Spring 2020

Decoupled finite element methods for general steady two-dimensional Boussinesq equations

Lioba Boveleth

Department:

DECOUPLED FINITE ELEMENT METHODS FOR GENERAL STEADY TWO-DIMENSIONAL BOUSSINESQ EQUATIONS

by

LIOBA BOVELETH

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

APPLIED MATHEMATICS

2020

Approved by

Dr. Xiaoming He, Advisor
Dr. Daozhi Han
Dr. John Singler

ABSTRACT

This work presents two kinds of decoupled finite element methods for the steady natural convection problem in two dimensions. Firstly, the standard Galerkin finite element method is derived in detail stating algorithms needed for the realization in `MATLAB`. A numerical example verifies the error convergence. Secondly, using iteration, the Boussinesq equations are decoupled into the Navier-Stokes equations and a parabolic problem. The resulting problems are solved either in parallel or sequentially. Finally, the same numerical example as before is used to confirm the convergence and analyze the methods in terms of iteration performance. In addition to a higher flexibility and the convenience of exploiting existing solvers, the new decoupled finite element methods can be realized with less iteration steps, and thus more efficiently, if the focus is only on some of the unknowns or more information is provided.

ACKNOWLEDGMENTS

First, I would like to thank my advisor Dr. Xiaoming He for his continuous support of my studies. It was a pleasure working with him as he always found the time to help me with his vast knowledge.

I would like to express my gratitude to Prof. Hans-Joachim Zwiesler from Ulm University and Dr. Martin Bohner for granting me the opportunity to study at Missouri University of Science and Technology.

Last but not least, I would like to thank my family and Sebastian for their unconditional encouragement and inspiration throughout my years of study and while writing this thesis. Thank you.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

NOMENCLATURE

Symbol	Description
	Roman
h	Edge size of the finite element mesh
K	Maximum number of decoupling iterations
k	Thermal conductivity parameter
L	Maximum number of Newton iterations
p	Pressure of the fluid
Pr	Prandtl number
Ra	Rayleigh number
$\boldsymbol{f_u}$	Forcing function for Navier-Stokes equations
$\boldsymbol{g_u}$	Boundary function for the velocity
$\boldsymbol{u}$	Velocity of the fluid
$\mathbb{D}$	Deformation tensor
$\mathbb{I}$	Unit matrix
$\mathbb{R}$	The real numbers
$\mathbb{T}$	Stress tensor
f_θ	Forcing function for convection-diffusion equation

g_θ	Boundary function for the temperature
N	Number of mesh elements
N_b	Number of finite element basis functions
N_{lb}	Number of local finite element nodes
P_b	Finite element information matrix for the nodes
T_b	Finite element information matrix for the elements

Greek

Δ	Laplacian operator
∇	Gradient operator
$\nabla\cdot$	Divergence operator
ν	Viscosity
Ω	Domain
$\partial\Omega$	Boundary of the domain
θ	Temperature of the fluid

1. INTRODUCTION

A typical everyday example of convection are the mechanics of a hot air balloon. A heater is used to heat the air inside the balloon, which causes the air to move upwards. Since the hot air is trapped inside the envelope, the whole balloon moves upwards. Thus, when the pilot wants to rise or descend, he heats the air more or releases some of the hot air, respectively. In general, the movement of the fluid (the air) is caused by the temperature difference. Since the pilot uses a heater to influence the air's temperature, it is referred to as forced convection. On the other hand, free or natural convection is when the fluid motion is generated by a gravitational field (see [12], for instance). Gravitation itself is not sufficient to set a fluid, either liquid or gas, in motion because fluid density change is required as well. The most common cause for density variation in natural convection is a change in temperature. As a different example, density variations occur also in the ocean when salt and fresh water come together since salt water is heavier than fresh water. In nature, free convection flows are observed in different situations, such as geophysics, weather, and ocean movement. In engineering, free convection is exploited in numerous applications including double-glazed windows, cooling in small electronic devices, building insulation and environmental transport problems (see [6]).

Assuming the density of the fluid is constant and the gravitational force depends on the temperature, natural convection can be modeled by the Boussinesq equations (see Equations (2.1)). In this approximation, the fluid and temperature are coupled by two terms. First, the buoyancy term is a source term that depends linearly on temperature and acts in the direction opposite to gravity. This is added to the stationary incompressible Navier–Stokes equations for the fluid variables. The second term is the convective term, which is based on the velocity of the fluid in the convection-diffusion equation for the temperature variable.

The modelling of natural convection is not the only purpose of solving the Boussinesq equations. In applied analysis, the three-dimensional Navier-Stokes equations are still not entirely explored as understanding the vortex stretching effect in three-dimensional flows remains a difficulty. Since the two-dimensional Boussinesq equations are analogous to the three-dimensional Navier–Stokes equations for axisymmetric swirling flow, their understanding contributes towards the extensive understanding of the vortex stretching effect in three-dimensional flows (see [21]).

This work focuses on solving the two-dimensional stationary Boussinesq equations using finite element methods. In Section 2, a literature review is performed to get the introduced methods in line with already conducted research. In Section 3, the standard Galerkin finite element method is recalled and derived at length. Then, algorithms for the implementation of the method in `MATLAB` are introduced, and the method is verified by a numerical example. In Section 4, the idea of decoupling the standard Galerkin finite element method is realized in two different ways. Keeping the same structure as before, the derivation is discussed first, followed by used algorithms, and the solution of a numerical example. Finally, the derived finite element methods are evaluated in terms of number of iterations.

2. LITERATURE REVIEW

The Navier-Stokes equations form a well-known equation system, whose finite element solution was already studied in the early 1970's (see [17] and the references therein). To the author's knowledge, the study of numerical methods for approximating the solution of Boussinesq approximations of the Navier-Stokes system began in 1980 [10] and significantly increased in recent years [1, 4, 5, 6, 22]. The interested reader may find the derivation of the Boussinesq equations in [7]. Recall the steady two dimensional natural convection equations in their typical form

$$\begin{cases} -Pr\Delta \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = PrRa\theta \boldsymbol{e_2} + \boldsymbol{f_u}, & \text{in } \Omega, \\ \nabla\cdot\boldsymbol{u} = \boldsymbol{0}, & \text{in } \Omega, \\ -k\Delta\theta + \boldsymbol{u}\cdot\nabla\theta = f_\theta, & \text{in } \Omega, \end{cases} \tag{2.1}$$

from [20], $\Omega \subset \mathbb{R}^2$ is a bounded domain with given Dirichlet boundary conditions on its Lipschitz continuous boundary $\partial\Omega$, $\boldsymbol{u} = (u_1, u_2)^T$ is the fluid velocity, p the pressure and θ the temperature. Furthermore, $\boldsymbol{f_u}$ and f_θ are given forcing functions, Pr and Ra refer to the Prandtl and Rayleigh numbers, respectively, and $k > 0$ is the thermal conductivity parameter. The term $\theta\boldsymbol{e_2}$ represents the buoyancy force. Throughout this work, vector valued functions are denoted by boldface. The symbols Δ, ∇ and $\nabla\cdot$ stand for the Laplacian, gradient and divergence operators, respectively.

In addition to the velocity and the pressure, the steady Boussinesq Equations (2.1) include the temperature field, making it even harder to find the numerical solution. First attempts to find efficient numerical schemes to solve (2.1) were coupled finite element methods like the standard Galerkin finite element method [5], the least squared finite element method [14], and the projection-based stabilized mixed finite element method [22].

That means, those methods generated large systems to solve for the velocity $\boldsymbol{u}$, the pressure p and the temperature θ simultaneously. Furthermore, the systems were not only large but also nonlinear, thus additional iterations were needed, making it expensive to find the numerical solution, in general.

Even though there were techniques to treat the nonlinear problem more efficiently [8], the idea of the decoupled technique arose [16]. By decoupling, the considered problem can be separated into a series of sub-problems, where each sub-problem includes less numerical components. The decoupled algorithm exploits existing computing resources, can be used in parallelism and is more flexible in terms of each variable. In general, there are two types of the decoupled method. The first one is based on the multigrid method [11, 20] and the second one on the numerical approximations of the previous time level [19]. Multigrid methods use several different mesh sizes. To give an example, one small nonlinear system on a coarse mesh and a few large linearized systems on a fine mesh are solved in [11]. The second type is used for time-dependent natural convection equations. As in [19], the decoupling can be based on the backward Euler scheme. The solutions of the previous time step are used to solve a linear parabolic equation for the temperature and linear Navier-Stokes equations for the velocity and the pressure in parallel.

While the progress in multigrid methods continues focussing on the time-dependent problem [18] and new approaches especially for temperature-dependent parameters are introduced [2, 3, 15], the purpose of this work is to diminish the gap between the two general types of the decoupled method. We start by solving the coupled stationary natural convection problem using the standard Galerkin finite element method with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ. A study of the existence and uniqueness of a solution can be found in [5]. To linearize the nonlinear terms of the Galerkin formulation, a Newton iteration is used. Based on the first method, we derive a new numerical scheme, where the original problem is decoupled

into the Navier-Stokes equations and a parabolic problem. We iterate over the two problems and solve them either in parallel or sequentially using the numerical approximation of the previous iteration step.

3. COUPLED FINITE ELEMENT METHOD

The goal of this section is to derive the formulations necessary for the coupled Galerkin finite element method to approximate the solution of the Boussinesq equations as a foundation for the decoupled method for the same equations. To realize this goal in a comprehensible and straight forward way, we do not work with Equations (2.1) directly, but consider the more general and non-dimensional equations

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla \cdot \mathbb{T}(\boldsymbol{u}, p) - \boldsymbol{e_2} c_1 \theta = \boldsymbol{f_u}, \quad \text{in } \Omega, \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad \text{in } \Omega, \tag{3.2}$$

$$\boldsymbol{u} \cdot \nabla \theta - \nabla \cdot (c_2 \nabla \theta) = f_\theta, \quad \text{in } \Omega, \tag{3.3}$$

with vector functions $\boldsymbol{u}(x, y) = (u_1(x, y), u_2(x, y))^T$ and $\boldsymbol{f_u}(x, y) = (f_1(x, y), f_2(x, y))^T$ and vector $\boldsymbol{e_2} = (0, 1)^T$. Furthermore, the stress tensor $\mathbb{T}(\boldsymbol{u}, p)$ is defined as

$$\mathbb{T}(\boldsymbol{u}, p) = 2\nu \mathbb{D}(\boldsymbol{u}) - p\mathbb{I},$$

where ν is the viscosity and $\mathbb{D}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)$ the deformation tensor. The derivation may be found in [13]. We consider the boundary condition

$$\boldsymbol{u} = \boldsymbol{g_u}, \quad \text{on } \partial\Omega \quad \text{and}$$

$$\theta = g_\theta, \quad \text{on } \partial\Omega$$

for given functions $\boldsymbol{g_u} = (g_1, g_2) : \mathbb{R}^2 \to \mathbb{R}^2$ and $g_\theta : \mathbb{R}^2 \to \mathbb{R}$. As stated in [10], an application of boundary conditions for the pressure p is in general not consistent with Equation (3.1). In this case, p is defined up to an arbitrary additive constant. Thus, we will fix p at one point in the domain Ω.

3.1. DERIVATION OF THE METHOD

This section consists of four parts that build on one another and eventually state the desired linear algebraic system we aim to solve. The fifth part then presents algorithms used for the realization in `MATLAB`.

3.1.1. Weak Formulation. To derive the weak formulation, we will deal with Equations (3.1) to (3.3) separately and sequentially. As the first step, we take the inner product with a vector function $\boldsymbol{v}(x, y) = (v_1, v_2)^T$ on both sides of the Navier-Stokes Equation (3.1):

$$\begin{aligned}
& (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nabla \cdot \mathbb{T}(\boldsymbol{u}, p) - \boldsymbol{e_2} c_1 \theta = \boldsymbol{f_u} \\
\Rightarrow \quad & (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} - (\nabla \cdot \mathbb{T}(\boldsymbol{u}, p)) \cdot \boldsymbol{v} - \boldsymbol{e_2} c_1 \theta \cdot \boldsymbol{v} = \boldsymbol{f_u} \cdot \boldsymbol{v} \\
\Rightarrow \quad & \int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} dxdy - \int_\Omega (\nabla \cdot \mathbb{T}(\boldsymbol{u}, p)) \cdot \boldsymbol{v} dxdy - \int_\Omega \boldsymbol{e_2} c_1 \theta \cdot \boldsymbol{v} dxdy = \int_\Omega \boldsymbol{f_u} \cdot \boldsymbol{v} dxdy.
\end{aligned}$$

Second, the divergence free Equation (3.2) is multiplied by a function $q(x, y)$:

$$\begin{aligned}
& \nabla \cdot \boldsymbol{u} = 0 \\
\Rightarrow \quad & (\nabla \cdot \boldsymbol{u}) q = 0 \\
\Rightarrow \quad & \int_\Omega (\nabla \cdot \boldsymbol{u}) q dxdy = 0.
\end{aligned}$$

Third, we multiply the convection-diffusion Equation (3.3) for the temperature variable by a function $w(x, y)$:

$$\begin{aligned}
& \boldsymbol{u} \cdot \nabla\theta - \nabla \cdot (c_2 \nabla\theta) = f_\theta \\
\Rightarrow \quad & (\boldsymbol{u} \cdot \nabla\theta) w - \nabla \cdot (c_2 \nabla\theta) w = f_\theta w \\
\Rightarrow \quad & \int_\Omega (\boldsymbol{u} \cdot \nabla\theta) w dxdy - \int_\Omega \nabla \cdot (c_2 \nabla\theta) w dxdy = \int_\Omega f_\theta w dxdy.
\end{aligned}$$

Here, $\boldsymbol{u}(x,y)$, $p(x,y)$ and $\theta(x,y)$ are called trial functions and $\boldsymbol{v}(x,y)$, $q(x,y)$ and $w(x,y)$ are called test functions. Using integration by parts in multi-dimensions yields

$$\int_\Omega (\nabla \cdot \mathbb{T}) \cdot \boldsymbol{v} dxdy = \int_{\partial\Omega} (\mathbb{T}\boldsymbol{n}) \cdot \boldsymbol{v} dS - \int_\Omega \mathbb{T} : \nabla \boldsymbol{v} dxdy,$$

where $\boldsymbol{n} = (n_1, n_2)^T$ is the unit outer normal vector of $\partial\Omega$ and the operator : denotes the matrix inner product. Thus,

$$\int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} dxdy + \int_\Omega \mathbb{T}(\boldsymbol{u}, p) : \nabla \boldsymbol{v} dxdy - \int_{\partial\Omega} (\mathbb{T}(\boldsymbol{u}, p)\boldsymbol{n}) \cdot \boldsymbol{v} dS - \int_\Omega \boldsymbol{e_2} c_1 \cdot \boldsymbol{v} dxdy = \int_\Omega \boldsymbol{f_u} \cdot \boldsymbol{v} dxdy$$

is obtained. First, we observe that

$$\mathbb{T}(\boldsymbol{u}, p) : \nabla \boldsymbol{v} = 2\nu \mathbb{D}(\boldsymbol{u}) : \mathbb{D}(\boldsymbol{v}) - p(\nabla \cdot \boldsymbol{v}).$$

Second, since the solution on $\partial\Omega$ is given by $\boldsymbol{u} = \boldsymbol{g_u}$, we choose the test function $\boldsymbol{v}$ such that $\boldsymbol{v} = \boldsymbol{0}$ on $\partial\Omega$. Hence, we obtain

$$\int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} dxdy + \int_\Omega 2\nu \mathbb{D}(\boldsymbol{u}) : \mathbb{D}(\boldsymbol{v}) dxdy - \int_\Omega p(\nabla \cdot \boldsymbol{v}) dxdy - \int_\Omega \boldsymbol{e_2} c_1 \theta \cdot \boldsymbol{v} dxdy = \int_\Omega \boldsymbol{f_u} \cdot \boldsymbol{v} dxdy.$$

On the other hand, we use Green's formula

$$\int_\Omega \nabla \cdot (c_2 \nabla\theta) w dxdy = \int_{\partial\Omega} (c_2 \nabla\theta \cdot \boldsymbol{n}) w dS - \int_\Omega c_2 \nabla\theta \cdot \nabla w dxdy,$$

to acquire

$$\int_\Omega (\boldsymbol{u} \cdot \nabla\theta) w dxdy + \int_\Omega c_2 \nabla\theta \cdot \nabla w dxdy = \int_{\partial\Omega} (c_2 \nabla\theta \cdot \boldsymbol{n}) w dS + \int_\Omega f_\theta w dxdy$$

for the convection-diffusion equation. Since the solution on $\partial\Omega$ is given by $\theta = g_\theta$, we choose the test function w such that $w = 0$ on $\partial\Omega$:

$$\int_\Omega (\boldsymbol{u} \cdot \nabla\theta) w dxdy + \int_\Omega c_2 \nabla\theta \cdot \nabla w dxdy = \int_\Omega f_\theta w dxdy.$$

At this point, we introduce the frequently used (see for example [20]) Sobolev spaces

$$\begin{aligned}
L^2(\Omega) &= \left\{ v : \Omega \to \mathbb{R} : \int_\Omega |v|^2 < \infty \right\}, \\
H^1(\Omega) &= \left\{ v \in L^2(\Omega) : \frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \in L^2(\Omega) \quad \forall \alpha_1 + \alpha_2 = 1 \right\}, \quad \text{and} \\
H_0^1(\Omega) &= \left\{ v \in H^1(\Omega) : v_{|\partial\Omega} = 0 \right\}.
\end{aligned}$$

In conclusion, we derived the following weak formulation: Find $\boldsymbol{u} \in \left(H^1(\Omega)\right)^2$, $p \in L^2(\Omega)$ and $\theta \in H^1(\Omega)$ such that

$$\begin{aligned}
c(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) + a(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) - (\boldsymbol{e_2} c_1 \theta, \boldsymbol{v}) &= (\boldsymbol{f_u}, \boldsymbol{v}), \\
b(\boldsymbol{u}, q) &= 0, \\
\tilde{c}(\boldsymbol{u}, \theta, w) + \tilde{a}(\theta, w) &= (f_\theta, w),
\end{aligned}$$

for any $\boldsymbol{v} \in \left(H_0^1(\Omega)\right)^2$, $q \in L^2(\Omega)$ and $w \in H_0^1(\Omega)$ with

$$\begin{aligned}
c(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega (\boldsymbol{w} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} dxdy, \\
a(\boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega 2\nu \mathbb{D}(\boldsymbol{u}) : \mathbb{D}(\boldsymbol{v}) dxdy, \\
b(\boldsymbol{u}, q) &= -\int_\Omega (\nabla \cdot \boldsymbol{u}) q dxdy, \\
(\boldsymbol{f}, \boldsymbol{v}) &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} dxdy \quad \text{or} \quad (f, w) = \int_\Omega f w dxdy, \\
\tilde{a}(\theta, w) &= \int_\Omega c_2 \nabla\theta \cdot \nabla w dxdy, \quad \text{and} \\
\tilde{c}(\boldsymbol{u}, \theta, w) &= \int_\Omega (\boldsymbol{u} \cdot \nabla\theta) w dxdy.
\end{aligned}$$

To be more precise, the operators can be written in the following detailed way.

$$
\begin{aligned}
c(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) &= \int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}dxdy \\
&= \int_\Omega u_1\frac{\partial u_1}{\partial x}v_1 + u_2\frac{\partial u_1}{\partial y}v_1 + u_1\frac{\partial u_2}{\partial y}v_2 + u_2\frac{\partial u_2}{\partial x}v_2 + u_2\frac{\partial u_2}{\partial y}v_2 dxdy \\
a(\boldsymbol{u},\boldsymbol{v}) &= \int_\Omega 2\nu\mathbb{D}(\boldsymbol{u}):\mathbb{D}(\boldsymbol{v})dxdy \\
&= \int_\Omega \nu\left(2\frac{\partial u_1}{\partial x}\frac{\partial v_1}{\partial x} + 2\frac{\partial u_2}{\partial y}\frac{\partial v_2}{\partial y} + \frac{\partial u_1}{\partial u}\frac{\partial v_1}{\partial y} + \frac{\partial u_1}{\partial y}\frac{\partial v_2}{\partial x} + \frac{\partial u_2}{\partial x}\frac{\partial v_1}{\partial y} + \frac{\partial u_2}{\partial x}\frac{\partial v_2}{\partial x}\right)dxdy \\
b(\boldsymbol{v},p) &= -\int_\Omega (\nabla\cdot\boldsymbol{u})qdxdy = -\int_\Omega p\frac{\partial v_1}{\partial x} + p\frac{\partial v_2}{\partial y}dxdy \\
(\boldsymbol{e_2}c_1\theta,\boldsymbol{v}) &= \int_\Omega c_1\theta v_2 dxdy \\
(\boldsymbol{f_u},\boldsymbol{v}) &= \int_\Omega \boldsymbol{f_u}\cdot\boldsymbol{v}dxdy = \int_\Omega f_1v_1 + f_2v_2 dxdy \\
b(\boldsymbol{u},q) &= -\int_\Omega (\nabla\cdot\boldsymbol{u})qdxdy = -\int_\Omega \frac{\partial u_1}{\partial x}q + \frac{\partial u_2}{\partial y}qdxdy \\
\tilde{c}(\boldsymbol{u},\theta,w) &= \int_\Omega (\boldsymbol{u}\cdot\nabla\theta)wdxdy = \int_\Omega u_1\frac{\partial\theta}{\partial x}w + u_2\frac{\partial\theta}{\partial y}wdxdy \\
\tilde{a}(\theta,w) &= \int_\Omega c_2\nabla\theta\cdot\nabla w dxdy = \int_\Omega c_2\left(\frac{\partial\theta}{\partial x}\frac{\partial w}{\partial x} + \frac{\partial\theta}{\partial y}\frac{\partial w}{\partial y}\right)dxdy \\
(f_\theta,w) &= \int_\Omega f_\theta w dxdy
\end{aligned}
$$

3.1.2. Galerkin Formulation. Based on the weak formulation, the Galerkin formulation is obtained by defining finite element spaces. We consider a finite element space $U_h \subset H^1(\Omega)$ for the velocity functions u_1 and u_2, a finite element space $P_h \subset L^2(\Omega)$ for the pressure p and a finite element space $W_h \subset H^1(\Omega)$ for the temperature θ. Since we will deal with the Dirichlet boundary conditions later for an easier implementation, this leads to the following Galerkin formulation: Find $\boldsymbol{u_h} \in (U_h)^2$, $p_h \in P_h$ and $\theta_h \in W_h$ such that

$$c(\boldsymbol{u_h},\boldsymbol{u_h},\boldsymbol{v_h}) + a(\boldsymbol{u_h},\boldsymbol{v_h}) + b(\boldsymbol{v_h},p_h) - (\boldsymbol{e_2}c_1\theta_h,\boldsymbol{v_h}) = (\boldsymbol{f_u},\boldsymbol{v_h}), \tag{3.4}$$

$$b(\boldsymbol{u_h},q_h) = 0, \tag{3.5}$$

$$\tilde{c}(\boldsymbol{u_h},\theta_h,w_h) + \tilde{a}(\theta_h,w_h) = (f_\theta,w_h), \tag{3.6}$$

for any $\boldsymbol{v_h} \in (U_h)^2$, $q_h \in P_h$ and $w_h \in W_h$.

In our numerical example, $U_h = span\{\varphi_j\}_{j=1}^{N_{bu}}$, $W_h = span\{\psi_j\}_{j=1}^{N_{b\theta}}$ and $P_h = span\{\zeta_j\}_{j=1}^{N_{bp}}$ are chosen to be the finite element spaces with the quadratic global basis functions $\{\varphi_j\}_{j=1}^{N_{bu}}$, $\{\psi_j\}_{j=1}^{N_{b\theta}}$ and linear global basis functions $\{\zeta_j\}_{j=1}^{N_{bp}}$, which are defined in the traditional way (see [9]).

3.1.3. Newton Iteration. Before we are able to obtain the linear system, we have to linearize the nonlinear terms in Equations (3.4) and (3.6). Therefore, we introduce a Newton iteration for the Galerkin formulation as follows. This common idea was already applied in [9].

Initial guess: $\boldsymbol{u^{(0)}}$ and $\theta^{(0)}$

For $l = 1, 2, ..., L$ find $\boldsymbol{u_h^{(l)}} \in (U_h)^2$, $p_h^{(l)} \in P_h$ and $\theta_h^{(l)} \in W_h$ such that

$$c(\boldsymbol{u_h^{(l)}}, \boldsymbol{u_h^{(l-1)}}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(l-1)}}, \boldsymbol{u_h^{(l)}}, \boldsymbol{v_h}) + a(\boldsymbol{u_h^{(l)}}, \boldsymbol{v_h}) + b(\boldsymbol{v_h}, p_h^{(l)}) - (\boldsymbol{e_2} c_1 \theta_h^{(l)}, \boldsymbol{v_h})$$
$$= (\boldsymbol{f_u}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(l-1)}}, \boldsymbol{u_h^{(l-1)}}, \boldsymbol{v_h}),$$
$$b(\boldsymbol{u_h}^{(l)}, q_h) = 0,$$
$$\tilde{c}(\boldsymbol{u_h^{(l)}}, \theta_h^{(l-1)}, w_h) + \tilde{c}(\boldsymbol{u_h^{(l-1)}}, \theta_h^{(l)}, w_h) + \tilde{a}(\theta_h^{(l)}, w_h) = (f_\theta, w_h) + \tilde{c}(\boldsymbol{u_h^{(l-1)}}, \theta_h^{(l-1)}, w_h),$$

for any $\boldsymbol{v_h} \in (U_h)^2$, $q_h \in P_h$ and $w_h \in W_h$.

3.1.4. Finite Element Discretization. Considering $u_{1h}^{(l)}, u_{2h}^{(l)} \in U_h = span\{\varphi_j\}_{j=1}^{N_{bu}}$, $p_h^{(l)} \in P_h = span\{\zeta_j\}_{j=1}^{N_{bp}}$ and $\theta_h^{(l)} \in W_h = span\{\psi_j\}_{j=1}^{N_{b\theta}}$, there exist coefficients $u_{1h}^{(l)}$, $u_{2h}^{(l)}$ $(j = 1, ..., N_{bu})$, $p_h^{(l)}$ $(j = 1, ..., N_{bp})$ and $\theta_h^{(l)}$ $(j = 1, ..., N_{b\theta})$ such that

$$u_{1h}^{(l)} = \sum_{j=1}^{N_{bu}} u_{1j}^{(l)} \varphi_j, \quad u_{2h}^{(l)} = \sum_{j=1}^{N_{bu}} u_{2j}^{(l)} \varphi_j, \quad p_h^{(l)} = \sum_{j=1}^{N_{bp}} p_j^{(l)} \zeta_j, \quad \text{and} \quad \theta_h^{(l)} = \sum_{j=1}^{N_{b\theta}} \theta_j^{(l)} \psi_j. \quad (3.7)$$

Thus, to approximate the velocity $\boldsymbol{u}$, the pressure p and the temperature θ, we derive a linear algebraic system to obtain those coefficients. For the first Equation (3.4) of the Galerkin formulation, we choose $\boldsymbol{v_h} = (\varphi_i, 0)^T$ $(i = 1, ..., N_{bu})$ and $\boldsymbol{v_h} = (0, \varphi_i)^T$ $(i = 1, ..., N_{bu})$,

which are linear independent. In other words, we choose $v_{1h} = \varphi_i$ and $v_{2h} = 0$ in the first set of test functions and $v_{1h} = 0$ and $v_{2h} = \varphi_i$ in the second set of test functions. For the second Equation (3.5) of the Galerkin formulation, the test function $q_h = \zeta_i$ $(i = 1, ..., N_{bp})$ is chosen. Finally, we select $w_h = \psi_i$ $(i = 1, ..., N_{b\theta})$ for the third Equation (3.6) of the Galerkin formulation. Then, at the step l $(l = 1, 2, ..., L)$ of the Newton iteration, the following four equations hold.

$$\begin{aligned}
&\int_\Omega \frac{\partial u_{1h}^{(l-1)}}{\partial x}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\varphi_j\right)\varphi_i dxdy + \int_\Omega \frac{\partial u_{1h}^{(l-1)}}{\partial y}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\varphi_j\right)\varphi_i dxdy \\
&+ \int_\Omega u_{1h}^{(l-1)}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\varphi_i dxdy + \int_\Omega u_{2h}^{(l-1)}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\varphi_i dxdy \\
&+ 2\int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\frac{\partial\varphi_i}{\partial x}dxdy + \int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\frac{\partial\varphi_i}{\partial y}dxdy \\
&+ \int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\frac{\partial\varphi_i}{\partial y}dxdy - \int_\Omega \nu\left(\sum_{j=1}^{N_{bp}} p_j^{(l)}\zeta_j\right)\frac{\partial\varphi_i}{\partial x}dxdy \\
&= \int_\Omega f_1\varphi_j dxdy + \int_\Omega u_{1h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial x}\varphi_i dxdy + \int_\Omega u_{2h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial y}\varphi_i dxdy
\end{aligned}$$

$$\begin{aligned}
&\int_\Omega \frac{\partial u_{2h}^{(l-1)}}{\partial x}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\varphi_j\right)\varphi_i dxdy + \int_\Omega \frac{\partial u_{2h}^{(l-1)}}{\partial y}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\varphi_j\right)\varphi_i dxdy \\
&+ \int_\Omega u_{1h}^{(l-1)}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\varphi_i dxdy + \int_\Omega u_{2h}^{(l-1)}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\varphi_i dxdy \\
&+ 2\int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\frac{\partial\varphi_i}{\partial y}dxdy + \int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\frac{\partial\varphi_i}{\partial x}dxdy \\
&+ \int_\Omega \nu\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\frac{\partial\varphi_i}{\partial x}dxdy - \int_\Omega \nu\left(\sum_{j=1}^{N_{bp}} p_j^{(l)}\zeta_j\right)\frac{\partial\varphi_i}{\partial y}dxdy - \int_\Omega c_1\left(\sum_{j=1}^{N_{b\theta}} \theta_j^{(l)}\psi_j\right)\psi_i dxdy \\
&= \int_\Omega f_2\varphi_j dxdy + \int_\Omega u_{1h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial x}\varphi_i dxdy + \int_\Omega u_{2h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial y}\varphi_i dxdy
\end{aligned}$$

$$-\int_{\Omega}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\frac{\partial\varphi_j}{\partial x}\right)\zeta_i dxdy-\int_{\Omega}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\frac{\partial\varphi_j}{\partial y}\right)\zeta_i dxdy=0$$

$$\begin{aligned}
&\int_{\Omega}\frac{\partial\theta_h^{(l-1)}}{\partial x}\left(\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\varphi_j\right)\psi_i dxdy+\int_{\Omega}\frac{\partial\theta_h^{(l-1)}}{\partial y}\left(\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\varphi_j\right)\psi_i dxdy\\
&+\int_{\Omega}u_{1h}^{(l-1)}\left(\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\frac{\partial\psi_j}{\partial x}\right)\psi_i dxdy+\int_{\Omega}u_{2h}^{(l-1)}\left(\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\frac{\partial\psi_j}{\partial y}\right)\psi_i dxdy\\
&+\int_{\Omega}c_2\left(\left(\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\frac{\partial\psi_j}{\partial x}\right)\frac{\partial\psi_i}{\partial x}+\left(\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\frac{\partial\psi_j}{\partial y}\right)\frac{\partial\psi_i}{\partial y}\right)dxdy\\
&=\int_{\Omega}f_\theta\psi_j dxdy+\int_{\Omega}u_{1h}^{(l-1)}\frac{\partial\theta_h^{(l-1)}}{\partial x}\psi_i dxdy+\int_{\Omega}u_{2h}^{(l-1)}\frac{\partial\theta_h^{(l-1)}}{\partial y}\psi_i dxdy
\end{aligned}$$

In the ensuing step, those equations are presented in a simplified form.

$$\begin{aligned}
&\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\left(\int_{\Omega}\frac{\partial u_{1h}^{(l-1)}}{\partial x}\varphi_j\varphi_i dxdy+\int_{\Omega}u_{1h}^{(l-1)}\frac{\partial\varphi_j}{\partial x}\varphi_i dxdy+\int_{\Omega}u_{2h}^{(l-1)}\frac{\partial\varphi_j}{\partial y}\varphi_i dxdy\right.\\
&\qquad\left.+2\int_{\Omega}\nu\frac{\partial\varphi_j}{\partial x}\frac{\partial\varphi_i}{\partial x}dxdy+\int_{\Omega}\nu\frac{\partial\varphi_j}{\partial y}\frac{\partial\varphi_i}{\partial y}dxdy\right)\\
&+\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\left(\int_{\Omega}\frac{\partial u_{1h}^{(l-1)}}{\partial y}\varphi_j\varphi_i dxdy+\int_{\Omega}\nu\frac{\partial\varphi_j}{\partial x}\frac{\partial\varphi_i}{\partial y}dxdy\right)\\
&+\sum_{j=1}^{N_{bp}} p_j^{(l)}\left(-\int_{\Omega}\zeta_j\frac{\partial\varphi_i}{\partial x}dxdy\right)\\
&=\int_{\Omega}f_1\varphi_j dxdy+\int_{\Omega}u_{1h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial x}\varphi_i dxdy+\int_{\Omega}u_{2h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial y}\varphi_i dxdy
\end{aligned}$$

$$\begin{aligned}
&\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\left(\int_{\Omega}\frac{\partial u_{2h}^{(l-1)}}{\partial x}\varphi_j\varphi_i dxdy+\int_{\Omega}\nu\frac{\partial\varphi_j}{\partial y}\frac{\partial\varphi_i}{\partial x}dxdy\right)\\
&+\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\left(\int_{\Omega}\frac{\partial u_{2h}^{(l-1)}}{\partial y}\varphi_j\varphi_i dxdy+\int_{\Omega}u_{1h}^{(l-1)}\frac{\partial\varphi_j}{\partial x}\varphi_i dxdy+\int_{\Omega}u_{2h}^{(l-1)}\frac{\partial\varphi_j}{\partial y}\varphi_i dxdy\right.
\end{aligned}$$

$$+2\int_\Omega \nu\frac{\partial\varphi_j}{\partial y}\frac{\partial\varphi_i}{\partial y}dxdy+\int_\Omega \nu\frac{\partial\varphi_j}{\partial x}\frac{\partial\varphi_i}{\partial x}dxdy\Bigg)$$
$$+\sum_{j=1}^{N_{bp}} p_j^{(l)}\left(-\int_\Omega \zeta_j\frac{\partial\varphi_i}{\partial y}dxdy\right)+\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\left(-\int_\Omega c_1\psi_j\varphi_i dxdy\right)$$
$$=\int_\Omega f_2\varphi_j dxdy+\int_\Omega u_{1h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial x}\varphi_i dxdy+\int_\Omega u_{2h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial y}\varphi_i dxdy$$

$$\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\left(-\int_\Omega \frac{\partial\varphi_j}{\partial x}\zeta_i dxdy\right)+\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\left(-\int_\Omega \frac{\partial\varphi_j}{\partial y}\zeta_i dxdy\right)=0$$

$$\sum_{j=1}^{N_{bu}} u_{1j}^{(l)}\left(\int_\Omega \frac{\partial\theta_h^{(l-1)}}{\partial x}\varphi_j\psi_i dxdy\right)+\sum_{j=1}^{N_{bu}} u_{2j}^{(l)}\left(\int_\Omega \frac{\partial\theta_h^{(l-1)}}{\partial y}\varphi_j\psi_i dxdy\right)$$
$$+\sum_{j=1}^{N_{b\theta}}\theta_j^{(l)}\left(\int_\Omega u_{1h}^{(l-1)}\frac{\partial\psi_j}{\partial x}\psi_i dxdy+\int_\Omega u_{2h}^{(l-1)}\frac{\partial\psi_j}{\partial y}\psi_i dxdy\right.$$
$$\left.+\int_\Omega c_2\frac{\partial\psi_j}{\partial x}\frac{\partial\psi_j}{\partial x}dxdy+\int_\Omega c_2\frac{\partial\psi_j}{\partial y}\frac{\partial\psi_j}{\partial y}dxdy\right)$$
$$=\int_\Omega f_\theta\psi_j dxdy+\int_\Omega u_{1h}^{(l-1)}\frac{\partial\theta_h^{(l-1)}}{\partial x}\psi_i dxdy+\int_\Omega u_{2h}^{(l-1)}\frac{\partial\theta_h^{(l-1)}}{\partial y}\psi_i dxdy$$

Finally, it is evident how to define the matrices and vectors to set up the necessary linear algebraic system. We start with the definitions of

$$A_1=\left[\int_\Omega \nu\frac{\partial\varphi_j}{\partial x}\frac{\partial\varphi_i}{\partial x}dxdy\right]_{i,j=1}^{N_{bu}},\qquad A_2=\left[\int_\Omega \nu\frac{\partial\varphi_j}{\partial y}\frac{\partial\varphi_i}{\partial y}dxdy\right]_{i,j=1}^{N_{bu}},$$
$$A_3=\left[\int_\Omega \nu\frac{\partial\varphi_j}{\partial x}\frac{\partial\varphi_i}{\partial y}dxdy\right]_{i,j=1}^{N_{bu}},\qquad A_4=\left[\int_\Omega \nu\frac{\partial\varphi_j}{\partial y}\frac{\partial\varphi_i}{\partial x}dxdy\right]_{i,j=1}^{N_{bu}},$$
$$A_5=\left[\int_\Omega -\zeta_j\frac{\partial\varphi_i}{\partial x}dxdy\right]_{i,j=1}^{N_{bu},N_{bp}},\qquad A_6=\left[\int_\Omega -\zeta_j\frac{\partial\varphi_i}{\partial y}dxdy\right]_{i,j=1}^{N_{bu},N_{bp}},$$
$$A_7=\left[\int_\Omega \frac{\partial\varphi_j}{\partial x}-\zeta_i dxdy\right]_{i,j=1}^{N_{bp},N_{bu}},\qquad A_8=\left[\int_\Omega \frac{\partial\varphi_j}{\partial y}-\zeta_i dxdy\right]_{i,j=1}^{N_{bp},N_{bu}},$$
$$A_9=\left[\int_\Omega -c_1\psi_j\varphi_i dxdy\right]_{i,j=1}^{N_{bu},N_{b\theta}},\qquad A_{10}=\left[\int_\Omega c_2\frac{\partial\psi_j}{\partial x}\frac{\partial\psi_i}{\partial x}dxdy\right]_{i,j=1}^{N_{b\theta}},$$

$$\text{and} \quad A_{11} = \left[\int_\Omega c_2 \frac{\partial \psi_j}{\partial y} \frac{\partial \psi_i}{\partial y} dxdy \right]_{i,j=1}^{N_{b\theta}}.$$

We realize that $A_4 = A_3^T$, $A_7 = A_5^T$ and $A_8 = A_6^T$. Furthermore, we generate the zero matrices $\mathbb{O}_1 = [0]_{i,j=1}^{N_{bp}}$, $\mathbb{O}_2 = [0]_{i,j=1}^{N_{bu},N_{bp}}$, $\mathbb{O}_3 = [0]_{i,j=1}^{N_{bu},N_{b\theta}}$ and $\mathbb{O}_4 = [0]_{i,j=1}^{N_{bp},N_{b\theta}}$. Now,

$$A = \begin{pmatrix} 2A_1 + A_2 & A_3 & A_5 & \mathbb{O}_3 \\ A_4 & 2A_2 + A_1 & A_6 & A_9 \\ A_7 & A_8 & \mathbb{O}_1 & \mathbb{O}_4 \\ \mathbb{O}_3^T & \mathbb{O}_3^T & \mathbb{O}_4^T & A_{10} + A_{11} \end{pmatrix}. \tag{3.8}$$

The next definitions are

$$\boldsymbol{b_1} = \left[\int_\Omega f_1 \varphi_i dxdy \right]_{i=1}^{N_{bu}}, \quad \boldsymbol{b_2} = \left[\int_\Omega f_2 \varphi_i dxdy \right]_{i=1}^{N_{bu}} \quad \text{and} \quad \boldsymbol{b_3} = \left[\int_\Omega f_\theta \psi_i dxdy \right]_{i=1}^{N_{b\theta}}$$

to get

$$\boldsymbol{b} = \begin{pmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{0} \\ \boldsymbol{b_3} \end{pmatrix} \tag{3.9}$$

with the zero vector $\boldsymbol{0} = [0]_{i=1}^{N_{bp}}$. For the parts within the Newton iteration, we define

$$AN_1 = \left[\int_\Omega \frac{\partial u_{1h}^{(l-1)}}{\partial x} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, \qquad AN_2 = \left[\int_\Omega u_{1h}^{(l-1)} \frac{\partial \varphi_j}{\partial x} \varphi_i dxdy \right]_{i,j=1}^{N_{bu}},$$

$$AN_3 = \left[\int_\Omega u_{2h}^{(l-1)} \frac{\partial \varphi_j}{\partial y} \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, \qquad AN_4 = \left[\int_\Omega \frac{\partial u_{1h}^{(l-1)}}{\partial y} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}},$$

$$AN_5 = \left[\int_\Omega \frac{\partial u_{2h}^{(l-1)}}{\partial x} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, \qquad AN_6 = \left[\int_\Omega \frac{\partial u_{2h}^{(l-1)}}{\partial y} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}},$$

$$AN_7 = \left[\int_\Omega \frac{\partial \theta_h^{(l-1)}}{\partial x}\varphi_j\psi_i dxdy\right]_{i,j=1}^{N_{b\theta},N_{bu}}, \qquad AN_8 = \left[\int_\Omega \frac{\partial \theta_h^{(l-1)}}{\partial y}\varphi_j\psi_i dxdy\right]_{i,j=1}^{N_{b\theta},N_{bu}},$$
$$AN_9 = \left[\int_\Omega u_{1h}^{(l-1)}\frac{\partial \psi_j}{\partial x}\psi_i dxdy\right]_{i,j=1}^{N_{b\theta}}, \qquad AN_{10} = \left[\int_\Omega u_{2h}^{(l-1)}\frac{\partial \psi_j}{\partial y}\psi_i dxdy\right]_{i,j=1}^{N_{b\theta}},$$

deriving

$$AN = \begin{pmatrix} AN_1 + AN_2 + AN_3 & AN_4 & \mathbb{O}_2 & \mathbb{O}_3 \\ AN_5 & AN_6 + AN_2 + AN_3 & \mathbb{O}_2 & \mathbb{O}_3 \\ \mathbb{O}_2^T & \mathbb{O}_2^T & \mathbb{O}_1 & \mathbb{O}_4 \\ AN_7 & AN_8 & \mathbb{O}_4^T & AN_9 + AN_{10} \end{pmatrix}. \tag{3.10}$$

Furthermore,

$$\boldsymbol{bN_1} = \left[\int_\Omega u_{1h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial x}\varphi_i dxdy\right]_{i=1}^{N_{bu}}, \qquad \boldsymbol{bN_2} = \left[\int_\Omega u_{2h}^{(l-1)}\frac{\partial u_{1h}^{(l-1)}}{\partial y}\varphi_i dxdy\right]_{i=1}^{N_{bu}},$$
$$\boldsymbol{bN_3} = \left[\int_\Omega u_{1h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial x}\varphi_i dxdy\right]_{i=1}^{N_{bu}}, \qquad \boldsymbol{bN_4} = \left[\int_\Omega u_{2h}^{(l-1)}\frac{\partial u_{2h}^{(l-1)}}{\partial y}\varphi_i dxdy\right]_{i=1}^{N_{bu}},$$
$$\boldsymbol{bN_5} = \left[\int_\Omega u_{1h}^{(l-1)}\frac{\partial \theta_h^{(l-1)}}{\partial x}\psi_i dxdy\right]_{i=1}^{N_{b\theta}}, \qquad \boldsymbol{bN_6} = \left[\int_\Omega u_{2h}^{(l-1)}\frac{\partial \theta_h^{(l-1)}}{\partial y}\psi_i dxdy\right]_{i=1}^{N_{b\theta}},$$

are defined to generate

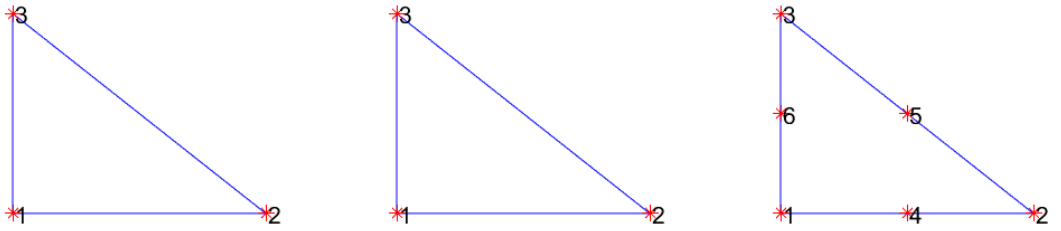
$$\boldsymbol{bN} = \begin{pmatrix} \boldsymbol{bN_1} + \boldsymbol{bN_2} \\ \boldsymbol{bN_3} + \boldsymbol{bN_4} \\ \boldsymbol{0} \\ \boldsymbol{bN_5} + \boldsymbol{bN_6} \end{pmatrix}. \tag{3.11}$$

Therefore, for each step l $(l = 1, 2, ..., L)$ of the Newton iteration, we solve the linear algebraic system

$$A^{(l)}\boldsymbol{x}^{(l)} = \boldsymbol{b}^{(l)},$$

where $A^{(l)} = A + AN$, $\boldsymbol{x}^{(l)} = (u_{1h}^{(l)}, u_{2h}^{(l)}, p_h^{(l)}, \theta_h^{(l)})^T$ and $\boldsymbol{b}^{(l)} = \boldsymbol{b} + \boldsymbol{bN}$.

3.1.5. Algorithms. In the following algorithms, N_b is the total number of finite element basis functions or finite element nodes, N_{lb} is the number of local finite element nodes and P_b and T_b are the finite element information matrices. More precisely, we use a triangular mesh and either linear or quadratic finite elements, as stated before. The nodes of each element are labeled as shown in Figure 3.1. Thus, there are either three or six local finite element nodes and N_b is given by N_{bu}, N_{bp} or $N_{b\theta}$ for the velocity functions u_1 and u_2, the pressure p or the temperature θ, respectively. The matrix P_b consists of the coordinates of all finite element nodes. That means P_b is a $2 \times N_b$ matrix storing the x-coordinate in the first row and the y-coordinate in the second row for each finite element node. On the other hand, the matrix T_b consists of the indices of the finite element nodes of all the mesh elements. Thus, T_b has N_{lb} rows and as many columns as mesh elements. While the number of mesh elements, denoted by N, is the same for the velocity $\boldsymbol{u}$, the pressure p, and the temperature θ, the number of finite element nodes varies, such that different P_{bu}, P_{bp}, $P_{b\theta}$ and T_{bu}, T_{bp}, $T_{b\theta}$ are generated.



(a) Mesh nodes (b) Linear FEM nodes (c) Quadratic FEM nodes

Figure 3.1. The nodes in one triangular element of the mesh and finite element spaces used to realize the standard Galerkin FEM in `MATLAB`

To realize the derived coupled finite element method, we structure our `MATLAB` code in the following way.

(i) Generate matrix A as in (3.8) using Algorithm 1

(ii) Generate vector $\boldsymbol{b}$ as in (3.9) using Algorithm 2

(iii) FOR $l = 1, 2, ..., L$ (begin Newton iteration for linearization)

(iv) Generate matrix AN as in (3.10) using Algorithm 3

(v) Generate vector $\boldsymbol{bN}$ as in (3.11) using Algorithm 4

(vi) Set $A^{(l)} = A + AN$ and $\boldsymbol{b}^{(l)} = \boldsymbol{b} + \boldsymbol{bN}$

(vii) Treat the Dirichlet boundary condition using Algorithm 5

(viii) Fix pressure $p_h^{(l)}$ at one point in Ω

(ix) Solve $A^{(l)}\boldsymbol{x}^{(l)} = \boldsymbol{b}^{(l)}$ using the backslash operator

(x) END

Steps (i) and (ii) are independent of the Newton iteration because they do not contain $u_{1h}^{(l-1)}$, $u_{2h}^{(l-1)}$ or $\theta_h^{(l-1)}$. Thus, A and $\boldsymbol{b}$ will be generated once and not altered within the loop. Contrariwise, steps (iv) and (v) depend on $u_{1h}^{(l-1)}$, $u_{2h}^{(l-1)}$, and $\theta_h^{(l-1)}$ and we rerun those in each iteration.

For the purpose of an easily adjustable implementation, Algorithm 1 and Algorithm 2 to generate A and $\boldsymbol{b}$ were derived. A Gauss quadrature with nine Gauss points is used to approximate the integrals here and in all other cases. With the coefficient function c and the derivative orders as main input, Algorithm 1 generates A_1 to A_{11} to set up A. Similarly, Algorithm 2 calculates $\boldsymbol{b_1}$ to $\boldsymbol{b_3}$ to set up $\boldsymbol{b}$.

Due to the fact that the coefficient function of AN is not a constant or a function depending on x and y anymore, but a finite element coefficient function, we modify Algorithm 1 slightly into Algorithm 3, where c is surrogated by a finite element coefficient function.

Algorithm 1 Assemble the stiffness matrix A

function ASSEMBLEMATRIX($fun\_c, ..., r, s, p, q$)
  $A = sparse(N_b, N_b)$
  for $n = 1, ..., N$ **do**
    for $\alpha = 1, ..., N_{lb}$ **do**
      for $\beta = 1, ..., N_{lb}$ **do**
        Compute $r = \int_{E_n} c \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q} dxdy$
        Add r to $A(T_b(\beta, n), T_b(\alpha, n))$
      end for
    end for
  end for
end function

Algorithm 2 Assemble the load vector $\boldsymbol{b}$

function ASSEMBLEVECTOR($fun\_f, ..., p, q$)
  $b = sparse(N_b, 1)$
  for $n = 1, ..., N$ **do**
    for $\beta = 1, ..., N_{lb}$ **do**
      Compute $r = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q} dxdy$
      Add r to $b(T_b(\beta, n), 1)$
    end for
  end for
end function

Similarly, for $\boldsymbol{bN}$, Algorithm 2 has to be modified into Algorithm 4 by replacing the coefficient function c by finite element coefficient functions f_{1h} and f_{2h}. Now, Algorithms 3 and 4 are used to generate AN_1 to AN_{10} and $\boldsymbol{bN_1}$ to $\boldsymbol{bN_6}$, respectively, choosing $u_{1h}^{(l-1)}$, $u_{2h}^{(l-1)}$ or $\theta_h^{(l-1)}$ from the previous iteration step or their derivatives as coefficients.

Ultimately, we have to deal with the Dirichlet boundary condition as stated in step (vii). We impose the boundary condition by replacing the boundary nodes equations in the linear system by

$$u_{1m}^{(l)} = g_1(x_m, y_m), \qquad u_{2m}^{(l)} = g_2(x_m, y_m), \qquad \text{and} \qquad \theta_m^{(l)} = g_\theta(x_m, y_m)$$

Algorithm 3 Assemble the linearized stiffness matrix AN

function ASSEMBLELINEARIZEDMATRIX($fun\_c, ..., r, s, p, q, d, e$)
  $A = sparse(N_b, N_b)$
  for $n = 1, ..., N$ **do**
    for $\alpha = 1, ..., N_{lb}$ **do**
      for $\beta = 1, ..., N_{lb}$ **do**
        Compute $r = \int_{E_n} \frac{\partial^{d+e} c_h}{\partial x^d \partial y^e} \frac{\partial^{r+s} \psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q} \psi_{n\beta}}{\partial x^p \partial y^q} dxdy$
        Add r to $A(T_b(\beta, n), T_b(\alpha, n))$
      end for
    end for
  end for
end function

Algorithm 4 Assemble the linearized load vector $\boldsymbol{bN}$

function ASSEMBLELINEARIZEDVECTOR($fun\_f_{1h}, fun\_f_{2h}, ..., p, q, d, e, r, s$)
  $b = sparse(N_b, 1)$
  for $n = 1, ..., N$ **do**
    for $\beta = 1, ..., N_{lb}$ **do**
      Compute $r = \int_{E_n} \frac{\partial^{d+e} f_{1h}}{\partial x^d \partial y^e} \frac{\partial^{r+s} f_{2h}}{\partial x^r \partial y^s} \frac{\partial^{p+q} \psi_{n\beta}}{\partial x^p \partial y^q} dxdy$
      Add r to $b(T_b(\beta, n), 1)$
    end for
  end for
end function

for all boundary nodes m. Since the boundary condition does not involve the pressure p, the third row of the blockmatrix $A^{(l)}$ and the third entry of $\boldsymbol{b}^{(l)}$ should not be altered. The modification is fulfilled by Algorithm 5. If the same finite element functions are used for the velocity $\boldsymbol{u}$ and the temperature θ, so $\varphi_j = \psi_j$ for $j = 1, ..., N_{bu} = N_{b\theta}$, then they have the same boundary nodes, and the two loops in Algorithm 5 can be merged together.

Algorithm 5 Treat Dirichlet boundary condition

function BOUNDARYTREATMENT($fun\_g_1, fun\_g_2, fun\_g_\theta, ..., A^{(l)}, \boldsymbol{b^{(l)}}, N_b, N_{bp}$)
  for $n = 1, ..., number\_of\_boundarynodes\_for\_\boldsymbol{u}$ **do**
    Set i = index of current boundary node and
    $A^{(l)}(i, :) = 0$
    $A^{(l)}(N_b + i, :) = 0$
    $A^{(l)}(i, i) = 1$
    $A^{(l)}(N_b + i, N_b + i) = 1$
    $\boldsymbol{b^{(l)}}(i) = fun\_g_1(P_b(:, i))$
    $\boldsymbol{b^{(l)}}(N_b + i) = fun\_g_2(P_b(:, i))$
  end for
  for $n = 1, ..., number\_of\_boundarynodes\_for\_\theta$ **do**
    Set i = index of current boundary node and
    $A^{(l)}(2 * N_b + N_{bp} + i, :) = 0$
    $A^{(l)}(2 * N_b + N_{bp} + i, 2 * N_b + N_{bp} + i) = 1$
    $\boldsymbol{b^{(l)}}(2 * N_b + N_{bp} + i) = fun\_g_\theta(P_b(:, i))$
  end for
end function

3.2. NUMERICAL EXAMPLE

The purpose of this section is to verify our previous derivation by a numerical example. Therefore, we solve the given Boussinesq equations

$$\begin{cases} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla \cdot \mathbb{T}(\boldsymbol{u}, p) - \boldsymbol{e_2}\theta = \boldsymbol{f_u}, & \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0, & \text{in } \Omega, \\ \boldsymbol{u} \cdot \nabla \theta - \nabla \cdot (\nabla \theta) = f_\theta, & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{g_u}, & \text{on } \partial\Omega, \\ \theta = g_\theta, & \text{on } \partial\Omega, \end{cases} \tag{3.12}$$

on the domain $\Omega = [0, 1] \times [-0.25, 0]$, where the viscosity $\nu = 1$ and the forcing functions

$$f_1 = -2\nu\left(x^2 + y^2\right) - \nu e^{-y} + \pi^2 \cos(\pi x)\cos(2\pi y) + 2xy^2\left(x^2y^2 + e^{-y}\right)$$

$$
\begin{aligned}
&+\left(-2x\frac{y^3}{3}+2-\pi\sin(\pi x)\right)\left(2x^2y-e^{-y}\right),\\
f_2 =&4\nu xy-\nu\pi^3\sin(\pi x)+2\pi\left(2-\pi\sin(\pi x)\right)\sin(2\pi y)\\
&+\left(x^2+y^2+e^{-y}\right)\left(-2\frac{y^3}{3}-\pi^2\cos(\pi x)\right)\\
&+\left(-2x\frac{y^3}{3}+2-\pi\sin(\pi x)\right)\left(-2xy^2\right)-e^{x+y},\\
f_\theta =&\left(x^2y^2+e^{-y}\right)e^{x+y}+\left(-\frac{2}{3}xy^3+2-\pi\sin(\pi x)\right)e^{x+y}-2e^{x+y}.
\end{aligned}
$$

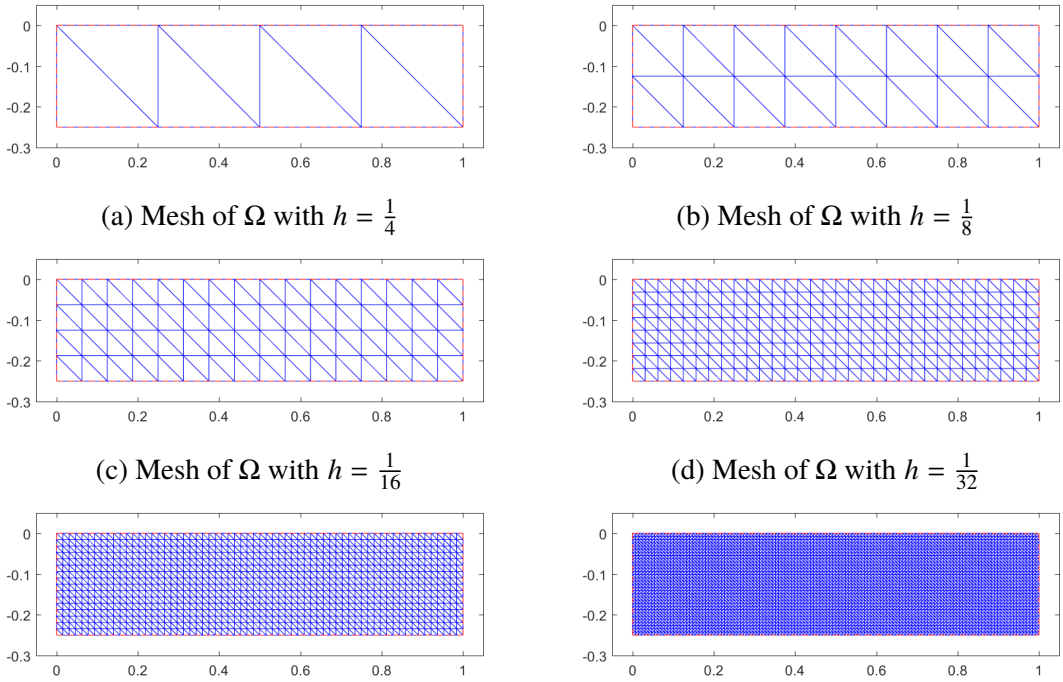
Furthermore, the boundary functions are given by

$$
\begin{aligned}
&u_1=e^{-y}, && u_2=2, && \text{and} && \theta=e^{y} && \text{on } x=0,\\
&u_1=y^2+e^{-y}, && u_2=-\frac{2}{3}y^3+2, && \text{and} && \theta=e^{1+y} && \text{on } x=1,\\
&u_1=\frac{1}{16}x^2+e^{0.25}, && u_2=\frac{1}{96}x+2-\pi\sin(\pi x), && \text{and} && \theta=e^{x-0.25} && \text{on } y=-0.25,\\
&u_1=e^{-y}, && u_2=2-\pi\sin(\pi x), && \text{and} && \theta=e^{x} && \text{on } y=0.
\end{aligned}
$$

The forcing functions $\boldsymbol{f_u}$ and f_θ are stated with nearly no simplifications such that the analytical solutions

$$
\begin{aligned}
u_1 &= x^2y^2+e^{-y},\\
u_2 &= -\frac{2}{3}xy^3+2-\pi\sin(\pi x),\\
p &= -(2-\pi\sin(\pi x))\cos(2\pi y), \quad \text{and}\\
\theta &= e^{x+y}
\end{aligned}
$$

are clearly evident. We solve Equation (3.12) using the derived finite element method for six different meshes with edge sizes $h \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right\}$. As shown in Figure 3.2, we generate an equidistant triangular mesh and halve h in every step.



(a) Mesh of Ω with $h = \frac{1}{4}$

(b) Mesh of Ω with $h = \frac{1}{8}$

(c) Mesh of Ω with $h = \frac{1}{16}$

(d) Mesh of Ω with $h = \frac{1}{32}$

(e) Mesh of Ω with $h = \frac{1}{64}$

(f) Mesh of Ω with $h = \frac{1}{128}$

Figure 3.2. The mesh of the domain $\Omega = [0, 1] \times [-0.25, 0]$ for different edge sizes h used in the standard Galerkin FEM

To analyze the finite element solution $\boldsymbol{u_h}$ of the velocity $\boldsymbol{u}$, we calculate not only the maximum error MaxError = $\max|\boldsymbol{u} - \boldsymbol{u_h}|$, but also the L^∞ norm error InfError, the L^2 norm error L2Error and the H^1 semi-norm error H1Error, which are known to be as follows.

$$\text{InfError} = \|\boldsymbol{u} - \boldsymbol{u_h}\|_\infty = \sup_{(x,y)\in\Omega} |\boldsymbol{u}(x, y) - \boldsymbol{u_h}(x, y)|$$

$$\text{L2Error} = \|\boldsymbol{u} - \boldsymbol{u_h}\|_0 = \sqrt{\int_\Omega |\boldsymbol{u} - \boldsymbol{u_h}|^2 dxdy}$$

$$\text{H1Error} = |\boldsymbol{u} - \boldsymbol{u_h}|_1 = \sqrt{\int_\Omega \left|\frac{\partial(\boldsymbol{u} - \boldsymbol{u_h})}{\partial x}\right|^2 dxdy + \int_\Omega \left|\frac{\partial(\boldsymbol{u} - \boldsymbol{u_h})}{\partial y}\right|^2 dxdy}$$

The same is done for the finite element solutions p_h of the pressure p and θ_h of the temperature θ. The results are stated in Tables 3.1, 3.2, 3.3 and 3.4 for the four different error types MaxError, InfError, L2Error and H1Error, respectively. Due to the high regularity of

Table 3.1. The maximum errors and corresponding convergence rate of standard Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	MaxError $\boldsymbol{u}$	Rate	MaxError p	Rate	MaxError θ	Rate
4	$1.4696 \cdot 10^{-3}$	–	$6.0149 \cdot 10^{-1}$	–	$2.6162 \cdot 10^{-5}$	–
8	$4.5401 \cdot 10^{-4}$	1.6946	$2.0711 \cdot 10^{-1}$	1.5381	$1.9435 \cdot 10^{-6}$	3.7507
16	$3.7082 \cdot 10^{-5}$	3.6139	$5.5743 \cdot 10^{-2}$	1.8935	$1.1358 \cdot 10^{-7}$	4.0969
32	$2.3809 \cdot 10^{-6}$	3.9612	$1.3928 \cdot 10^{-2}$	2.0008	$8.5492 \cdot 10^{-9}$	3.7318
64	$1.8440 \cdot 10^{-7}$	3.6906	$3.4684 \cdot 10^{-3}$	2.0057	$5.9340 \cdot 10^{-10}$	3.8487
128	$2.1410 \cdot 10^{-8}$	3.1065	$8.6461 \cdot 10^{-4}$	2.0042	$3.8963 \cdot 10^{-11}$	3.9288

Table 3.2. The L^∞ errors and corresponding convergence rate of standard Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	InfError $\boldsymbol{u}$	Rate	InfError p	Rate	InfError θ	Rate
4	$1.1759 \cdot 10^{-2}$	–	$3.7781 \cdot 10^{-1}$	–	$3.0632 \cdot 10^{-4}$	–
8	$1.6853 \cdot 10^{-3}$	2.8026	$1.3616 \cdot 10^{-1}$	1.4724	$4.0349 \cdot 10^{-5}$	2.9244
16	$2.0224 \cdot 10^{-4}$	3.0589	$4.5862 \cdot 10^{-2}$	1.5699	$5.1810 \cdot 10^{-6}$	2.9612
32	$2.5167 \cdot 10^{-5}$	3.0065	$1.2533 \cdot 10^{-2}$	1.8716	$6.5635 \cdot 10^{-7}$	2.9807
64	$3.1048 \cdot 10^{-6}$	3.0190	$3.2510 \cdot 10^{-3}$	1.9468	$8.2594 \cdot 10^{-8}$	2.9904
128	$3.8464 \cdot 10^{-7}$	3.0129	$8.2650 \cdot 10^{-4}$	1.9758	$1.0359 \cdot 10^{-8}$	2.9952

Table 3.3. The L^2 errors and corresponding convergence rate of standard Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	L2Error $\boldsymbol{u}$	Rate	L2Error p	Rate	L2Error θ	Rate
4	$2.7603 \cdot 10^{-3}$	–	$8.2694 \cdot 10^{-2}$	–	$6.9312 \cdot 10^{-5}$	–
8	$3.5640 \cdot 10^{-4}$	2.9532	$2.2577 \cdot 10^{-2}$	1.8729	$8.5992 \cdot 10^{-6}$	3.0108
16	$4.4016 \cdot 10^{-5}$	3.0174	$8.6669 \cdot 10^{-3}$	1.3813	$1.0718 \cdot 10^{-6}$	3.0042
32	$5.4798 \cdot 10^{-6}$	3.0058	$2.4764 \cdot 10^{-3}$	1.8073	$1.3389 \cdot 10^{-7}$	3.0009
64	$6.8421 \cdot 10^{-7}$	3.0016	$6.5584 \cdot 10^{-4}$	1.9168	$1.6733 \cdot 10^{-8}$	3.0002
128	$8.5497 \cdot 10^{-8}$	3.0005	$1.6841 \cdot 10^{-4}$	1.9613	$2.0916 \cdot 10^{-9}$	3.0001

Table 3.4. The H^1 errors and corresponding convergence rate of standard Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	H1Error $\boldsymbol{u}$	Rate	H1Error p	Rate	H1Error θ	Rate
4	$8.0560 \cdot 10^{-2}$	–	$2.3556 \cdot 10^{0}$	–	$2.6175 \cdot 10^{-3}$	–
8	$2.0429 \cdot 10^{-2}$	1.9795	$1.2648 \cdot 10^{0}$	0.8972	$6.5321 \cdot 10^{-4}$	2.0025
16	$5.0681 \cdot 10^{-3}$	2.0111	$6.3069 \cdot 10^{-1}$	1.0039	$1.6323 \cdot 10^{-4}$	2.0006
32	$1.2623 \cdot 10^{-3}$	2.0054	$3.1369 \cdot 10^{-1}$	1.0076	$4.0804 \cdot 10^{-5}$	2.0001
64	$3.1523 \cdot 10^{-4}$	2.0016	$1.5658 \cdot 10^{-1}$	1.0024	$1.0201 \cdot 10^{-5}$	2.0000
128	$7.8782 \cdot 10^{-5}$	2.0004	$7.8254 \cdot 10^{-2}$	1.0007	$2.5502 \cdot 10^{-6}$	2.0000

the selected functions $\boldsymbol{u}$, p, and θ in Equation (3.12), the optimal convergence capability expected from piecewise quadratic finite element functions is third order convergence $O(h^3)$ in the L^∞ and L^2 norm, and second order convergence $O(h^2)$ in the H^1 semi-norm. The columns called rate, where the convergence rate of the error after each bisection of h calculated by linear regression is stated, of Tables 3.2, 3.3, and 3.4 confirm that the optimal convergence rate is reached for $\boldsymbol{u_h}$ and θ_h. For the MaxError, in Table 3.1 we even observe fourth order convergence $O(h^4)$. From piecewise linear finite element functions we expect second order convergence $O(h^2)$ in the L^∞ and L^2 norm, and first order convergence $O(h)$ in the H^1 semi-norm. As Tables 3.2, 3.3 and 3.4 show, this expectation is fulfilled for p_h. As anticipated and stated in Table 3.1, p_h exhibits second order convergence $O(h^2)$ using the maximum error.

4. DECOUPLED FINITE ELEMENT METHODS

In this section, we finally derive the new decoupled finite element methods using the previous derived standard Galerkin finite element method as a foundation. The structure of this section is the same as the structure of the preceding one such that the transformation is easily comprehensible. Due to different decoupling techniques, we introduce a parallel and a sequential approach in Sections 4.1 and 4.2, respectively.

4.1. DERIVATION OF THE PARALLEL APPROACH

4.1.1. Weak Formulation. By carrying out the first steps of Section 3.1 again, we start with the coupled Boussinesq equations

$$\begin{aligned} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nabla \cdot \mathbb{T}(\boldsymbol{u}, p) - \boldsymbol{e_2} c_1 \theta &= \boldsymbol{f_u}, \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= 0, \quad \text{in } \Omega, \quad \text{and} \\ \boldsymbol{u} \cdot \nabla \theta - \nabla \cdot (c_2 \nabla \theta) &= f_\theta, \quad \text{in } \Omega, \end{aligned}$$

and treat them separately by introducing test functions $\boldsymbol{v}$, q and w, integrating in multi dimensions and applying the boundary condition $\boldsymbol{u} = \boldsymbol{g_u}$ and $\theta = g_\theta$ on $\partial\Omega$ until we get the three equations

$$\begin{aligned} &\int_\Omega (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \cdot \boldsymbol{v} dxdy + \int_\Omega 2\nu \mathbb{D}(\boldsymbol{u}) : \mathbb{D}(\boldsymbol{v}) dxdy - \int_\Omega p(\nabla \cdot \boldsymbol{v}) dxdy - \int_\Omega \boldsymbol{e_2} c_1 \theta \cdot \boldsymbol{v} dxdy \\ &\quad = \int_\Omega \boldsymbol{f_u} \cdot \boldsymbol{v} dxdy, \\ &\int_\Omega (\nabla \cdot \boldsymbol{u}) q dxdy = 0, \quad \text{and} \\ &\int_\Omega (\boldsymbol{u} \cdot \nabla \theta) w dxdy + \int_\Omega c_2 \nabla \theta \cdot \nabla w dxdy = \int_\Omega f_\theta w dxdy. \end{aligned}$$

To increase the clarity of further formulations, we take advantage of the already defined operators

$$
\begin{aligned}
c(\boldsymbol{w},\boldsymbol{u},\boldsymbol{v}) &= \int_\Omega (\boldsymbol{w}\cdot\nabla)\boldsymbol{u}\cdot\boldsymbol{v}dxdy,\\
a(\boldsymbol{u},\boldsymbol{v}) &= \int_\Omega 2\nu\mathbb{D}(\boldsymbol{u}):\mathbb{D}(\boldsymbol{v})dxdy,\\
b(\boldsymbol{u},q) &= -\int_\Omega (\nabla\cdot\boldsymbol{u})qdxdy,\\
(\boldsymbol{f},\boldsymbol{v}) &= \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}dxdy \quad \text{or} \quad (f,w) = \int_\Omega fwdxdy,\\
\tilde{a}(\theta,w) &= \int_\Omega c_2\nabla\theta\cdot\nabla w dxdy, \quad \text{and}\\
\tilde{c}(\boldsymbol{u},\theta,w) &= \int_\Omega (\boldsymbol{u}\cdot\nabla\theta)wdxdy.
\end{aligned}
$$

Now, we decouple those equations into the Navier-Stokes equations and a parabolic equation by introducing a new iteration. Therefore, we obtain the following weak formulation.

Initial guess: $\boldsymbol{u^{(0)}}$ and $\theta^{(0)}$

FOR $k = 1, 2, ..., K$

Step 1:
$$
\begin{cases}
\text{Find } \boldsymbol{u^{(k)}} \in \left(H^1(\Omega)\right)^2 \text{ and } p^{(k)} \in L^2(\Omega) \text{ such that}\\
c(\boldsymbol{u^{(k)}},\boldsymbol{u^{(k)}},\boldsymbol{v}) + a(\boldsymbol{u^{(k)}},\boldsymbol{v}) + b(\boldsymbol{v},p^{(k)}) - (\boldsymbol{e_2}c_1\theta^{(k-1)},\boldsymbol{v}) = (\boldsymbol{f_u},\boldsymbol{v}),\\
b(\boldsymbol{u^{(k)}},q) = 0,\\
\text{for any } \boldsymbol{v} \in \left(H_0^1(\Omega)\right)^2 \text{ and } q \in L^2(\Omega).
\end{cases}
$$

Step 2:
$$
\begin{cases}
\text{Find } \theta^{(k)} \in H^1(\Omega) \text{ such that}\\
\tilde{c}(\boldsymbol{u^{(k-1)}},\theta^{(k)},w) + \tilde{a}(\theta^{(k)},w) = (f_\theta,w),\\
\text{for any } w \in H_0^1(\Omega).
\end{cases}
$$

END

Here, step 1 solves the Navier-Stokes equations for the velocity $\boldsymbol{u}^{(\boldsymbol{k})}$ and the pressure $p^{(k)}$ using the solution for the temperature $\theta^{(k-1)}$ of the previous iteration step. On the other hand, in step 2 the convection-diffusion equation is solved for the temperature $\theta^{(k)}$ using only the velocity approximation $\boldsymbol{u}^{(\boldsymbol{k-1})}$ of the previous iteration step. Hence, step 1 and step 2 are independent of each other and can be executed in parallel.

At this point, the flexibility of the new technique becomes apparent. Since the two steps are independent of each other, different methods could be applied. For instance, the finite volume method could be used for step 2 to conserve the properties of the temperature θ. Furthermore, we emphasize the convenience if there exist solvers for the Navier-Stokes equations or the parabolic equation that can be exploited now.

4.1.2. Galerkin Formulation. We obtain the Galerkin formulation based on the weak formulation by defining the same finite element spaces as before: $U_h \subset H^1(\Omega)$ for the velocity functions u_1 and u_2, $P_h \subset L^2(\Omega)$ for the pressure p and $W_h \subset H^1(\Omega)$ for the temperature θ. Since we will deal with the Dirichlet boundary conditions later for an easier implementation, this leads to the following Galerkin formulation.

Initial guess: $\boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{0})}$ and $\theta_h^{(0)}$

FOR $k = 1, 2, ..., K$

$$\text{Step 1:} \begin{cases} \text{Find } \boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k})} \in (U_h)^2 \text{ and } p_h^{(k)} \in P_h \text{ such that} \\ c(\boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k})}, \boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k})}, \boldsymbol{v_h}) + a(\boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k})}, \boldsymbol{v_h}) + b(\boldsymbol{v_h}, p_h^{(k)}) - (\boldsymbol{e_2} c_1 \theta_h^{(k-1)}, \boldsymbol{v_h}) = (\boldsymbol{f_u}, \boldsymbol{v_h}), \\ b(\boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k})}, q_h) = 0, \\ \text{for any } \boldsymbol{v_h} \in (U_h)^2 \text{ and } q_h \in P_h. \end{cases}$$

Step 2: $\begin{cases} \text{Find } \theta_h^{(k)} \in W_h \text{ such that} \\ \tilde{c}(\boldsymbol{u_h^{(k-1)}}, \theta_h^{(k)}, w_h) + \tilde{a}(\theta_h^{(k)}, w_h) = (f_\theta, w_h), \\ \text{for any } w_h \in W_h. \end{cases}$

END

Again, we choose the traditional quadratic global basis functions $\{\varphi_j\}_{j=1}^{N_{bu}}$, $\{\psi_j\}_{j=1}^{N_{b\theta}}$ and linear global basis functions $\{\zeta_j\}_{j=1}^{N_{bp}}$ and set $U_h = span\{\varphi_j\}_{j=1}^{N_{bu}}$, $W_h = span\{\psi_j\}_{j=1}^{N_{b\theta}}$ and $P_h = span\{\zeta_j\}_{j=1}^{N_{bp}}$.

4.1.3. Newton Iteration. Due to the decoupling, the nonlinear term in the convection-diffusion equation vanished. Nonetheless, there is still a nonlinear term in the Navier-Stokes equations, which we linearize by the following Newton iteration.

Initial guess: $\boldsymbol{u_h^{(0)}}$ and $\theta_h^{(0)}$

FOR $k = 1, 2, ..., K$

Step 1: $\begin{cases} \text{Initial guess: } \boldsymbol{u_h^{(k,0)}} \\ \text{FOR } l = 1, 2, ..., L \\ \quad \text{Find } \boldsymbol{u_h^{(k,l)}} \in (U_h)^2 \text{ and } p_h^{(k,l)} \in P_h \text{ such that} \\ \quad c(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) + a(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) \\ \quad +b(\boldsymbol{v_h}, p_h^{(k,l)}) - (\boldsymbol{e_2} c_1 \theta_h^{(k-1)}, \boldsymbol{v_h}) = (\boldsymbol{f_u}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}), \\ \quad b(\boldsymbol{u_h^{(k,l)}}, q_h) = 0, \\ \quad \text{for any } \boldsymbol{v_h} \in (U_h)^2 \text{ and } q_h \in P_h. \\ \text{END} \end{cases}$

$$\text{Step 2: } \begin{cases} \text{Find } \theta^{(k)} \in W_h \text{ such that} \\ \tilde{c}(\boldsymbol{u}_{\boldsymbol{h}}^{(\boldsymbol{k-1})}, \theta_h^{(k)}, w_h) + \tilde{a}(\theta_h^{(k)}, w_h) = (f_\theta, w_h), \\ \text{for any } w_h \in W_h. \end{cases}$$

END

4.1.4. Finite Element Discretization. Now, the Equations in (3.7) still hold. This being said, for $u_{1h}^{(l)}, u_{2h}^{(l)} \in U_h$, $p_h^{(l)} \in P_h$ and $\theta_h^{(l)} \in W_h$ we want to obtain the coefficients $u_{1h}^{(l)}$, $u_{2h}^{(l)}$ $(j = 1, ..., N_{bu})$, $p_h^{(l)}$ $(j = 1, ..., N_{bp})$ and $\theta_h^{(l)}$ $(j = 1, ..., N_{b\theta})$ to approximate the velocity $\boldsymbol{u}$, the pressure p and the temperature θ using a linear algebraic system.

For step 1, we choose $\boldsymbol{v_h} = (\varphi_i, 0)^T$ $(i = 1, ..., N_{bu})$ and $\boldsymbol{v_h} = (0, \varphi_i)^T$ $(i = 1, ..., N_{bu})$ for the first equation and $q_h = \zeta_i$ $(i = 1, ..., N_{bp})$ for the second equation for each iteration step k, l $(k = 1, 2, ..., K; l = 1, 2, ..., L)$. As before, we simplify the equations to make the matrix formulations more obvious.

$$\begin{aligned}
&\sum_{j=1}^{N_{bu}} u_{1j}^{(k,l)} \left(\int_\Omega \frac{\partial u_{1h}^{(k,l-1)}}{\partial x} \varphi_j \varphi_i dxdy + \int_\Omega u_{1h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial x} \varphi_i dxdy + \int_\Omega u_{2h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial y} \varphi_i dxdy \right. \\
&\qquad \left. +2 \int_\Omega \nu \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial x} dxdy + \int_\Omega \nu \frac{\partial \varphi_j}{\partial y} \frac{\partial \varphi_i}{\partial y} dxdy \right) \\
&+ \sum_{j=1}^{N_{bu}} u_{2j}^{(k,l)} \left(\int_\Omega \frac{\partial u_{1h}^{(k,l-1)}}{\partial y} \varphi_j \varphi_i dxdy + \int_\Omega \nu \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial y} dxdy \right) \\
&+ \sum_{j=1}^{N_{bp}} p_j^{(k,l)} \left(- \int_\Omega \zeta_j \frac{\partial \varphi_i}{\partial x} dxdy \right) \\
&= \int_\Omega f_1 \varphi_j dxdy + \int_\Omega u_{1h}^{(k,l-1)} \frac{\partial u_{1h}^{(k,l-1)}}{\partial x} \varphi_i dxdy + \int_\Omega u_{2h}^{(k,l-1)} \frac{\partial u_{1h}^{(k,l-1)}}{\partial y} \varphi_i dxdy
\end{aligned}$$

$$\sum_{j=1}^{N_{bu}} u_{1j}^{(k,l)} \left(\int_\Omega \frac{\partial u_{2h}^{(k,l-1)}}{\partial x} \varphi_j \varphi_i dxdy + \int_\Omega \nu \frac{\partial \varphi_j}{\partial y} \frac{\partial \varphi_i}{\partial x} dxdy \right)$$

$$
\begin{aligned}
&+\sum_{j=1}^{N_{bu}} u_{2j}^{(k,l)} \left(\int_\Omega \frac{\partial u_{2h}^{(k,l-1)}}{\partial y} \varphi_j \varphi_i dxdy + \int_\Omega u_{1h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial x} \varphi_i dxdy + \int_\Omega u_{2h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial y} \varphi_i dxdy \right. \\
&\qquad \left. +2 \int_\Omega \nu \frac{\partial \varphi_j}{\partial y} \frac{\partial \varphi_i}{\partial y} dxdy + \int_\Omega \nu \frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_i}{\partial x} dxdy \right) \\
&+\sum_{j=1}^{N_{bp}} p_j^{(k,l)} \left(- \int_\Omega \zeta_j \frac{\partial \varphi_i}{\partial y} dxdy \right) \\
&= \int_\Omega f_2 \varphi_j dxdy + \int_\Omega u_{1h}^{(k,l-1)} \frac{\partial u_{2h}^{(k,l-1)}}{\partial x} \varphi_i dxdy + \int_\Omega u_{2h}^{(k,l-1)} \frac{\partial u_{2h}^{(k,l-1)}}{\partial y} \varphi_i dxdy \\
&+ \int_\Omega c_1 \theta_h^{(k-1)} \varphi_i dxdy
\end{aligned}
$$

$$
\sum_{j=1}^{N_{bu}} u_{1j}^{(k,l)} \left(- \int_\Omega \frac{\partial \varphi_j}{\partial x} \zeta_i dxdy \right) + \sum_{j=1}^{N_{bu}} u_{2j}^{(k,l)} \left(- \int_\Omega \frac{\partial \varphi_j}{\partial y} \zeta_i dxdy \right) = 0
$$

To set up the linear system, retaining the definitions of A_1 to A_8, $\boldsymbol{b_1}$, $\boldsymbol{b_2}$ and the zero matrices and vectors from the previous section, we get

$$
A_{\boldsymbol{u},p} = \begin{pmatrix} 2A_1 + A_2 & A_3 & A_5 \\ A_4 & 2A_2 + A_1 & A_6 \\ A_7 & A_8 & \mathbb{O}_1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{b}_{\boldsymbol{u},p} = \begin{pmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{0} \end{pmatrix}. \tag{4.1}
$$

For the parts within the Newton iteration, we define

$$
\begin{aligned}
AN_1 &= \left[\int_\Omega \frac{\partial u_{1h}^{(k,l-1)}}{\partial x} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, & AN_2 &= \left[\int_\Omega u_{1h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial x} \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, \\
AN_3 &= \left[\int_\Omega u_{2h}^{(k,l-1)} \frac{\partial \varphi_j}{\partial y} \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, & AN_4 &= \left[\int_\Omega \frac{\partial u_{1h}^{(k,l-1)}}{\partial y} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, \\
AN_5 &= \left[\int_\Omega \frac{\partial u_{2h}^{(k,l-1)}}{\partial x} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}}, & AN_6 &= \left[\int_\Omega \frac{\partial u_{2h}^{(k,l-1)}}{\partial y} \varphi_j \varphi_i dxdy \right]_{i,j=1}^{N_{bu}},
\end{aligned}
$$

deriving

$$AN_{\boldsymbol{u},p} = \begin{pmatrix} AN_1 + AN_2 + AN_3 & AN_4 & \mathbb{O}_2 \\ AN_5 & AN_6 + AN_2 + AN_3 & \mathbb{O}_2 \\ \mathbb{O}_2^T & \mathbb{O}_2^T & \mathbb{O}_1 \end{pmatrix}. \tag{4.2}$$

Furthermore, we define

$$\boldsymbol{bN_1} = \left[\int_\Omega u_{1h}^{(k,l-1)} \frac{\partial u_{1h}^{(k,l-1)}}{\partial x} \varphi_i dxdy \right]_{i=1}^{N_{bu}}, \qquad \boldsymbol{bN_2} = \left[\int_\Omega u_{2h}^{(k,l-1)} \frac{\partial u_{1h}^{(k,l-1)}}{\partial y} \varphi_i dxdy \right]_{i=1}^{N_{bu}},$$

$$\boldsymbol{bN_3} = \left[\int_\Omega u_{1h}^{(k,l-1)} \frac{\partial u_{2h}^{(k,l-1)}}{\partial x} \varphi_i dxdy \right]_{i=1}^{N_{bu}}, \qquad \boldsymbol{bN_4} = \left[\int_\Omega u_{2h}^{(k,l-1)} \frac{\partial u_{2h}^{(k,l-1)}}{\partial y} \varphi_i dxdy \right]_{i=1}^{N_{bu}},$$

and $\quad \boldsymbol{bN_5} = \left[\int_\Omega c_1 \theta_h^{(k-1)} \varphi_i dxdy \right]_{i=1}^{N_{bu}},$

to generate

$$\boldsymbol{bN}_{\boldsymbol{u},p} = \begin{pmatrix} \boldsymbol{bN_1} + \boldsymbol{bN_2} \\ \boldsymbol{bN_3} + \boldsymbol{bN_4} + \boldsymbol{bN_5} \\ \boldsymbol{0} \end{pmatrix}. \tag{4.3}$$

Thus, for each step k, l $(k = 1, 2, ..., K; l = 1, 2, ..., L)$ of the iterations of step 1, we solve the linear algebraic system

$$A_{\boldsymbol{u},p}^{(k,l)} \boldsymbol{x^{(k,l)}} = \boldsymbol{b^{(k,l)}}_{\boldsymbol{u},p},$$

where $A_{\boldsymbol{u},p}^{(k,l)} = A_{\boldsymbol{u},p} + AN_{\boldsymbol{u},p}$, $\boldsymbol{x^{(k,l)}} = (u_{1h}^{(k,l)}, u_{2h}^{(k,l)}, p_h^{(k,l)})^T$ and $\boldsymbol{b^{(k,l)}}_{\boldsymbol{u},p} = \boldsymbol{b}_{\boldsymbol{u},p} + \boldsymbol{bN}_{\boldsymbol{u},p}$.

Similarly for step 2, we choose the test function $w_h = \psi_i$ $(i = 1, ..., N_{b\theta})$ in each iteration k $(k = 1, 2, ..., K)$ and derive the simplified equation

$$\sum_{j=1}^{N_{b\theta}} \theta_j^{(k)} \left(\int_\Omega u_{1h}^{(k-1)} \frac{\partial \psi_j}{\partial x} \psi_i dxdy + \int_\Omega u_{2h}^{(k-1)} \frac{\partial \psi_j}{\partial y} \psi_i dxdy \right.$$
$$\left. + \int_\Omega c_2 \frac{\partial \psi_j}{\partial x} \frac{\partial \psi_j}{\partial x} dxdy + \int_\Omega c_2 \frac{\partial \psi_j}{\partial y} \frac{\partial \psi_j}{\partial y} dxdy \right) = \int_\Omega f_\theta \psi_j dxdy.$$

Thus, exploiting the previous definitions, we get

$$A_\theta = A_{10} + A_{11} \quad \text{and} \quad \boldsymbol{b}_\theta = \boldsymbol{b_3}. \tag{4.4}$$

For the parts within the iteration, we define

$$AN_9 = \left[\int_\Omega u_{1h}^{(k-1)} \frac{\partial \psi_j}{\partial x} \psi_i dxdy \right]_{i,j=1}^{N_{b\theta}} \quad \text{and} \quad AN_{10} = \left[\int_\Omega u_{2h}^{(k-1)} \frac{\partial \psi_j}{\partial y} \psi_i dxdy \right]_{i,j=1}^{N_{b\theta}}$$

to generate

$$AN_\theta = AN_9 + AN_{10}. \tag{4.5}$$

Hence, in step 2 for each iteration k $(k = 1, 2, ..., K)$, we solve the linear algebraic system

$$A_\theta^{(k)} \theta_h^{(k)} = \boldsymbol{b}_\theta,$$

where $A_\theta^{(k)} = A_\theta + AN_\theta$.

4.1.5. Algorithms. Throughout the derivation of the decoupled finite element method, we revealed the similarity of the coupled and decoupled scheme. Nonetheless, we have to alter Algorithms 4 and 5. First of all, $\boldsymbol{bN_5}$ consists of a constant coefficient function and a finite element coefficient function. Thus, we modify Algorithm 4 into Algorithm 6. Next, the boundary condition for the velocity $\boldsymbol{u}$ in the Navier-Stokes equations

Algorithm 6 Assemble the linearized load vector $\boldsymbol{bN_5}$

function ASSEMBLELINEARIZEDVECTOR($fun\_c, fun\_f_{1h}, ..., p, q, r, s$)
  $b = sparse(N_b, 1)$
  for $n = 1, ..., N$ **do**
    for $\beta = 1, ..., N_{lb}$ **do**
      Compute $r = \int_{E_n} c \frac{\partial^{r+s} f_{1h}}{\partial x^r \partial y^s} \frac{\partial^{p+q} \psi_{n\beta}}{\partial x^p \partial y^q} dxdy$
      Add r to $b(T_b(\beta, n), 1)$
    end for
  end for
end function

Algorithm 7 Treat Dirichlet boundary condition for the velocity $\boldsymbol{u}$

function BOUNDARYTREATMENT($fun\_g_1, fun\_g_2, ..., A^{(k,l)}, \boldsymbol{b^{(k,l)}}, N_b$)
  for $n = 1, ..., number\_of\_boundarynodes\_for\_\boldsymbol{u}$ **do**
    Set i = index of current boundary node and
    $A^{(k,l)}(i, :) = 0$
    $A^{(k,l)}(N_b + i, :) = 0$
    $A^{(k,l)}(i, i) = 1$
    $A^{(k,l)}(N_b + i, N_b + i) = 1$
    $\boldsymbol{b^{(k,l)}}(i) = fun\_g_1(P_b(:, i))$
    $\boldsymbol{b^{(k,l)}}(N_b + i) = fun\_g_2(P_b(:, i))$
  end for
end function

Algorithm 8 Treat Dirichlet boundary condition for the temperature θ

function BOUNDARYTREATMENT($fun\_h, ..., A^{(k)}, \boldsymbol{b^{(k)}}$)
  for $n = 1, ..., number\_of\_boundarynodes\_for\_\theta$ **do**
    Set i = index of current boundary node and
    $A^{(k)}(i, :) = 0$
    $A^{(k)}(i, i) = 1$
    $\boldsymbol{b^{(k)}}(i) = fun\_h(P_b(:, i))$
  end for
end function

and for the temperature θ in the convection-diffusion equation need to be treated separately. To do this, we split Algorithm 5 into the two Algorithms 7 and 8 and apply them to $A_{\boldsymbol{u},p}^{(k,l)}$, $\boldsymbol{b}^{(\boldsymbol{k},\boldsymbol{l})}{}_{\boldsymbol{u},p}$ and $A_{\theta}^{(k)}$, $\boldsymbol{b}^{(\boldsymbol{k})}{}_{\theta}$, respectively.

To realize the derived decoupled finite element method, we structure our `MATLAB` code in the following way.

(i) Generate matrix $A_{\boldsymbol{u},p}$ and vector $\boldsymbol{b}_{\boldsymbol{u},p}$ as in (4.1) using Algorithms 1 and 2

(ii) Generate matrix A_{θ} and vector $\boldsymbol{b}_{\theta}$ as in (4.4) using Algorithms 1 and 2

(iii) FOR $k = 1, 2, ..., K$ (begin iteration for decoupling)

(iv) FOR $l = 1, 2, ..., L$ (begin Newton iteration for linearization)

(v) Generate matrix $AN_{\boldsymbol{u},p}$ as in (4.2) using Algorithm 3

(vi) Generate vector $\boldsymbol{bN}_{\boldsymbol{u},p}$ as in (4.3) using Algorithms 4 and 6

(vii) Set $A_{\boldsymbol{u},p}^{(k,l)} = A_{\boldsymbol{u},p} + AN_{\boldsymbol{u},p}$ and $\boldsymbol{b}^{(\boldsymbol{k},\boldsymbol{l})} = \boldsymbol{b}_{\boldsymbol{u},p} + \boldsymbol{bN}_{\boldsymbol{u},p}$

(viii) Deal with the Dirichlet boundary condition $\boldsymbol{u} = \boldsymbol{g}_{\boldsymbol{u}}$ on $\partial\Omega$ using Algorithm 7

(ix) Fix pressure $p_h^{(k,l)}$ at one point in Ω

(x) Solve $A_{\boldsymbol{u},p}^{(k,l)}\boldsymbol{x}^{(\boldsymbol{k},\boldsymbol{l})} = \boldsymbol{b}^{(\boldsymbol{k},\boldsymbol{l})}{}_{\boldsymbol{u},p}$ using the backslash operator

(xi) END

(xii) Generate matrix AN_{θ} as in (4.5) using Algorithm 3

(xiii) Set $A_{\theta}^{(k)} = A_{\theta} + AN_{\theta}$

(xiv) Deal with the Dirichlet boundary condition $\theta = g_{\theta}$ on $\partial\Omega$ using Algorithm 8

(xv) Solve $A_{\theta}^{(k)}\theta^{(k)} = \boldsymbol{b}_{\theta}$ using the backslash operator

(xvi) END

To clarify, step (i), (iv) - (xi) correspond to step 1 and step (ii), (xii) - (xv) to step 2. Thus, their order could be switched or they could be executed in parallel if suitable software is used. As we observed before, steps (i) and (ii) are independent of the iteration because they do not contain $u_{1h}^{(k-1)}$, $u_{2h}^{(k-1)}$ or $\theta_h^{(k-1)}$. Hence, $A_{\boldsymbol{u},p}$, $\boldsymbol{b}_{\boldsymbol{u},p}$, A_θ and $\boldsymbol{b}_\theta$ will be generated once and not altered within the loop. Contrariwise, all the other steps have to be executed in each loop.

4.2. DERIVATION OF THE SEQUENTIAL APPROACH

Another way of realizing the decoupled scheme is dealing with step 1 and step 2 sequentially, not in parallel. In other words, in the second step the solution of the first step is used. To avoid unnecessary repetition, we use the same definitions as in the previous section and just state the resulting formulations.

4.2.1. Weak Formulation. Depending on which variable our main focus is on, we either solve for the velocity $\boldsymbol{u}$ and pressure p first and then, based on that, for the temperature θ or we solve for θ first and then for $\boldsymbol{u}$ and p. The weak formulations of both ways are given as follows.

Initial guess: $\theta^{(0)}$

FOR $k = 1, 2, ..., K$

Step 1:
$$\begin{cases} \text{Find } \boldsymbol{u}^{(\boldsymbol{k})} \in \left(H^1(\Omega)\right)^2 \text{ and } p^{(k)} \in L^2(\Omega) \text{ such that} \\ c(\boldsymbol{u}^{(\boldsymbol{k})}, \boldsymbol{u}^{(\boldsymbol{k})}, \boldsymbol{v}) + a(\boldsymbol{u}^{(\boldsymbol{k})}, \boldsymbol{v}) + b(\boldsymbol{v}, p^{(k)}) - (\boldsymbol{e_2} c_1 \theta^{(k-1)}, \boldsymbol{v}) = (\boldsymbol{f_u}, \boldsymbol{v}), \\ b(\boldsymbol{u}^{(\boldsymbol{k})}, q) = 0, \\ \text{for any } \boldsymbol{v} \in \left(H_0^1(\Omega)\right)^2 \text{ and } q \in L^2(\Omega). \end{cases}$$

$$\text{Step 2:} \begin{cases} \text{Find } \theta^{(k)} \in H^1(\Omega) \text{ such that} \\ \tilde{c}(\boldsymbol{u}^{(k)}, \theta^{(k)}, w) + \tilde{a}(\theta^{(k)}, w) = (f_\theta, w), \\ \text{for any } w \in H_0^1(\Omega). \end{cases}$$

END

Initial guess: $\boldsymbol{u}^{(0)}$

FOR $k = 1, 2, ..., K$

$$\text{Step 1:} \begin{cases} \text{Find } \theta^{(k)} \in H^1(\Omega) \text{ such that} \\ \tilde{c}(\boldsymbol{u}^{(k-1)}, \theta^{(k)}, w) + \tilde{a}(\theta^{(k)}, w) = (f_\theta, w), \\ \text{for any } w \in H_0^1(\Omega). \end{cases}$$

$$\text{Step 2:} \begin{cases} \text{Find } \boldsymbol{u}^{(k)} \in \left(H^1(\Omega)\right)^2 \text{ and } p^{(k)} \in L^2(\Omega) \text{ such that} \\ c(\boldsymbol{u}^{(k)}, \boldsymbol{u}^{(k)}, \boldsymbol{v}) + a(\boldsymbol{u}^{(k)}, \boldsymbol{v}) + b(\boldsymbol{v}, p^{(k)}) - (\boldsymbol{e_2} c_1 \theta^{(k)}, \boldsymbol{v}) = (\boldsymbol{f_u}, \boldsymbol{v}), \\ b(\boldsymbol{u}^{(k)}, q) = 0, \\ \text{for any } \boldsymbol{v} \in \left(H_0^1(\Omega)\right)^2 \text{ and } q \in L^2(\Omega). \end{cases}$$

END

In addition, for this technique we observe a different kind of flexibility. If an approximation either of the velocity $\boldsymbol{u}$ and the pressure p or of the temperature θ is already known or we are only interested in one of the variables, the scheme can be easily adjusted and the running time performance increased.

4.2.2. Galerkin Formulation and Newton Iteration. Using the same finite element spaces and global basis functions as before and nonlinearizing the Navier-Stokes equations with the Newton iteration, the following Galerkin formulations are obtained.

Solving for the velocity $\boldsymbol{u}$ and the pressure p first yields

Initial guess: $\theta_h^{(0)}$

FOR $k = 1, 2, ..., K$

Step 1:
$$\begin{cases} \text{Initial guess: } \boldsymbol{u_h^{(k,0)}} \\ \text{FOR } l = 1, 2, ..., L \\ \quad \text{Find } \boldsymbol{u_h^{(k,l)}} \in (U_h)^2 \text{ and } p_h^{(k,l)} \in P_h \text{ such that} \\ \quad c(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) + a(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) \\ \quad +b(\boldsymbol{v_h}, p_h^{(k,l)}) - (\boldsymbol{e_2} c_1 \theta_h^{(k-1)}, \boldsymbol{v_h}) = (\boldsymbol{f_u}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}), \\ \quad b(\boldsymbol{u_h^{(k,l)}}, q_h) = 0, \\ \quad \text{for any } \boldsymbol{v_h} \in (U_h)^2 \text{ and } q_h \in P_h. \\ \text{END} \end{cases}$$

Step 2:
$$\begin{cases} \text{Find } \theta^{(k)} \in W_h \text{ such that} \\ \tilde{c}(\boldsymbol{u_h^{(k)}}, \theta_h^{(k)}, w_h) + \tilde{a}(\theta_h^{(k)}, w_h) = (f_\theta, w_h), \\ \text{for any } w_h \in W_h. \end{cases}$$

END,

and vice versa, solving for the temperature θ first provides

Initial guess: $\boldsymbol{u_h^{(0)}}$

FOR $k = 1, 2, ..., K$

Step 1:
$$\begin{cases} \text{Find } \theta^{(k)} \in W_h \text{ such that} \\ \tilde{c}(\boldsymbol{u_h^{(k-1)}}, \theta_h^{(k)}, w_h) + \tilde{a}(\theta_h^{(k)}, w_h) = (f_\theta, w_h), \\ \text{for any } w_h \in W_h. \end{cases}$$

Step 2:
$$\begin{cases} \text{Initial guess: } \boldsymbol{u_h^{(k,0)}} \\ \text{FOR } l = 1, 2, ..., L \\ \quad \text{Find } \boldsymbol{u_h^{(k,l)}} \in (U_h)^2 \text{ and } p_h^{(k,l)} \in P_h \text{ such that} \\ \quad c(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) + a(\boldsymbol{u_h^{(k,l)}}, \boldsymbol{v_h}) \\ \quad +b(\boldsymbol{v_h}, p_h^{(k,l)}) - (\boldsymbol{e_2} c_1 \theta_h^{(k)}, \boldsymbol{v_h}) = (\boldsymbol{f_u}, \boldsymbol{v_h}) + c(\boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{u_h^{(k,l-1)}}, \boldsymbol{v_h}), \\ \quad b(\boldsymbol{u_h^{(k,l)}}, q_h) = 0, \\ \quad \text{for any } \boldsymbol{v_h} \in (U_h)^2 \text{ and } q_h \in P_h. \\ \text{END} \end{cases}$$

END.

4.2.3. Finite Element Discretization. Either way, compared to the parallel approach, the modification of the equations of the linear algebraic system and, therefore, of the matrix formulation is small. If we solve for the velocity $\boldsymbol{u_h^{(k)}}$ and the pressure $p_h^{(k)}$ first, we have to change AN_9 and AN_{10} to

$$AN_9 = \left[\int_\Omega u_{1h}^{(k)} \frac{\partial \psi_j}{\partial x} \psi_i dxdy \right]_{i,j=1}^{N_{b\theta}} \quad \text{and} \quad AN_{10} = \left[\int_\Omega u_{2h}^{(k)} \frac{\partial \psi_j}{\partial y} \psi_i dxdy \right]_{i,j=1}^{N_{b\theta}}.$$

On the other hand, if we solve for the temperature $\theta_h^{(k)}$ first, we change $\boldsymbol{bN_5}$ to

$$\boldsymbol{bN_5} = \left[\int_\Omega c_1 \theta_h^{(k)} \varphi_i dxdy \right]_{i=1}^{N_{bu}}.$$

4.3. NUMERICAL EXAMPLE

The purpose of this section is to verify our derivation of the decoupled finite element methods by a numerical example. To make the results comparable to the results of Section 3, we solve the same Boussinesq Equations (3.12). We recall that the analytical solutions

were given by

$$
\begin{aligned}
u_1 &= x^2y^2 + e^{-y}, \\
u_2 &= -\frac{2}{3}xy^3 + 2 - \pi\sin(\pi x), \\
p &= -(2 - \pi\sin(\pi x))\cos(2\pi y), \quad \text{and} \\
\theta &= e^{x+y}.
\end{aligned}
$$

Again, Equation (3.12) is solved on the domain $\Omega = [0,1] \times [-0.25, 0]$ for six different meshes with edge sizes $h \in \left\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\right\}$, as shown in Figure 3.2. The gained results are stated in Tables 4.1, 4.2, 4.3 and 4.4 for the maximum error, the L^∞ norm error,

Table 4.1. The maximum errors and corresponding convergence rate of decoupled parallel Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	MaxError $\boldsymbol{u}$	Rate	MaxError p	Rate	MaxError θ	Rate
4	$1.4696 \cdot 10^{-3}$	–	$6.0149 \cdot 10^{-1}$	–	$2.6162 \cdot 10^{-5}$	–
8	$4.5401 \cdot 10^{-4}$	1.6946	$2.0711 \cdot 10^{-1}$	1.5381	$1.9435 \cdot 10^{-6}$	3.7507
16	$3.7082 \cdot 10^{-5}$	3.6139	$5.5743 \cdot 10^{-2}$	1.8935	$1.1358 \cdot 10^{-7}$	4.0969
32	$2.3809 \cdot 10^{-6}$	3.9612	$1.3928 \cdot 10^{-2}$	2.0008	$8.5492 \cdot 10^{-9}$	3.7318
64	$1.8440 \cdot 10^{-7}$	3.6906	$3.4684 \cdot 10^{-3}$	2.0057	$5.9340 \cdot 10^{-10}$	3.8487
128	$2.1410 \cdot 10^{-8}$	3.1065	$8.6461 \cdot 10^{-4}$	2.0042	$3.8963 \cdot 10^{-11}$	3.9288

Table 4.2. The L^∞ errors and corresponding convergence rate of decoupled parallel Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	InfError $\boldsymbol{u}$	Rate	InfError p	Rate	InfError θ	Rate
4	$1.1759 \cdot 10^{-2}$	–	$3.7781 \cdot 10^{-1}$	–	$3.0632 \cdot 10^{-4}$	–
8	$1.6853 \cdot 10^{-3}$	2.8026	$1.3616 \cdot 10^{-1}$	1.4724	$4.0349 \cdot 10^{-5}$	2.9244
16	$2.0224 \cdot 10^{-4}$	3.0589	$4.5862 \cdot 10^{-2}$	1.5699	$5.1810 \cdot 10^{-6}$	2.9612
32	$2.5167 \cdot 10^{-5}$	3.0065	$1.2533 \cdot 10^{-2}$	1.8716	$6.5635 \cdot 10^{-7}$	2.9807
64	$3.1048 \cdot 10^{-6}$	3.0190	$3.2510 \cdot 10^{-3}$	1.9468	$8.2594 \cdot 10^{-8}$	2.9904
128	$3.8464 \cdot 10^{-7}$	3.0129	$8.2650 \cdot 10^{-4}$	1.9758	$1.0359 \cdot 10^{-8}$	2.9952

Table 4.3. The L^2 errors and corresponding convergence rate of decoupled parallel Galerkin FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	L2Error $\boldsymbol{u}$	Rate	L2Error p	Rate	L2Error θ	Rate
4	$2.7603 \cdot 10^{-3}$	–	$8.2694 \cdot 10^{-2}$	–	$6.9312 \cdot 10^{-5}$	–
8	$3.5640 \cdot 10^{-4}$	2.9532	$2.2577 \cdot 10^{-2}$	1.8729	$8.5992 \cdot 10^{-6}$	3.0108
16	$4.4016 \cdot 10^{-5}$	3.0174	$8.6669 \cdot 10^{-3}$	1.3813	$1.0718 \cdot 10^{-6}$	3.0042
32	$5.4798 \cdot 10^{-6}$	3.0058	$2.4764 \cdot 10^{-3}$	1.8073	$1.3389 \cdot 10^{-7}$	3.0009
64	$6.8421 \cdot 10^{-7}$	3.0016	$6.5584 \cdot 10^{-4}$	1.9168	$1.6733 \cdot 10^{-8}$	3.0002
128	$8.5497 \cdot 10^{-8}$	3.0005	$1.6841 \cdot 10^{-4}$	1.9613	$2.0916 \cdot 10^{-9}$	3.0001

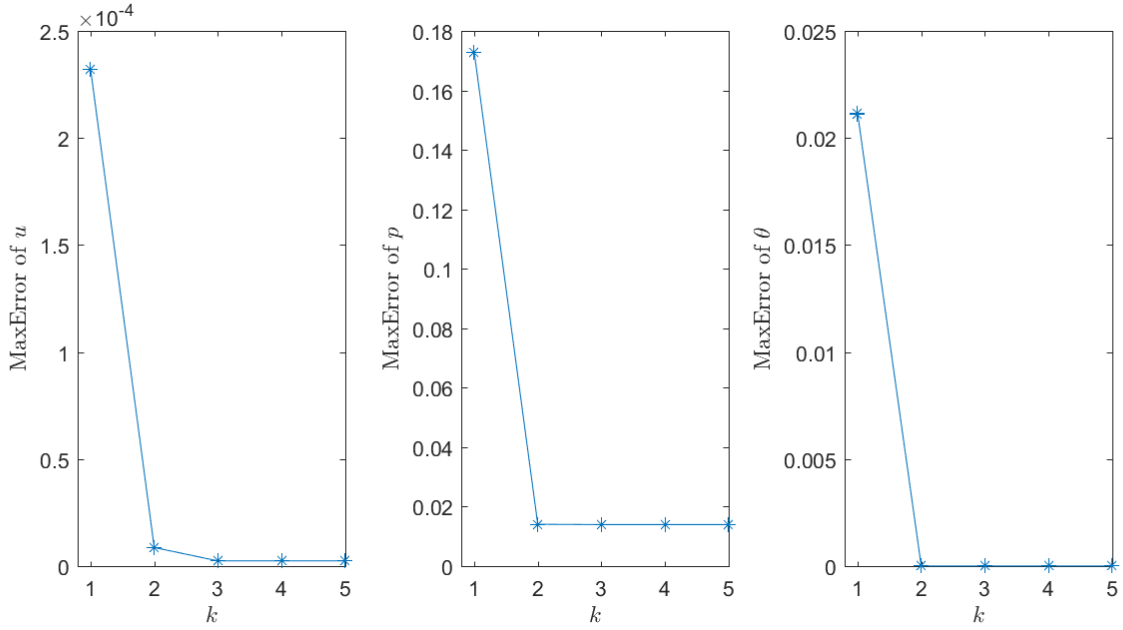
Table 4.4. The H^1 errors and corresponding convergence rate of decoupled parallel FEM for the steady Boussinesq Equations (3.12) with Taylor-Hood finite elements for velocity $\boldsymbol{u}$ and pressure p and quadratic finite elements for temperature θ

$\frac{1}{h}$	H1Error $\boldsymbol{u}$	Rate	H1Error p	Rate	H1Error θ	Rate
4	$8.0560 \cdot 10^{-2}$	–	$2.3556 \cdot 10^{0}$	–	$2.6175 \cdot 10^{-3}$	–
8	$2.0429 \cdot 10^{-2}$	1.9795	$1.2648 \cdot 10^{0}$	0.8972	$6.5321 \cdot 10^{-4}$	2.0025
16	$5.0681 \cdot 10^{-3}$	2.0111	$6.3069 \cdot 10^{-1}$	1.0039	$1.6323 \cdot 10^{-4}$	2.0006
32	$1.2623 \cdot 10^{-3}$	2.0054	$3.1369 \cdot 10^{-1}$	1.0076	$4.0804 \cdot 10^{-5}$	2.0001
64	$3.1523 \cdot 10^{-4}$	2.0016	$1.5658 \cdot 10^{-1}$	1.0024	$1.0201 \cdot 10^{-5}$	2.0000
128	$7.8782 \cdot 10^{-5}$	2.0004	$7.8254 \cdot 10^{-2}$	1.0007	$2.5502 \cdot 10^{-6}$	2.0000

the L^2 norm error and H^1 semi-norm error, respectively. At first glance, we recognize that those results coincide with the results of the previous section. Additionally, we get exactly the same results if we use the sequential techniques. Therefore, no matter which scheme we use, we solve the given Boussinesq equations with optimal convergence rates.

4.4. EVALUATION OF INTRODUCED METHODS

Finally, to summarize our findings, we contrast the derived methods. As we do not use parallel coding in our `MATLAB` realization, the measurement of the CPU time is unnecessary. Therefore, we analyze the decoupled schemes one by one in terms of iteration performance. That means, we solve the example in Section 4.3 three times

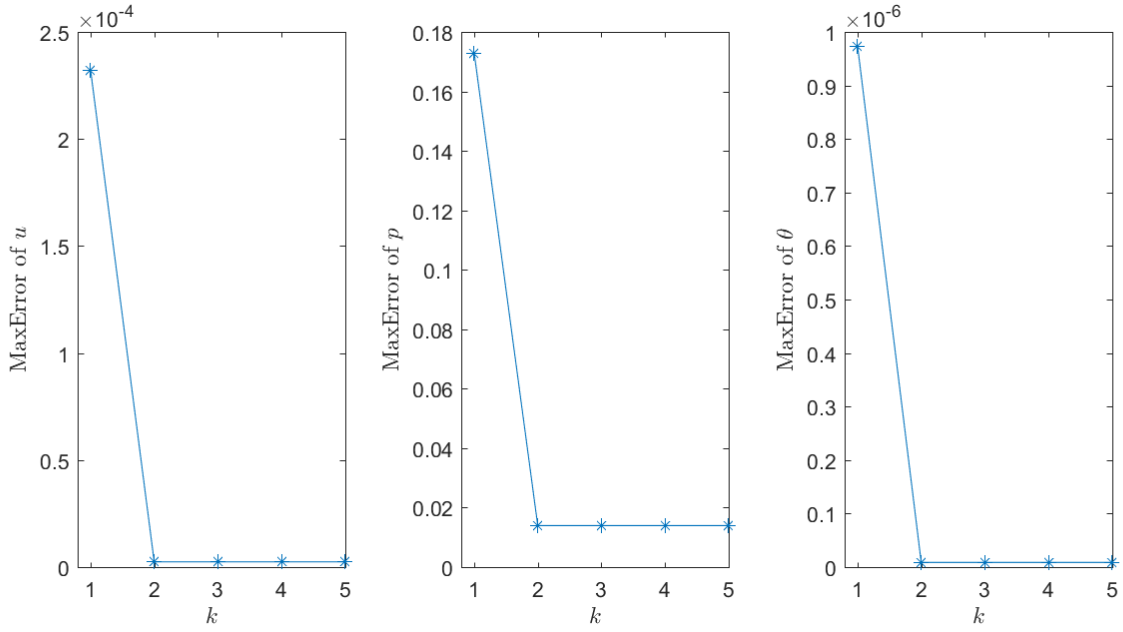


(a) Error of the velocity $\boldsymbol{u}$ (b) Error of the pressure p (c) Error of the temperature θ

Figure 4.1. The maximum error of the velocity $\boldsymbol{u}$, the pressure p and the temperature θ in each iteration step k $(k = 1, 2, ..., 5)$ of the decoupled parallel FEM with mesh edge size $h = \frac{1}{32}$

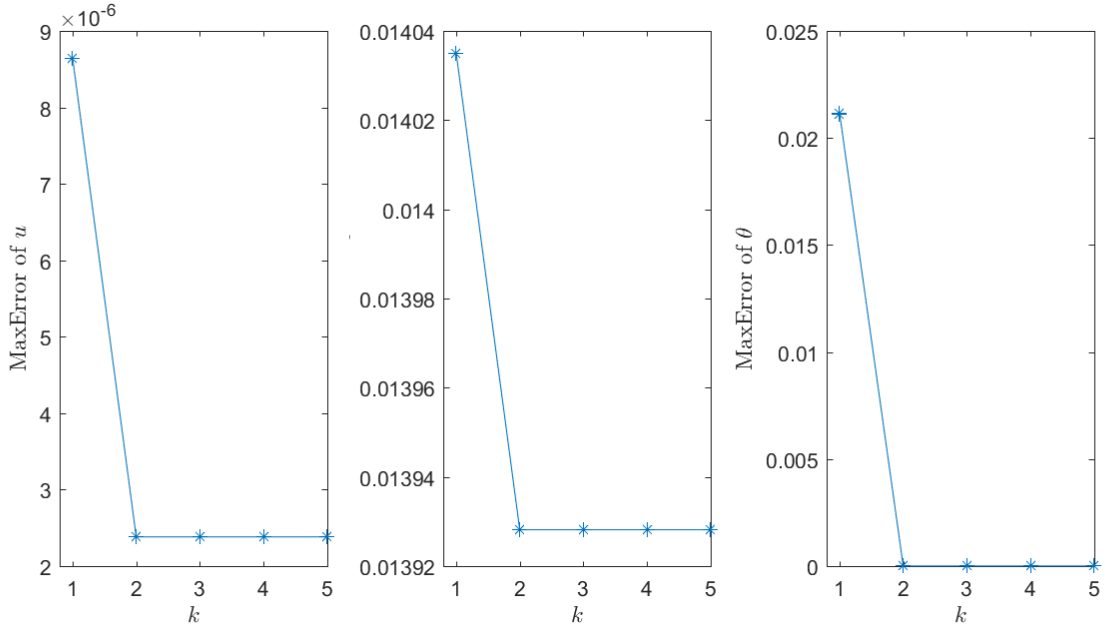
using three different methods, and for each method we calculate after each iteration step k $(k = 1, 2, 3, 4, 5)$ the maximum errors $\max|\boldsymbol{u} - \boldsymbol{u_h}|$, $\max|p - p_h|$ and $\max|\theta - \theta_h|$. The resulting plots for the mesh edge size $h = \frac{1}{32}$ are shown in Figures 4.1, 4.2 and 4.3.

We start with the parallel decoupled scheme derived in Section 4.1. After the first iteration, the maximum error of the velocity $\boldsymbol{u}$ amounts to $2.3208 \cdot 10^{-4}$. While this error is unacceptably high, a rapid drop can be observed after the second iteration to $8.6436 \cdot 10^{-6}$ and then $2.3809 \cdot 10^{-6}$ for $k = 3$, which is already the optimal error stated in Table 4.1. The same pattern can be found for the pressure p, where we start at $1.7287 \cdot 10^{-1}$ after the first iteration, drop to $1.4035 \cdot 10^{-2}$ and reach the optimal error $1.3928 \cdot 10^{-2}$ after the third iteration. For the temperature θ, we measure the steepest decline from the first to the second iteration. The maximum errors are $2.1146 \cdot 10^{-2}$, $9.7283 \cdot 10^{-}7$, $3.5588 \cdot 10^{-8}$ and $8.5492 \cdot 10^{-9}$ for $k = 1$, 2, 3 and 4, respectively. Thus, we need one more iteration



(a) Error of the velocity $\boldsymbol{u}$ (b) Error of the pressure p (c) Error of the temperature θ

Figure 4.2. The maximum error of the velocity $\boldsymbol{u}$, the pressure p and the temperature θ in each iteration step k $(k = 1, 2, ..., 5)$ of the decoupled sequential FEM with mesh edge size $h = \frac{1}{32}$, where we solve the Navier-Stokes equations first



(a) Error of the velocity $\boldsymbol{u}$ (b) Error of the pressure p (c) Error of the temperature θ

Figure 4.3. The maximum error of the velocity $\boldsymbol{u}$, the pressure p and the temperature θ in each iteration step k $(k = 1, 2, ..., 5)$ of the decoupled sequential FEM with mesh edge size $h = \frac{1}{32}$, where we solve the convection-diffusion equation first

step to reach the optimal error, but we observe that the difference between the third and the fourth iteration step is negligibly low. In conclusion, for the parallel decoupled finite element method aiming for sufficiently accurate results, the maximum number of iterations K should not be higher than 4, in terms of efficiency advisably set as 3.

In Section 4.2 two decoupled methods, distinguishable by the equation which is first solved, were derived. Figure 4.2 shows the maximum error after each iteration step for the sequential decoupled finite element method solving the Navier-Stokes equations first. After the first iteration step, the velocity $\boldsymbol{u}$ and the pressure p exhibit errors of $2.3208 \cdot 10^{-4}$ and $1.7287 \cdot 10^{-1}$, respectively, exactly as observed for the parallel approach. As expected, the error for the temperature θ is with $9.7283 \cdot 10^{-7}$ considerably smaller than the one for the parallel approach. Furthermore, for all variables the optimal error as stated in Table 4.1 is reached after the second iteration.

Contrariwise, using the sequential decoupled finite element method solving the convection-diffusion equation first, for θ the same maximum error $2.1146 \cdot 10^{-2}$ at $k = 1$ as in the parallel approach is observable (see Figure 4.3). For $\boldsymbol{u}$ and p the reduced errors amount to $8.6436 \cdot 10^{-6}$ and $1.4035 \cdot 10^{-2}$, respectively, and the optimal errors are reached after the second iteration step, again. Similar to the parallel approach, θ needs one iteration step more as it shows the errors $3.5588 \cdot 10^{-8}$ and $8.5492 \cdot 10^{-9}$ for $k = 2$ and 3. Therefore, the maximum number of iterations K should not be higher than 3 and advisably set as 2 for both sequential decoupled finite element methods.

Additionally, if the variables of interest are either only the velocity $\boldsymbol{u}$ and the pressure p or only the temperature θ, it might be a preferable option to choose $K = 1$ and the corresponding sequential approach as the loss in accuracy is low compared to the gain in efficiency. On the other hand, if we care about both accuracy and efficiency we can choose $K = 2$ and the corresponding sequential approach, but stop after step 1 in the second iteration. This way, we solve one linear algebraic system less compared to the original approach.

5. CONCLUSIONS

In this work, we have designed two decoupled finite element methods for a generalized steady two-dimensional Boussinesq problem with Dirichlet boundary condition. We considered the derivation of the standard Galerkin finite element method using Taylor-Hood finite elements for the fluid variables and quadratic finite elements for the temperature variable as a foundation for the new decoupling methods. This coupled method was applied to a numerical example and the error convergence in different norms was measured.

The idea of decoupling the original problem into the Navier-Stokes equations and a parabolic problem was realized by introducing a new iteration and solving the equations either in parallel or sequentially while all other parameters remained the same. During the derivation, the increased flexibility and convenience of the new methods came to light. After confirming the error convergence of the new methods in different norms by the numerical example, the performance of the decoupled schemes was judged by the necessary number of iterations. Three iteration steps for the parallel and two iteration steps for the sequential approach turned out to be recommendable. Especially for the sequential decoupled finite element method, the performance could be improved if we are mainly focused on one specific variable.

Ensuing works could not only realize the derived parallel decoupled finite element method using parallel coding and evaluating the CPU time, but also prove the existence and uniqueness of the solution of the derived methods and analyze their stability and convergence.

REFERENCES

[1] K. Allali. A priori and a posteriori error estimates for boussinesq equations. *International Journal of Numerical Analysis and Modeling*, 2(2):179–196, 2005.

[2] A. Allendes, C. Naranjo, and E. Otárola. Stabilized finite element approximations for a generalized boussinesq problem: A posteriori error analysis. *Computer Methods in Applied Mechanics and Engineering*, 361:112703, 2020.

[3] J. A. Almonacid, G. N. Gatica, and R. Oyarzúa. A posteriori error analysis of a mixed-primal finite element method for the boussinesq problem with temperature-dependent viscosity. *Journal of Scientific Computing*, 78(2):887–917, 2019.

[4] C. Bernardi, B. Métivet, and B. Pernaud-Thomas. Couplage des équations de navier–stokes et de la chaleur: le modèle et son approximation par éléments finis. *RAIRO Modélisation mathématique et analyse numérique*, 29:871–921, 1995.

[5] J. Boland and W. Layton. Error analysis for finite element methods for steady natural convection problems. *Numerical Functional Analysis and Optimization*, 11:449–483, 1990.

[6] J. Deteix, A. Jendoubi, and D. Yakoubi. A coupled prediction scheme for solving the navier–stokes and convection-diffusion equations. *SIAM Journal on Numerical Analysis*, 52(5):2415–2439, 2014.

[7] P. G. Drazin and W. H. Reid. *Hydrodynamic stability. Cambridge Monographs on Mechanics and Applied Mathematics*. Cambridge University Press, Cambridge-New York, 1981.

[8] H. Elman, M. Mihajlović, and D. Silvester. Fast iterative solvers for buoyancy driven flow problems. *Journal of Computational Physics*, 230(10):3900–3914, 2011.

[9] V. Girault and P.-A. Raviart. Finite element methods for navier-stokes equations. theory and algorithms. *Springer Series in Computational Mathematics*, 5, 1986.

[10] P. M. Gresho, R. L. Lee, S. T. Chan, and R. L. Sani. Solution of the time-dependent incompressible navier-stokes and boussinesq equations using the galerkin finite element method. *Approximation methods for Navier-Stokes problems*, 771:203–222, 1980.

[11] P. Huang. An efficient two-level finite element algorithm for the natural convection equations. *Applied Numerical Mathematics*, 118:75–86, 2017.

[12] L. M. Jiji. *Heat Convection*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[13] V. John. *Finite Element Methods for Incompressible Flow Problems*, volume 51. Springer International Publishing, 2016.

[14] Z. D. Luo and X. M. Lu. A least-squares galerkin/petrov mixed finite element method for stationary conduction-convection problems. *Mathematica Numerica Sinica*, 25:231–244, 2003.

[15] R. Oyarzúa and P. Zúñiga. Analysis of a conforming finite element method for the boussinesq problem with temperature-dependent parameters. *Journal of Computational and Applied Mathematics*, 323:71–94, 2017.

[16] Z. Si, Y. Shang, and T. Zhang. New one- and two-level newton iterative mixed finite element methods for stationary conduction–convection problems. *Finite Elements in Analysis and Design*, 47(2):175–183, 2011.

[17] C. Taylor and P. Hood. A numerical solution of the navier-stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.

[18] T. Zhang, T. Jiang, and J. Yuan. One-level and two-level finite element methods for the boussinesq problem. *Mathematical Methods in the Applied Sciences*, 41(7):2748–2768, 2018.

[19] T. Zhang and H. Liang. Decoupled stabilized finite element methods for the boussinesq equations with temperature-dependent coefficients. *International Journal of Heat and Mass Transfer*, 110:151–165, 2017.

[20] T. Zhang, X. Zhao, and P. Huang. Decoupled two level finite element methods for the steady natural convection problem. *Numerical Algorithms*, 68(4):837–866, 2015.

[21] K. Zhao. 2d inviscid heat conductive boussinesq equations on a bounded domain. *The Michigan Mathematical Journal*, 59(2):329–352, 08 2010.

[22] A. Çıbık and S. Kaya. A projection-based stabilized finite element method for steady-state natural convection problem. *Journal of Mathematical Analysis and Applications*, 381(2):469–484, 2011.

VITA

Lioba Boveleth was born in Mönchengladbach, Germany, on May 30, 1996. She graduated from Ulm University with a B.S. in Mathematics and Economics in July 2018. During her Bachelor degree, she studied the winter semester 2017/18 abroad at Charles University in Prague, Czech Republic, and interned at Commerzbank in Frankfurt am Main in the summer of 2018. By July 2019 she had completed her first two semesters of her master's degree in Mathematics with a minor in Computer Science at Ulm University. In Fall 2019, she enrolled at Missouri University of Science and Technology being part of a double degree program between Ulm University and Missouri University of Science and Technology. During her studies at both universities, she was employed as a graduate teaching assistant focusing on teaching labs. She received her master's degree in Applied Mathematics from Missouri University of Science and Technology in May 2020 and continued toward her master's degree at Ulm University.