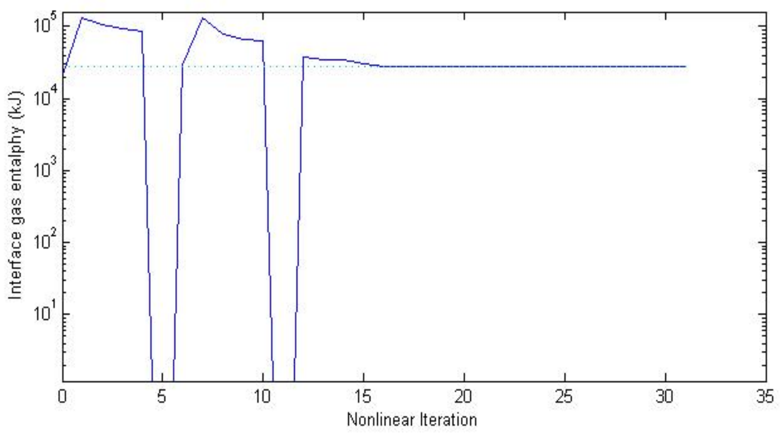

Figure 6.44: Iteration of interface gas enthalpy - two layers problem (Newton method)

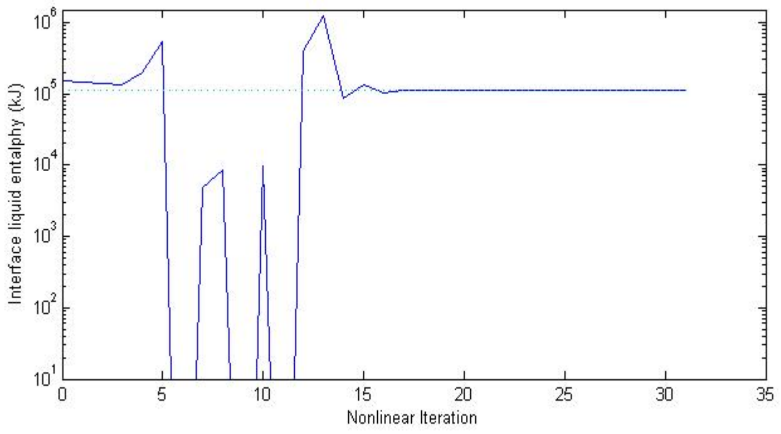

Figure 6.45: Iteration of interface liquid enthalpy - two layers problem (Newton method)

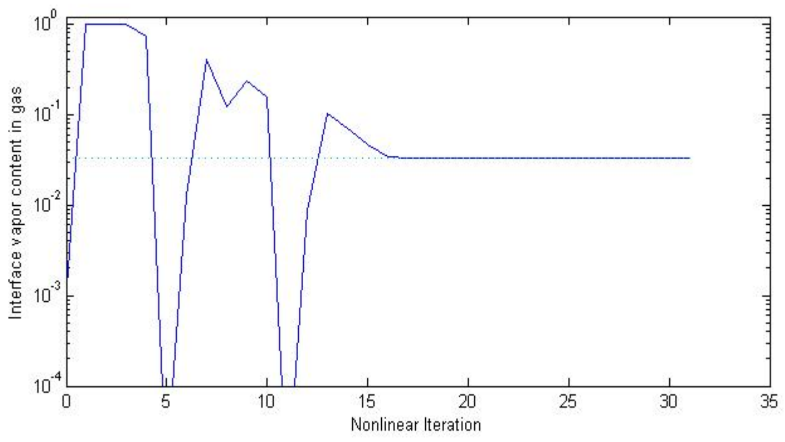

Figure 6.46: Iteration of interface vapor content in gas - two layers problem (Newton method)

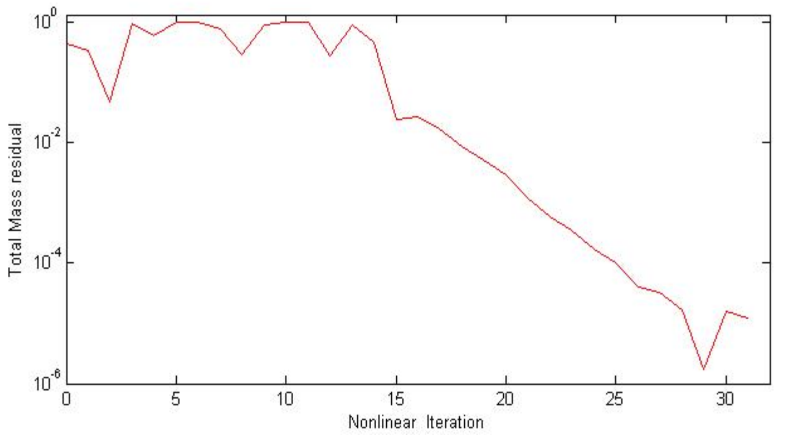

Figure 6.47: Normalized total mass residual at interface - two layers problem (Newton method)

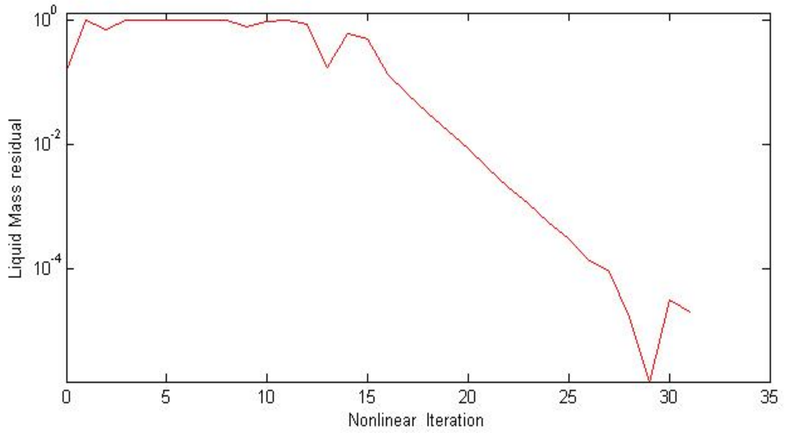

Figure 6.48: Normalized liquid mass residual at interface - two layers problem (Newton method)

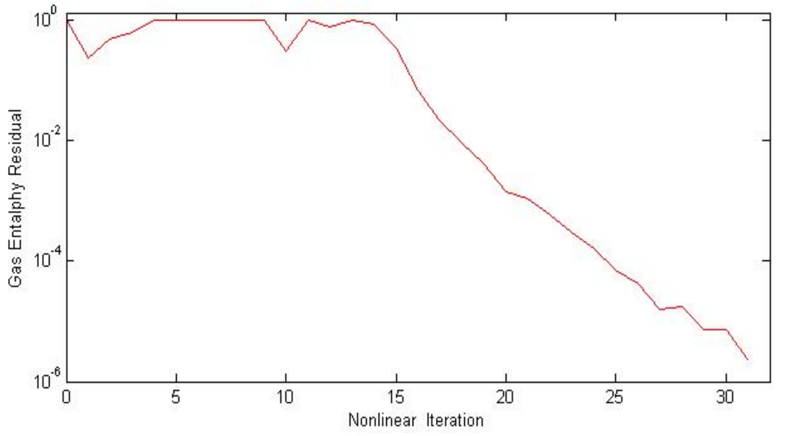

Figure 6.49: Normalized gas enthalpy residual at interface - two layers problem (Newton method)

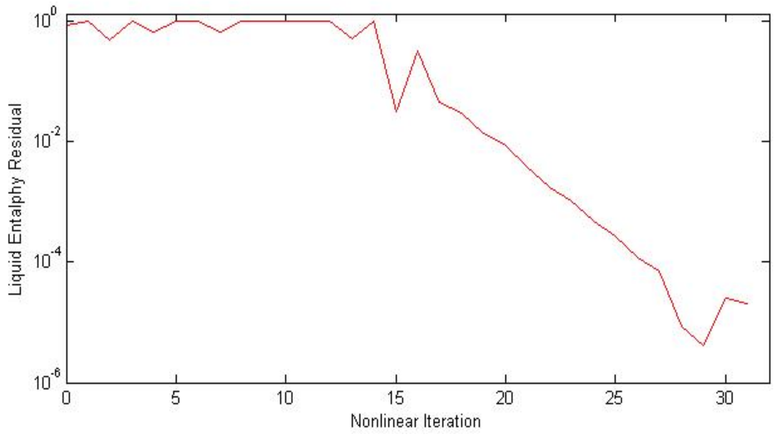

Figure 6.50: Normalized liquid enthalpy residual at interface - two layers problem (Newton method)

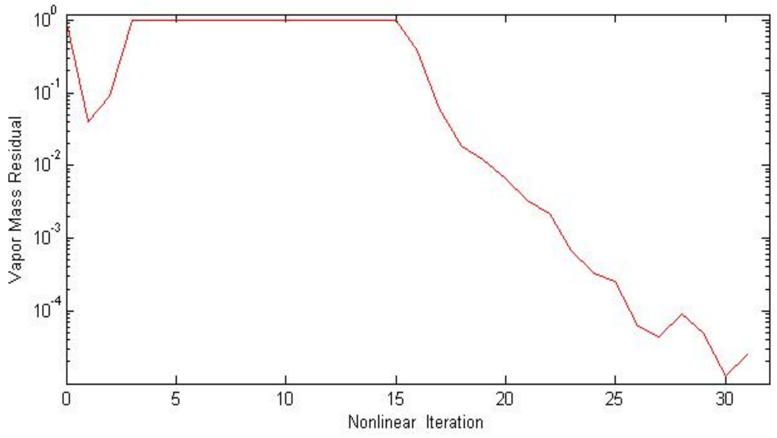

Figure 6.51: Normalized vapor mass residual at interface - two layers problem (Newton method)

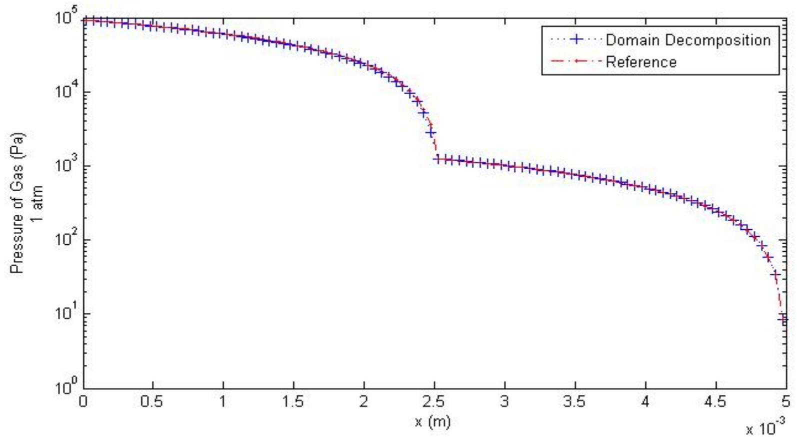

Figure 6.52: Comparison of pressure between domain decomposition and reference solution (two layers)

while the last Newton iteration gives

$$\begin{bmatrix} 1.891\,10^{-5} & 9.430\,10^{-4} & 5.330\,10^{-8} & 0.0000 & -8.019\,10^{-2} \\ 1.883\,10^{-5} & -3.222\,10^{-4} & 9.900\,10^{-8} & 0.0000 & -1.237\,10^{-1} \\ 3.548\,10^{-3} & 4.517\,10^{+1} & -2.751\,10^{-4} & 0.0000 & 9.155\,10^{+2} \\ 2.568\,10^{00} & 5.298\,10^{+2} & -3.433\,10^{-3} & 0.0478 & 2.441\,10^{+2} \\ 3.912\,10^{-8} & 1.413\,10^{-3} & -4.885\,10^{-8} & 0.0000 & 4.701\,10^{-2} \end{bmatrix}$$

Both matrices indicate that the system created by the defect correction scheme has a non-diagonal dominant Jacobian matrix.

6.4 Closure

This chapter has demonstrated the implementation of some defect correction schemes in the single-phase and multiphase problems whose descriptions have been illustrated in the previous chapter. The schemes couple two subdomain solutions computed by the PHOENICS code.

In the multiphase problem, five interface variables are chosen for the schemes, i.e. one of the phase pressure, one of the phase volume fraction, temperatures of both phases, and the vapour concentration. This setting then needs five defect equations at the interface, where the equations are represented by the residuals of governing equations.

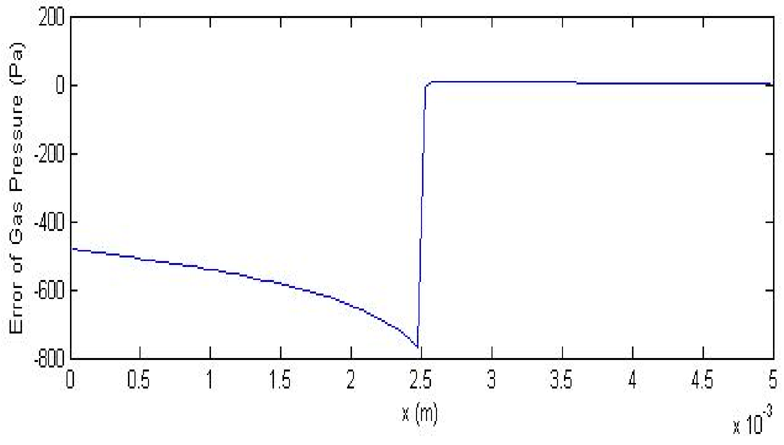

Figure 6.53: Difference of pressure between domain decomposition and reference solution (two layers)

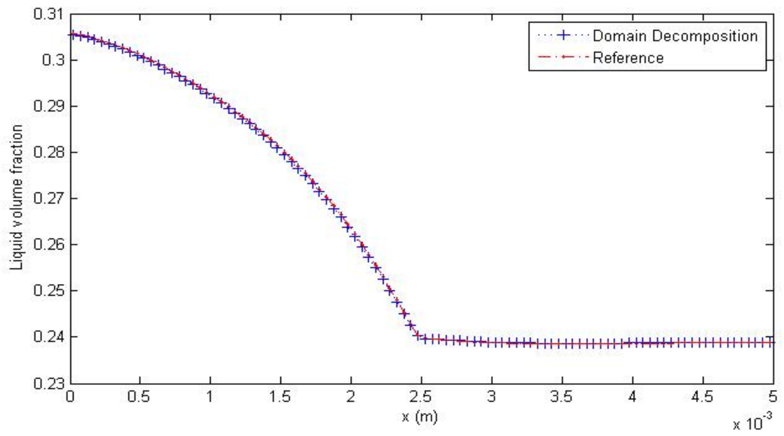

Figure 6.54: Comparison of liquid volume fraction between domain decomposition and reference solution (two layers)

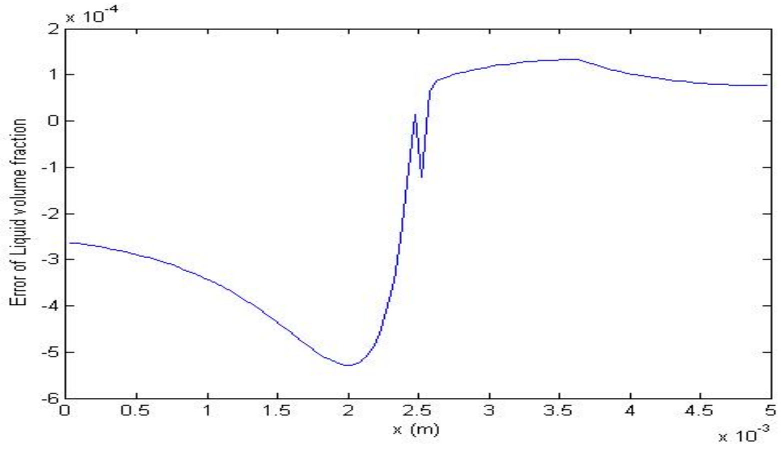

Figure 6.55: Difference of liquid volume fraction between domain decomposition and reference solution (two layers)

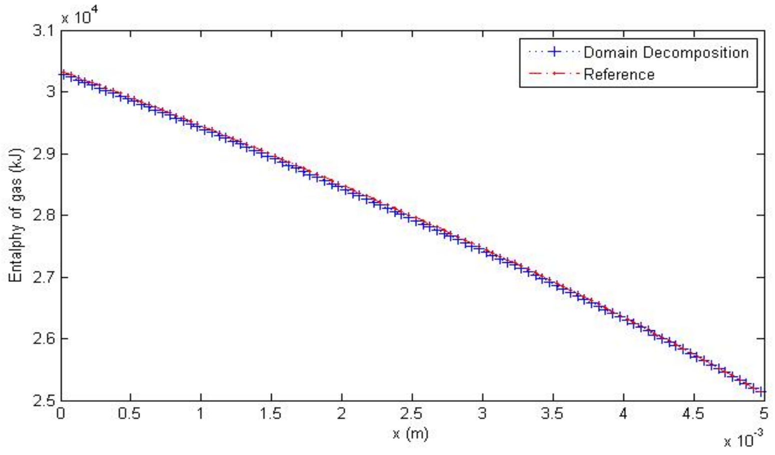

Figure 6.56: Comparison of gas enthalpy between domain decomposition and reference solution (two layers)

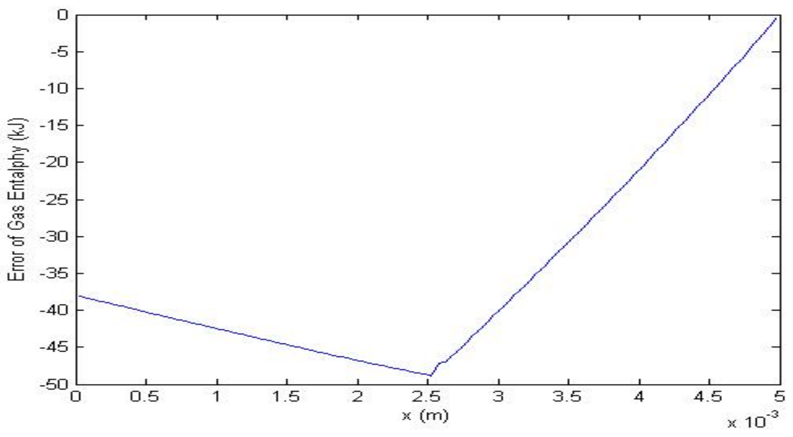

Figure 6.57: Difference of gas enthalpy between domain decomposition and reference solution (two layers)

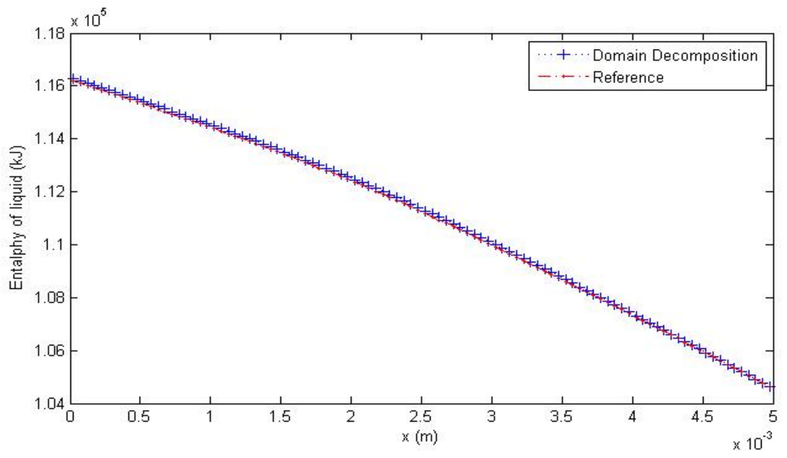

Figure 6.58: Comparison of liquid enthalpy between domain decomposition and reference solution (two layers)

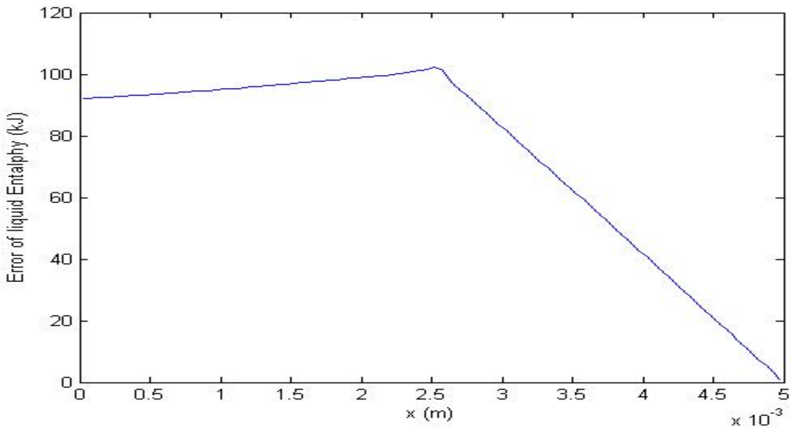

Figure 6.59: Difference of liquid enthalpy between domain decomposition and reference solution (two layers)

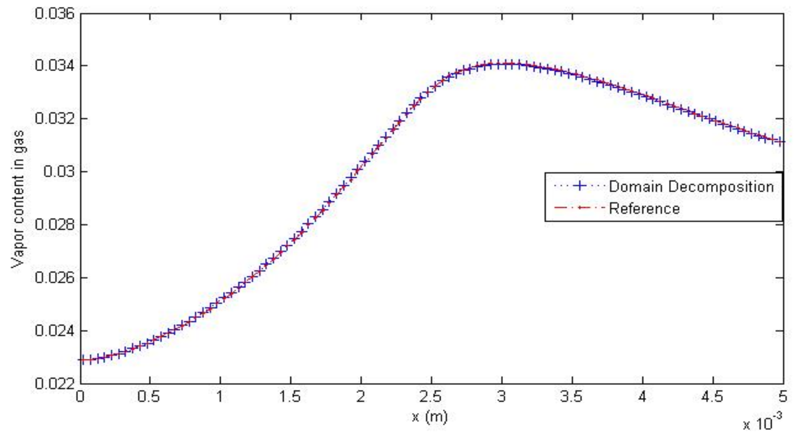

Figure 6.60: Comparison of vapor content (in gas) between domain decomposition and reference solution (two layers)

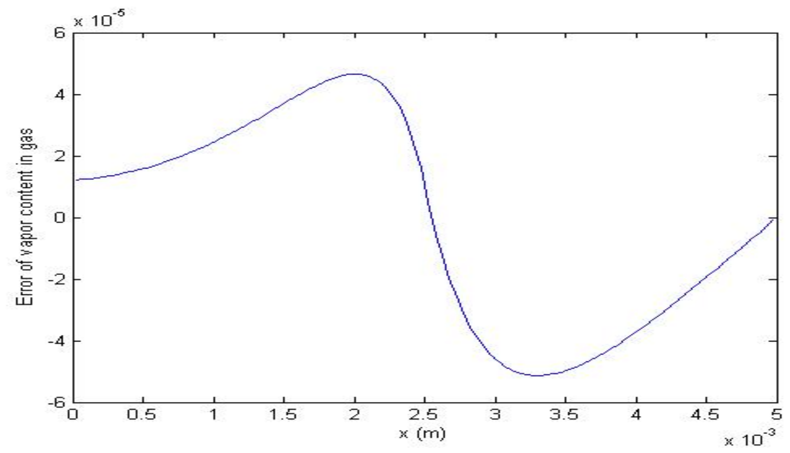

Figure 6.61: Difference of vapor content (in gas) between domain decomposition and reference solution (two layers)

One of the schemes uses the modified-α method and it uses a set of five scalar adaptive variables to represent its nonlinear solver where each variable corresponds one-to-one with a defect equation and an interface variable. Therefore the iteration of each interface variable is controlled by its only corresponding scalar variable. In both the single layer and two layers problems, this fails to deliver a good convergent solution.

Another scheme is also implemented where it uses a Newton method as the nonlinear solver and the Jacobian matrix is computed using the finite difference approximation. In both problems, this scheme produces satisfying results where it can lead an initially distant interface data into a convergent solution which is very close to the reference. When computed with a finite difference approximation, the resulting Jacobian approximations indicate that Jacobian matrix of the system is not diagonal dominant for both problems. This may explain the failure of the modified α method since it solely approximates the diagonal element of the Jacobian matrix and hence requires a much smaller local neighbourhood for a multivariable system like this multiphase problem.

In the single phase problem where the solved variables are pressure and enthalpy, the implementation of the defect correction scheme in PHOENICS yields a pressure profile which approximates well the conventional solution when mesh is refined. Yet similar pattern is not found for the enthalpy profile. Furthermore, the numerical experiments show that the number of nonlinear iteration increases with the mesh refinement, which is also indicated in the heat conduction problem in Chapter-4.

Chapter 7

Conclusions and Further Works

The aim of this thesis has been to examine the iterative framework of a class of nonoverlapping domain decomposition methods which allow autonomous subdomain computations and coupling flexibilities at the interface. These methods have capabilities such as coupling problems of different physics, accommodating different numerical treatments, and enabling component meshing and gluing. A framework of defect correction schemes is proposed where interface residuals, in the form of defect functions, and their solvers are at the heart of the schemes.

In order to take care of the residual at the interface, some nonlinear solvers are proposed. An adaptive-α method is presented where a numerical analysis has been conducted and it is found that by using this method, the solution can converge locally in the ∞-norm, where the sufficient condition needs a smaller local neighbourhood than many other quasi-Newton methods, and the problem must have a strongly diagonal dominant Jacobian matrix with a very small condition number. Yet its advantage can be of high significance in the computational cost. It simply uses a scalar value to represent the Jacobian matrix of the interface problem. Therefore, both the computation and the inversion of the Jacobian matrix can be circumvented.
There are three other nonlinear solvers employed in this work which do not require any Jacobian approximation, i.e. a Newton-GMRES method and nonlinear conjugate gradient solvers with Fletcher-Reeves and Pollak-Ribiere searching direction formulas. For the sake of some comparisons, a Newton method with a finite difference Jacobian approximation is also included.

The defect correction scheme is not originally developed as a preconditioner of a linear system as can be found in many classical domain decomposition methods. Nevertheless, it has been shown that if the scheme is employed for a linear problem, it may act as a left preconditioner of the corresponding linear system. In the case of the adaptive α, the resulting preconditioner is a variable preconditioner due to the changing nature in its adaptivity. Given the fact that it is a a diagonal matrix of uniform elements, it contains less information than the Neumann-Neumann or the Dirichlet-Neumann preconditioners which are built from the Schur complement matrices. However, this simple structure enables the flexibility of the scheme which separates the interface solver from subdomain computations. Moreover the flexibility can also be extended to the mortar element framework.

Some implementations of the defect correction schemes in a nonlinear heat conduction process in a multichip module have been performed to satisfy the concept of component meshing and gluing. The scheme divides the module into 12 subdomains according to the physical structure of the component, solves the 12 subproblem independently, and couples the subdomain solutions using the defect equations. Two defect equations are tried in the interface coupling, i.e. the equality of normal derivative and a discretisation of the residual of the governing equation. While the latter yields a much more accurate solution, the former basically disobeys the physics of the heat across the interface. Of the nonlinear solvers attempted, the nonlinear conjugate gradient solvers produce more satisfying results. Yet the Newton-GMRES needs some attention since its average number of inner iteration in one domain decomposition step is relatively smaller.

The implementations of some defect correction schemes in fluid flows with heat transfer problems have been carried out which couples the subdomain solution produced by the PHOENICS CFD code. The first problem is a single-phase 2D problem and the second one is a 1D multiphase fluid flow with heat and moisture in porous fabrics. The second problem is governed by a coupled system of differential equations. For the setting of its interface coupling, five interface variables and five defect equations are chosen. A scheme uses a modification of the adaptive-α method to represent its nonlinear solver, where there are five scalar variables which corresponds one-to-one with a defect equation and an interface variable. This modification is chosen to adjust with the number of defect equations and at the same time avoids any Jacobian matrix computation and inversion. The numerical results show that this method fails to obtain satisfying solutions in two cases (the single layer and two layers of fabric) even when the initial interface data is rather close to the reference solution.

Another scheme is also implemented where it uses a Newton method as the nonlinear solver and the Jacobian matrix is computed using the finite difference approximation. In both cases, this scheme produces satisfying results where it can lead an initially distant interface data into a convergent solution which is very close to the reference. When computed with a finite difference approximation, the resulting Jacobian approximations indicate that Jacobian matrix of the system is not diagonal dominant for both problems. This may explain the failure of the modified α method since it solely approximates the diagonal element of the Jacobian matrix and hence requires a much smaller local neighbourhood for a multivariable system like this multiphase problem.

This Newton method also manages to solve the coupling of PHOENICS' subdomain computations for the first problem (2D single-phase fluid problem). The Newton-GMRES solver is also tried for this problem, yet it works very slow towards reaching the desired interface solution. In general, it is found that the number of nonlinear iteration of the defect correction scheme increases with the mesh refinement.

Further work may begin with the development of a good and efficient nonlinear solver. A complex problem such as multiphase heat and moisture transfer will need a nonlinear solver which contains a good information of the Jacobian matrix. Yet the approximation of Jacobian matrix with finite difference will be too expensive for problems with many interface points. The compromise between the convergence of the solver and the computational load may be required.

After that, an extension of the current steady state multiphase problem into transient problems which include sorption process inside fibres can be considered. When boundary conditions are set where humidity and saturated vapour concentration affect the diffusive and convective interaction with environments, the physical model of the heat and moisture transfer in the clothing material will be more realistic.

Bibliography

[1] Y. Achdou, P.L.Tallec, F.Nataf, and M.Vidrascu, 'A domain decomposition preconditioner for an advection-diffusion problem', *Comp. Methods Appl. Mech. Engrg*, 184 (2000), pp.145-170.

[2] M. Aihua, L. Yi, 'Establishment of computational models for clothing engineering design', *Journal WSEAS Transactions on Computers*, 7 (2008), pp. 770-780

[3] G. Allaire, *Numerical Analysis and Optimization : An Introduction to Mathematical Modelling.* (Oxford University Press, 2007).

[4] T. Arbogast, G. Pencheva, M. F. Wheeler, and I. Yotov, 'A multiscale mortar mixed finite element method', *SIAM J. Multiscale Modeling and Simulation*, 6:1 (2007) pp. 319-346.

[5] L. Armijo, 'Minimization of functions having Lipschitz continuous first derivatives', *Pacific. Journal of Mathematics* , 16 (1966) pp.1-3.

[6] T. Ayepe,'Collaborating nonlinear elliptic and parabolic problems', *MSc thesis*, University of Greenwich.

[7] B. Bang and D. Lukkassen, 'Application of Homogenization Theory Related to Stokes Flow in Porous Media', *Application of Mathematics*, 44 (1999), pp.309319.

[8] R. Barrett, M. Berry, T.F.Chan, *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods.* (SIAM, 1994).

[9] J.J. Barry, R.W.Hill, 'Computational modeling of protective clothing', *International Nonwovens Journal*, 12 (3) (2003), pp.2534.

[10] F.B. Belgacem, 'The mortar finite element method with lagrange multipliers', *Numer. Math.*, 84 (1999), pp.173197.

[11] C. Bernardi, Y. Maday, and A. Patera, 'Domain decomposition by the mortar finite element method', *Asymptotic and Numerical Methods for PDEs with Critical Parameters*, 384 (1993), pp.269286.

[12] P.E. Bjorstad, 'Multiplicative and additive Schwarz methods: Convergence in the 2-domain case, *Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, (1989), pp. 147-159.

[13] P.E. Bjorstad and O.Widlund, 'To overlap or not to overlap : A note on a domain decomposition method for elliptic problems'. *SIAM J Sci Statist Comput*, 10 (1989), pp.1053-1061.

[14] D.Braess *Finite Elements*. (Cambridge University Press, 2002).

[15] J.H. Bramble, J.E. Pasciak, and A.H. Schatz,'The construction of preconditioners for elliptic problems by substructuring', *Mathematics of Computation*, 47 (1986), pp.103-134.

[16] P.N. Brown, Y. Saad, 'Hybrid Krylov methods for nonlinear systems of equations', *SIAM J. Sci. Stat. Comput.* , 11 (1990) pp.450481.

[17] K.P. Byrne, 'Calculating the acoustical properties of fabric constructions', *Journal of Sound and Vibration*, 123 (3) (1988), pp.423435.

[18] X.-C. Cai, 'Additive Schwarz algorithms for parabolic convection-diffusion equations', *Numer. Math.*, 60 (1991), pp.41-61.

[19] X.-C. Chai, W.D.Gropp, D.E.Keyes, R.G.Melvin, and D.P.Young, 'Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation', *SIAM J. Sci. Comput*, 19.1(1998), pp.246-265. .

[20] X.-C. Chai, W.D.Gropp, D.E.Keyes, and M.D.Tidriri, 'Newton-Krylov-Schwarz methods in CFD', *Proceeding of the International Workshop on the Navier-Stokes Equations, Notes in Fluid Mechanics*, (Vieweg Verlag, Braunschweig, 1994), pp.17-30.

[21] T.F. Chan, T.P.Mathew, 'Domain decomposition algorithms', *Acta Numerica*, (1994), pp.61-143.

[22] R. Choquet, J.Erhel, 'Some Convergence Results for the Newton-GMRES Algorithm', *Technical report, INRIA*, 2065 (1993) pp.119.

[23] P. Chow and C.Addison, 'Putting domain decomposition at the heart of a mesh-based simulation process'. *Int. J. Numer. Meth. Fluids*, 40 (2002), pp.1471-1484.

[24] P. Chow, C.-H. Lai, 'Electronic packaging and reduction in modelling time using domain decomposition'. *Lecture Notes in Computational Science and Engineering*, 40 (Springer-Verlag, 2004), pp. 193-200.

[25] P. Chow, C.-H.Lai,'Collaborating components in mesh based electronic packaging', *Scientific Programming*, 12(IOS Press, 2004), pp.65-70

[26] A.M. Cuffe, C.-H.Lai and K.Pericleous, 'Adaptive zonal recognition for viscous/inviscid coupling ', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.690-697.

[27] C. Dawson, Q. Du and T.F. Dupont,'A finite difference domain decomposition algorithm for numerical solution of the heat equation',*Math. Comput.*, 57 (1991), pp.63-71.

[28] J.A. Dean, *Lange's Handbook of Chemistry, 15th Edition.* (McGraw-Hill Professional, 1998).

[29] J.E. Dennis and R.B. Schnabel,*Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, (SIAM, 1996).

[30] S. Deparis, M. Discacciati, G. Fourtestey, A.Quarteroni,'Heterogeneous domain decomposition methods for fluid-structure problems',*Proceedings of the 17th International Conference on Domain Decomposition Methods*, (2007), pp.41-52.

[31] Q.V. Dinh, R. Glowinski, B. Mantel, J. Periaux, P.Perrier, 'Subdomain solution of nonlinear problems in fluid dynamics on parallel processors', *Computing Methods in Applied Sciences and Engineering*,North-Holland, Amsterdam, (1982), pp.123-164.

[32] Q.V. Dinh, R. Glowinski, J. Periaux, Solving Elliptic Problems by Domain Decomposition Methods with Applications, *Elliptic Problem Solvers II*, Academic Press, Orlando, (1984), pp. 395-426.

[33] V. Dolean, M. Gander, 'Why classical Schwarz methods applied to hyperbolic systems can converge even without overlap', *Lecture Notes in Computational Science and Engineering*,Springer Verlag,60 (2008), pp.467475.

[34] V. Dolean and F.Nataf, 'An Optimized Schwarz algorithm for the compressible Euler equations',*Proceedings of the 17th International Conference on Domain Decomposition Methods*, (2007), pp.173-180.

[35] M. Dryja,'A finite element-capacitance method for elliptic problems on regions partitioned into subregions',*Numer. Math.*, 44 (1984), pp.153-168.

[36] M. Dryja, O. Widlund, 'An additive variant of the Schwarz alternating method for the case of many subregions',*Tech Report 339 Ultracomputer Note 131 Courant Institute*, New York, NY, (1987).

[37] F. Duval, F. Fichot and M. Quintard, 'A local thermal non-equilibrium model for two-phase flows with phase-change in porous media', *Int.J. of Heat and Mass Transfer*, 47 (2004), pp.613639.

[38] M.Espedal, K.Hersvik and B.Ergsland, 'Domain decomposition methods for flow in heterogeneous porous media', *Proceedings of the 10th International Conference on Domain Decomposition Methods*, (1998), pp.104-120.

[39] C. Farhat and M. Geradin, 'Using a reduced number of Lagrange multipliers for assembling parallel incomplete field finite element approximations', *Comput. Methods Appl. Mech. Engrg.*, 97 (1992), pp. 333-354.

[40] C. Farhat, F.X. Roux, 'A method of finite element tearing and interconnecting and its parallel solution algorithm', *Internat. J. Numer. Meth. Engrg.*, 32 (1991), pp.1205-1227.

[41] X.Fengl, 'Interface conditions and non-overlapping domain decomposition methods for a fluid-structure interaction problem', *Proceedings of the 10th International Conference on Domain Decomposition Methods*, (1998), pp.417-424.

[42] B.Flemisch, B.Wohlmuth,'Nonconforming methods for nonlinear elasticity problems',*Proceedings of the 17th International Conference on Domain Decomposition Methods*, (2007), pp.65-76.

[43] R. Fletcher and C. M. Reeves, 'Function Minimization by Conjugate Gradients', *Comp. J.*, 7 (1964), pp.149-154.

[44] J.Galvis, M.Sarkis, 'Balancing domain decomposition methods for mortar coupling Stokes-darcy systems', *Proceedings of the 16th International Conference on Domain Decomposition Methods*, (2007), pp.373-382.

[45] J.Galvis, M.Sarkis, 'FETI-DP for Stokes-Mortar-Darcy Systems', *Lecture Notes in Computational Science and Engineering*, 78 (2011) pp.27-38.

[46] M.Gander, 'Overlapping Schwarz for parabaolic problems', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.97-104.

[47] M.Gander, L.Halpern, C.Japhet, 'Optimized Schwarz algorithms for coupling convection and convection-diffusion problems', *Proceedings of the 13th International Conference on Domain Decomposition Methods*, (2002), pp.255-262.

[48] B. Ganis and I. Yotov, 'Implementation of a Mortar Mixed Finite Element Method using a Multiscale Flux Basis', *Comp. Meth. in Appl. Mech. and Engng.*, 198 (2009) pp.3989-3998.

[49] M. Garbey, C.Picard, and R.S.Tay, 'Heterogeneous domain decomposition for multiscale problems', *43rd Aerospace Sciences Meeting and Exhibit Conference*, Reno, 2005.

[50] F. Gastaldi, A. Quarteroni and G. Sacchi-Landriani, 'On the coupling of two dimensional hyperbolic and elliptic equations: analytical and numerical approach', *Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, (1990), pp. 23-63.

[51] K.Ghali, N.Ghaddar, B.Jones, 'Modeling of heat and moisture transport by periodic ventilation of thin cotton fibrous media', *International Journal of Heat and Mass Transfer*, 45 (2002), pp.37033714.

[52] K.Ghali, B.Jones, J.Tracy, 'Modeling heat and mass transfer in fabrics', *International Journal of Heat and Mass Transfer*, 38 (1995), pp.1321.

[53] R.Glowinski, 'Numerical solution of partial differential equation problems by domain decomposition. Implementation on an array processors system', *Proceedings of International Symposium on Applied Mathematics and Information Science*, Kyoto University, 1982.

[54] G.H. Golub, C.F. Van Loan, *Matrix Computations.* (The Johns Hopkins University Press, 1996).

[55] I.Graham, P.Lechner, 'Domain decomposition for heterogeneous media', *Proceedings of the 16th International Conference on Domain Decomposition Methods*, (2007), pp.575-580.

[56] A. Grama, G.Karypis, V.Kumar, A.Gupta *Introduction to Parallel Computing.* (Addison Wesley, 2003).

[57] S.Guo, *Distributed Algorithms for Heat Transfer related Problems*, (MSc thesis, University of Greenwich, 2010).

[58] C.A. Harper, *Electronic packaging and interconnection handbook.* (McGraw-Hill, 2005).

[59] C.Hirsch, *Numerical computation of internal and external flows: fundamentals of computational fluid dynamics, Volume 1.* (Butterworth-Heinemann, 2007).

[60] K.H. Hoffmann, J. Zou, 'Parallel efficiency of domain decomposition methods', *Parallel Comput.*, 19 (1993), pp.13751391.

[61] P.R.Hugh, D.A.Knoll, and D.E.Keyes, 'Application of a Schwarz-preconditioned Newton-Krylov algorithm to a low-speed reacting flow problem', *AIAA Journal* 36 (1998), pp.290-292.

[62] F.P.Incropera, D.P.De Witt, *Introduction to Heat Transfer.* (John Wiley and Sons, 1990).

[63] C.Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method.* (Cambridge University Press, 1988).

[64] D.E.Kaushik, D.E.Keyes, and B.F.Smith, 'NKS methods for compressible and incompressible flows on unstructured grids', *Domain Decomposition Methods in Sciences and Engineering : Eleventh International Conference*, (1999), pp.513-520.

[65] C.T.Kelley, *Iterative Methods for Linear and Nonlinear Equations*, (SIAM, 1995).

[66] C.T.Kelley, *Solving Nonlinear Equations with Newton's Method*, (SIAM, 2003).

[67] D.E.Keyes, 'Domain decomposition methods for the parallel computation of reacting flows', *Comput. Phys. Comp.*, 53 (1989), pp. 181-200.

[68] D.E. Keyes and W.D.Gropp, 'A comparisons of domain decomposition techniques for elliptic partial differential equations and their parallel implementation'. *SIAM J Sci Statist Comput*, 8 (1987), pp.166-202.

[69] H.H. Kim, 'Domain Decomposition Algorithms for Mortar Discretizations', *Lecture Notes in Computational Science and Engineering*, 60 (2008) pp.81-92.

[70] R. Klessig and E. Polak, 'Efficient Implementation of the Polak-Ribiere Conjugate Gradient Algorithm', *SIAM J. Control*, 10 (1972), pp.524-549.

[71] D.A.Knoll, and D.E.Keyes, 'Jacobian-free NewtonKrylov methods: a survey of approaches and applications', *Journal of Computational Physics* 193 (2004), pp.357397.

[72] M. Kreienmeyer, E. Stein, 'Efficient parallel solvers for boundary element equations using data decomposition', *Engineering Analysis with Boundary Elements*, 19 (1997), pp.33-39.

[73] E.Kreyszig *Introductory functional analysis with applications.* (Wiley, 1978).

[74] C.Lacour, Y.Maday, 'Two different approaches for matching nonconforming grids: the mortar element method and the FETI method', *BIT Numerical Mathematics 37*, 3 (1997), pp.720-738.

[75] C.-H.Lai, 'An iteration scheme for non-symmetric interface operator', *ERCIM Research Report*, (1992).

[76] C.-H.Lai, 'An iteration scheme for non-symmetric operator', *Contemporary Mathematics*, 157 (1993), pp.279-285.

[77] C.-H. Lai. An application of quasi-Newton methods for the numerical solution of interface problems. *Advances in Engineering Software*, 28 (1997), pp. 333 - 339.

[78] C.-H. Lai, A.M. Cuffe and K.A.Pericleous, 'A defect equation approach for the coupling of subdomains in domain decomposition methods'. *Advances in Engineering Software*, 26 (1996), pp.151 - 159.

[79] S.C. Lee, M.N. Vouvakisa, J.F. Lee, 'A non-overlapping domain decomposition method with non-matching grids for modeling large finite antenna arrays', *Journal of Computational Physics*, 203 (2005), pp.1-21.

[80] R.J. Leveque, Z. Li, 'The immersed interface method for elliptic equations with discontinuous coefficients and singular sources', *SIAM Journal on Numerical Analysis*, 31 (1994), pp.10191044.

[81] J.Li, 'A Dual-Primal FETI Method for Solving Stokes/Navier-Stokes Equations', *Proceedings of the Fourteenth International Symposium on Domain Decomposition Methods*, (2003) pp.225-231.

[82] Y.Li and Z.X. Luo, 'Physical mechanisms of moisture diffusion into hygroscopic fabrics during humidity transients', *J.Textile Inst.*, 91 (2000), pp.302323.

[83] P.L. Lions, 'On the Schwarz alternating method I', *Proceedings of First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, (1988), pp. 1-42.

[84] D.G. Luenberger, Y. Ye, *Linear and nonlinear programming.* (Springer, 2008).

[85] R.J.Mackinnon, G.F.Carey, 'Analysis of material interface discontinuities and superconvergent fluxes in finite difference theory', *J.Computational Physics*, 75 (1983), pp.151166.

[86] A.Mikelic, 'Homogenization of nonstationary Navier-Stokes equations in a domain with a grained boundary', *Annali di Matematica Pura ed Applicata*, 158 (1991), pp.167-179.

[87] M.L. Minges, *Electronic Materials Handbook: Packaging, Volume I.* (CRC Press, 1989).

[88] F.Nataf, G.Rapin, 'Construction of a new domain decomposition method for the Stokes equations',*Proceedings of the 17th International Conference on Domain Decomposition Methods*, (2007), pp.247-254.

[89] S.Ovtchinnikov, F.Dobrian, X-C.Cai, D-Keyes, 'Domain-decomposed fully coupled implicit methods for a magnetohydrodynamics',*Proceedings of the 17th International Conference on Domain Decomposition Methods*, (2007), pp.333-340.

[90] N. Pan and P. Gibson, *Thermal and moisture transport in fibrous materials.* (Woodhead Publishing Limited in association with Textile Institute and CRC Press, 2006).

[91] S.V.Patankar, *Numerical heat transfer and fluid flow.* (Hemisphere Publishing Corporation, 1980).

[92] G.Pavliotis and A.Stuart, *Multiscale Methods:Averaging and Homogenization.* (Springer, 2008).

[93] J. S. Przeminiecki, *Theory of Matrix Structural Analysis.* (McGraw-Hill, 1968).

[94] A.Quarteroni, 'Domain decomposition and parallel processing for the numerical solution of partial differential equations', *Surv. Math. Ind.*, (1991), pp.75-118.

[95] A.Quarteroni and A.Valli , *Domain Decomposition Methods for Partial Differential Equations.* (Oxford University Press, 1999).

[96] A.Quarteroni, A. Veneziani, and P.Zunino, 'A domain decomposition method for advection-diffusion processes with application to blood solutes',*SIAM J. Sci. Comput.*, 6 (2002), pp.1959-1980.

[97] P.A. Raviart, J.M. Thomas, 'Primal hybrid finite element methods for 2nd order elliptic equations', *Math. Comp.*, 31 (1977), pp. 391-413.

[98] C.Rey, F. Lene, 'Reuse of Krylov Spaces in the Solution of Large-scale Nonlinear Elasticity Problems', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.465-471.

[99] F.Risler and C.Rey, 'Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems', *Numerical Algorithms*, 23 (2000), pp.1-30.

[100] Ronquist, 'A domain decomposition solver for three-dimensional steady free surface flow', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.792-799.

[101] F-X.Roux, 'Parallel implementation of a domain decomposition method for non-linear elasticity problems',*Domain based parallelism and problem decomposition methods in computational science and engineering, D.Keyes, Y.Saad, D.Truhlar eds*, (1994). pp.161-175.

[102] Y.Saad, *Iterative Methods for Sparse Linear Systems.* (SIAM, 2003).

[103] M.Sala, 'An algebraic 2-level domain decomposition preconditioner with applications to the compressible Euler equations', *Internat.J.Numer.Methods in Fluids*, 40.12(2002), pp.1551-1560.

[104] A.A.Samarskii, *The Theory of Difference Schemes.* (Marcel Dekker Inc, 2001).

[105] H.A. Schwarz, 'Uber Einige Abbildungsaufgaben', *Gesammelte Mathematische Abhandlungen*, 11 (1869), pp. 65-83.

[106] J.R.Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain.* (School of Computer Science, Carnegie Mellon University, 1994).

[107] B.Smith, P.Bjorstad and W.Gropp. *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations.* (Cambridge University Press, 1996).

[108] D.B.Spalding, 'A General Purpose Computer Program for Multidimensional One and Two Phase Flow, *Mathematics and Computers in Simulation*, 13 (1981), pp.267-276.

[109] D.B.Spalding, *Four Lectures on the PHOENICS Code, Computational Fluid Dynamics Unit, Report, CFD/82/5.* (Imperial College of Science and Technology, London, UK, 1982).

[110] D.B.Spalding, *PHOENICS 1984. A Multi-dimensional Multi-phase General-Purpose Computer Simulator for Fluid Flow, Heat Transfer and Combustion, Computational Fluid Dynamics Unit, Report, CFD/84/18.* (Imperial College of Science and Technology, London, UK, 1984).

[111] H.L. Stone, 'Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations', *SIAM Journal of Numerical Analysis*, 5 (1968), pp.530538.

[112] X.-C. Tai, 'Domain decomposition for linear and nonlinear elliptic problems via function or space decomposition.', *Proc. of the 7th international conference on domain decomposition*, 35.4 (1995), pp.355-360.

[113] X.-C. Tai and M. Espedal, 'Rate of convergence of some space decomposition methods for linear and nonlinear problems.', *SIAM J. Numer. Anal.* 35.4 (1998), pp.1558-1570.

[114] J.C. Tannehill, D.A. Anderson, R.H. Pletcher, *Computational fluid mechanics and heat transfer.* (Taylor & Francis, 1997).

[115] L.Tartar,'Incompressible fluid flow in a porous medium convergence of the homogenization process', *Appendix to Lecture Notes in Physics*, 127 (1980). Springer-Velag, Berlin .

[116] L.H. Thomas, *Elliptic Problems in Linear Differential Equations over a Network*, (Watson Sci. Comput. Lab Report, Columbia University, New York, 1949).

[117] A.Toselli and O.Widlund , *Domain Decomposition Methods - Algorithms and Theory.* (Springer, 2005).

[118] R.S.Varga, *Matrix Iterative Analysis.* (Prentice-Hall, 1962).

[119] P.S. Vassilevski,'Poincar-Steklov operators for elliptic difference problems', *C. R. Acad. Bulgare Sci.*, 38 (1985), pp.543-546.

[120] H.Versieux, M.Sarkis, 'A three-scale finite element method for elliptic equations with rapidly oscillating periodic coefficients', *Proceedings of the 16th International Conference on Domain Decomposition Methods*, (2007), pp.765-772.

[121] H. Versteeg, W. Malalasekra, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* (Prentice Hall, 2007).

[122] Wang and Ersland, 'A characteristic domain decomposition algorithm for multi phase flows with interfaces', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.835-845.

[123] P.Wesseling, *An Introduction To Multigrid Methods.* (John Wiley and Sons, 1992).

[124] M.Wheeler, I.Yotov, 'Physical and computational domain decompositions for modeling subsurface flows', *Proceedings of the 10th International Conference on Domain Decomposition Methods*, (1998), pp.217-227.

[125] R.E. White *Computational Mathematics: Models, Methods and Analysis.* (CRC Press, 2003).

[126] C.Wieners, B.Wohlmuth, 'The coupling of mixed and conforming finite element discretizations', *Proceedings of the 10th International Conference on Domain Decomposition Methods*, (1998), pp.547-555.

[127] P.Wilders and G.Fotia, 'One-level Krylov-Schwarz domain decomposition for finite volume advection-diffusion ', *Proceedings of the 9th International Conference on Domain Decomposition Methods*, (1998), pp.558-565.

[128] Y.Wu, X-C.Cai, and D.Keyes, 'Additive Schwarz methods for hyperbolic equations', *Proceedings of the 10th International Conference on Domain Decomposition Methods*, (1998), pp.468-475.

[129] H.Wu, J.Fan, 'Study of heat and moisture transfer within multi-layer clothing assemblies consisting of different types of battings', *International Journal of Thermal Sciences*, 47 (2008), pp.641-647.

[130] L.Yi, L.Fengzhi, Z.Qingyong, 'Numerical simulation of virus diffusion in facemask during breathing cycles', *International Journal of Heat and Mass Transfer*, 48 (2005), pp.4229-4242.

[131] I. Yotov, 'Interface solvers and preconditioners of domain decomposition type for multiphase flow in multiblock porous media', *Advances in Computation: Theory and Practice*, 7 (2001), pp.157-167.

[132] I. Yotov, 'A multilevel Newton-Krylov interface solver for multiphysics couplings of flow in porous media', *Numer. Linear Algebra Appl.*, 8 (2001), pp.551–570.

[133] http://www.cham.co.uk/phoenics/d_polis/d_lecs/numerics/solution.htm.

[134] http://www.netlib.org/linalg/htmltemplates/node121.html.

[135] http://www.netlib.org/lapack/lug/node27.html.