Numerical Construction of Diffeomorphism and the Applications to Grid Generation and Image Registration

by

XI CHEN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

Aug 2016

To my mother Lihua Wang,

and my father Huaizhen Chen

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Guojun Liao for constantly motivating and encouraging me, and also for his invaluable advice during the 5 years of my doctoral studies. I wish to thank my academic advisors Dr. Benito Chen, Dr. Rencang Li, Dr. Chaoqun Liu and Dr. Jianzhong Su for their interest in my research and for taking time to serve in my graduate committee.

I would also like to extend my appreciation to our department of mathematics lead previously by Dr. Jianping Zhu and now by Dr. Jianzhong Su, for supporting my doctoral studies. I wish to thank all the professors of our department who taught me in these years, especially Dr. Benito Chen, Dr. Rencang Li and Dr. Gaik Ambartsoumian, from whose classes I benefitted most.

I am grateful to all the teachers who taught me during the exactly 20 years I spent in schools, in China and the Unites States. Especially I would like to thank Mr. Benyan Dou, my math teacher in middle school, who taught and encouraged me to self-study when I was a 10-year-old boy. Without him, I wouldn't be so interested in mathematics and choose it as my major. He inspired me that studying is a lifetime journey, and real studying is to study by myself.

I would like to express my deep gratitude to all the schools and universities I attended, which are Hefei ShiYan school, Hefei No.1 high school, University of Science and Technology of China, and UT Arlington. I am proud to be one part of these

great schools and universities. I wish they will be proud of me in the future.

Finally, I would like to say thank you to all my friends who have helped and companied me in my time of living in the United States. I am extremely fortunate to meet you.

May 27, 2016

ABSTRACT

Numerical Construction of Diffeomorphism
and the Applications to Grid Generation and Image Registration

Xi Chen, Ph.D.
The University of Texas at Arlington, 2016

Supervising Professor: Guojun Liao

Diffeomorphism is an active research topic in differential geometry. In this area, the existence and construction of diffeomorphism under certain constraints is an interesting and meaningful task. J. Moser first proved the existence of diffeomorphism under a Jacobian determinant constraint. Later, Dr. Liao along with his co-authors, proposed the deformation method to construct diffeomorphisms. A div-curl system is created in the construction of diffeomorphisms. Since the Jacobian determinant has a direct physical meaning in grid generation, i.e. the grid cell size, the deformation method was applied successfully to grid generation and adaptation problems.

In this dissertation, we review the deformation method, focus again on the construction of diffeomorphisms, address clearly a new formation of the deformation problem especially for moving domains. In theory, the deformation method provides one diffeomorphic solution to a nonlinear differential equation.

Inspired by the div-curl system in the deformation method, we developed a new method to construct diffeomorphisms, through a completely different approach. The idea is to control directly the Jacobian determinant and the curl vector of a transformation. Based on calculus of variation and optimization, we proposed a new variational method with prescribed Jacobian determinant and curl vector.

In the study of the two methods of diffeomorphisms construction, we observed the important role of the Jacobian determinant and the curl vector in determining a diffeomorphism. Hence, the corresponding uniqueness problem deserves an investigation. In this dissertation, we discuss this problem by both numerical experiments and theoretical analysis.

Last, we turn to non-rigid image registration, which shares the basic idea of finding transformations. The same equations for divergence and curl vectors are used as constraints to minimize a similarity measure. We designed graphical user interfaces for grid generation and image registration to demonstrate all the methods discussed in this dissertation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 From diffeomorphism to the deformation method of grid generation

In differential geometry, a diffeomorphism is map between manifolds which is differentiable and has a differentiable inverse, and it is a active research topic. In 1990, B.Dacorogna and J.Moser [1] proved the existence of diffeomorphisms $\phi : \Omega \to \Omega$ such that

$$\begin{cases} \det\nabla\phi(\xi) = f(\xi) & \xi \in \Omega, \\ \phi(\xi) = \xi & \xi \in \partial\Omega. \end{cases} \tag{1.1}$$

Since 1992-1993, G. Liao, collaborated with D. Anderson [2], J.Su [3, 4], H. Liu [5], F. Liu[6, 7], S.Osher [7], D.Fleitas [8], G. Pena [7, 9], Z.Lei[10], improved B.Dacorogna and J. Moser's technique, and proposed *the deformation method of grid generation*. In this method, a diffeomorphism $\phi : \Omega \to \Omega$ is numerically constructed, such that ϕ maps a grid of Ω to a new grid of Ω, whose grid sizes (approximated by the Jacobian determinant of ϕ, namely $\det\nabla\phi$) equal to a prescribed scalar monitor function $f > 0$ at $\phi(\xi)$. Correspondingly, the PDE 1.1 is extended to the following one, which is more meaningful in the physical domain:

$$\begin{cases} \det\nabla\phi(\xi) = f(\phi(\xi)) & \xi \in \Omega, \\ \phi(\xi) = \xi & \xi \in \partial\Omega. \end{cases} \tag{1.2}$$

This is the steady version of *the deformation method*. A series of applications were made, including adaptive moving grid [11, 12, 10], and steady euler flow calculations [6]. In the meantime, the dynamic version of *the deformation method* on fixed domains was developed in [7],[9],[13], based on solving Poisson's equations.

A milestone occurred in 2004. The Least-Squares Finite Element Method was first applied to solve the *div-curl system* in [14](see (2.10)), which extends *the deformation method of grid generation* to moving domains. This version constructs numerically a diffeomorphism $\phi : \Omega_0 \to \Omega_t$, such that for $\forall t \in [0, 1]$

$$\begin{cases} \det\nabla\phi(\xi, \mathrm{t}) = f(\phi(\xi, t), t) & \xi \in \Omega_0, \\ \phi(\xi, 0) = \xi, & \\ \phi(\xi, t) \in \partial\Omega_t & \xi \in \partial\Omega_0. \end{cases} \tag{1.3}$$

In the first part of this dissertation (chapter 2), we review the mathematical foundation of *the deformation method of grid generation*, and propose a new formation of the moving domain problem.

1.2 From the deformation method to a new variational method

The success of *the deformation method of grid generation* actually relies on the *div-curl system*, it makes us notice the significant importance of the curl vector ($\nabla\times$) in the construction of a diffeomorphism. Therefore, we tried to explore another completely different approach by directly controlling the Jacobian determinant ($\det\nabla$) and the curl vector ($\nabla\times$). We used calculus of variation to formulate a *new variational method with prescribed Jacobian determinant and curl vector* of constructing transformations[15]. A recovering experiment was designed by the *new variational method*, which leads to a potential uniqueness problem about transformations with the same Jacobian determinant and curl vector. In the second part of this dissertation (chapter 3 and 4), we discuss the *new variational method* and study the uniqueness problem.

1.3 From diffeomorphism to non-rigid image registration

Image registration is the process of establishing a one-to-one correspondence between pixels of two images such that a similarity measure (energy function) is optimized. It's about to construct transformations. In order to determine the one-to-one correspondence accurately and efficiently, we can optimize a similarity measure under some similar constraints to the *div-curl system* of *the deformation method*. In 2008, G. Liao proposed optimal control approach in [16] and further developed it in [17]. Our Image registration algorithms, along with a symmetric scheme are described in the third part of this dissertation (chapter 5).

1.4 Organization of the dissertation

There are 5 chapters in this dissertation. Chapter 1 is the Introduction. Chapter 2 presents a note on *the deformation method* for moving domains. Chapter 3 proposes a *new variational method*. Chapter 4 studies a uniqueness problem. Chapter 5 focuses on image registration problems.

CHAPTER 2

A NOTE ON THE DEFORMATION METHOD FOR MOVING DOMAINS

2.1 Introduction

The deformation method of grid generation has been extended to moving domain problem since 2004, when LSFEM was introduced to solve the *div-curl system*[14]. As a competitive method in solving practical grid generation and adaptation problems, *the deformation method of grid generation* usually focuses on constructing one diffeomorphism. Namely, for a given monitor function $f_0(\boldsymbol{x})$, let $f(\boldsymbol{x},t) = 1-t+tf_0(\boldsymbol{x})$ on $t\in[0,1]$, apply *the deformation method* to construct $\boldsymbol{\phi}(\boldsymbol{\xi},1)$ such that $J(\boldsymbol{\phi}(\boldsymbol{\xi},1)) = f_0(\boldsymbol{\phi}(\boldsymbol{\xi},1))$. (Note: $J=\det\nabla$, means the Jacobian determinant, same hereinafter). The intermediate steps are not cared, and the difference between $J(\boldsymbol{\phi}(\boldsymbol{\xi},1))$ and $f_0(\boldsymbol{\phi}(\boldsymbol{\xi},1))$ is not carefully examined. Also there are some ambiguities in boundary conditions.

Here we want to review the *the deformation method* back again as a mathematical method to construct diffeomorphisms. We carefully address the conditions and equations, clearly set up the boundary conditions, examine the domain issues, moreover, prove the existence of the solution. Indeed, *the deformation method* is a method to construct a family of diffeomorphisms $\boldsymbol{\phi}(\boldsymbol{x},t)$ with property $J(\boldsymbol{\phi}) = f(\boldsymbol{\phi},t)$ for a given monitor function $f(\boldsymbol{x},t)$, and the property is guaranteed to be true for any t.

Before starting our review of *the deformation method*, let's refresh the problem we study again by the following illustration. Given domains Ω_t and a monitor function $f(\boldsymbol{x},t)$, we want to construct diffeomorphisms $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ from Ω_0 to Ω_t, such that $J(\boldsymbol{\phi}(\boldsymbol{\xi},t)) = f(\boldsymbol{\phi}(\boldsymbol{\xi},t))$.

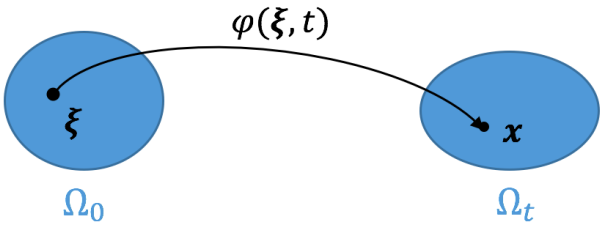


Figure 2.1. The problem we study.

We now start our review in one-dimensional case (1D) first, then move to general two- and three-dimensional cases.

2.2 One-dimensional case

2.2.1 Problem formulation

Let $\Omega_t := [a(t), b(t)] \subset \mathbb{R}$ be smoothly moving domains, i.e. $a(t), b(t)$ is differentiable, with $\Omega_0 = [a, b]$. Given a function $f(x, t) \in C^1(x, t) > 0$ on domain of $(x, t) : \Omega_t \times [0, 1]$, such that

$$f(x, 0) \quad = 1, \tag{2.1}$$

$$\int_{\Omega_t} \frac{1}{f(x,t)} dx \quad = |\Omega_0| = b - a. \tag{2.2}$$

A diffeomorphism

$$\phi(\xi, t) : \Omega_0 \to \Omega_t$$

such that $\forall t \in [0, 1]$

$$\phi_\xi(\xi, t) = f(\phi(\xi, t), t) \tag{2.3}$$

can be constructed by solving the following 2 differential equations (2.4) and (2.5).

- First, determine $u(x,t)$ on $\mathbb{R}$ by solving:

$$\begin{cases} u_x = -\dfrac{\partial}{\partial t}(\dfrac{1}{f(x,t)}), \\ u(a(t),t) = \dfrac{a'(t)}{f(a(t),t)}, \qquad \text{or} \\ u(b(t),t) = \dfrac{b'(t)}{f(b(t),t)}. \end{cases} \tag{2.4}$$

- Second, determine $\phi(\xi,t)$ on Ω_0 by solving:

$$\begin{cases} \dfrac{\partial \phi}{\partial t}(\xi,t) = f(\phi(\xi,t),t)u(\phi(\xi,t),t), \\ \phi(\xi,0) = \xi. \end{cases} \tag{2.5}$$

We name this method *the deformation method(1D).*

2.2.2 Theoretical derivation

It is clear that the above 2 differential equations (2.4) and (2.5) are both solvable (see Remark 1 below about the domain issue of (2.4), Remark 2 about the boundary condition of (2.4)). So here we need to prove:

S1. $\phi_\xi(\xi,t)$ exists;

S2. (2.3) is satisfied, namely $\phi_\xi(\xi,t) = f(\phi(\xi,t),t)$;

S3. $\phi(a,t) = a(t)$ and $\phi(b,t) = b(t)$ for $\forall t$, namely $\phi(\xi,t)$ does map Ω_0 onto Ω_t;

in order to claim $\phi(\xi,t)$ is the desired diffeomorphism from Ω_0 to Ω_t.

S1. comes naturally true from the following theorem about smooth dependence on initial conditions of ordinary differential equations [18].

Theorem 1. *Let $\Omega \subset \mathbb{R}^{n+1}$ be an open set, and suppose that $f : \Omega \to \mathbb{R}^n$ is C^1. For $(s,p) \in \Omega$, the unique local solution $x(t,s,p)$ of the initial value problem*

$$\frac{d}{dt}x(t,s,p) = f(t,x(t,s,p)), \qquad x(s,s,p) = p$$

is C^1 in its open domain of definition

$$D = \{(t,s,p) \in \mathbb{R}^{n+2} : \alpha(s,p) < t < \beta(s,p), (s,p) \in \Omega\}.$$

The differential matrix $D_p x(t,s,p)$ satisfies the so-called linear variational equation

$$\begin{cases} \dfrac{d}{dt} D_p x(t,s,p) = D_x f(t, x(t,s,p)) D_p x(t,s,p), \\ D_p x(s,s,p) = I. \end{cases}$$

Proof of S1. Simply Let $n = 1, (s,p) = (0,\xi), x(t,s,p) = \phi(t,\xi), f = f(x,t)u(x,t)$ in Theorem 1, we can get $D_p x(t,s,p) = \phi_\xi(t,\xi) = \phi_\xi(\xi,t)$ exists. □

After (2.4), (2.5) and S1., we now have a differentiable map $\phi(\xi,t) : \Omega_0 \to \mathbb{R}$. Proofs of S2. and S3. continue as follows:

Proof of S2. Define $H(\xi,t) = \frac{\phi_\xi(\xi,t)}{f(\phi(\xi,t),t)}$ on $\Omega_0 \times [0,1]$ (also see Remark 1 about the domain issue of $f(\phi(\xi,t),t)$ here). Consequently $H(\xi,0) = 1$.

Our goal here is to prove $\frac{dH(\xi,t)}{dt} \equiv 0$, then $H(\xi,t) \equiv H(\xi,0) = 1$.

We have

$$\begin{aligned} \frac{dH(\xi,t)}{dt} &= [\frac{d}{dt}\phi_\xi(\xi,t)]\frac{1}{f(\phi(\xi,t),t)} + \phi_\xi(\xi,t)[\frac{d}{dt}\frac{1}{f(\phi(\xi,t),t)}] \\ &= \frac{\frac{d}{dt}\phi_\xi(\xi,t) f(\phi(\xi,t),t) - \phi_\xi(\xi,t)\frac{d}{dt} f(\phi(\xi,t),t)}{f(\phi(\xi,t),t)^2}. \end{aligned}$$

The numerator $\frac{d}{dt}\phi_\xi(\xi,t) f(\phi(\xi,t),t) - \phi_\xi(\xi,t)\frac{d}{dt} f(\phi(\xi,t),t)$

$$\begin{aligned} &= [\phi_t(\xi,t)]_\xi f(\phi(\xi,t),t) - \phi_\xi(\xi,t)[f_x(\phi(\xi,t),t)\phi_t(\xi,t) + f_t(\phi(\xi,t),t)] \\ &= [f(\phi(\xi,t),t)u(\phi(\xi,t),t)]_\xi f(\phi(\xi,t),t) - \phi_\xi(\xi,t)[f_x(\phi(\xi,t),t)\phi_t(\xi,t) \\ &+ f_t(\phi(\xi,t),t)] \\ &= [f_x(\phi(\xi,t),t)\phi_\xi(\xi,t)u(\phi(\xi,t),t) + f(\phi(\xi,t),t)u_x(\phi(\xi,t),t)\phi_\xi(\xi,t)] \\ &f(\phi(\xi,t),t) - \phi_\xi(\xi,t)[f_x(\phi(\xi,t),t)f(\phi(\xi,t),t)u(\phi(\xi,t),t) + f_t(\phi(\xi,t),t)] \\ &= f^2(\phi(\xi,t),t)u_x(\phi(\xi,t),t)\phi_\xi(\xi,t) - \phi_\xi(\xi,t)f_t(\phi(\xi,t),t) \\ &= [f^2(\phi(\xi,t),t)u_x(\phi(\xi,t),t) - f_t(\phi(\xi,t),t)]\phi_\xi(\xi,t) \\ &= [f^2(x,t)u_x(x,t) - f_t(x,t)]\phi_\xi(\xi,t) \\ &= 0. \end{aligned}$$

Hence, $H(\xi,t)=\frac{\phi_\xi(\xi,t)}{f(\phi(\xi,t),t)}\equiv 1 \Longrightarrow \phi_\xi(\xi,t)\equiv f(\phi(\xi,t),t)$. □

Proof of S3. The boundary condition of (2.4) gives us

$$a'(t)=f(a(t),t)u(a(t),t).$$

And let $\xi=a$ in (2.5), we have

$$\begin{cases}\dfrac{\partial\phi}{\partial t}(a,t)=f(\phi(a,t),t)u(\phi(a,t),t),\\ \phi(a,0)=a=a(0).\end{cases}$$

Since $f(y,t)\in C^1(y,t)$, and $u(y,t)$ is C^1 in y, *continuous* in t, by *Picard's existence theorem*, the ordinary differential equation (ODE) initial value problem

$$\begin{cases}\dfrac{\partial y}{\partial t}=f(y,t)u(y,t),\\ y(0)=a,\end{cases}$$

has a unique solution. Comparing the above two, we must have $\phi(a,t)=a(t)$. Then, let's recall S2. and use integration by substitution to get

$$\begin{aligned}\int_{\phi(a,t)}^{\phi(b,t)}\frac{1}{f(x,t)}dx&=\int_{\phi(a,t)}^{\phi(b,t)}\frac{1}{f(\phi(\xi,t),t)}d\phi(\xi,t)\\ &=\int_a^b\frac{\phi_\xi(\xi,t)}{f(\phi(\xi,t),t)}d\xi\\ &=\int_a^b 1d\xi=|\Omega_0|\\ &=\int_{a(t)}^{b(t)}\frac{1}{f(x,t)}dx. \qquad \text{by (2.2)}\end{aligned}$$

Noticing that $\phi(a,t)=a(t)$ and $f(x,t)>0$, we must have $\phi(b,t)=b(t)$. □

Now, we finish the theoretical derivation of *the deformation method(1D)*. Given domains Ω_t and a scalar function $f(x,t)$ on Ω_t with certain properties (2.1,2.2), a diffeomorphism $\phi(\xi,t):\Omega_0\to\Omega_t$ with $\phi_\xi(\xi,t)=f(\phi(\xi,t),t)$ can be constructed.

Several remarks are listed below, Remark 1,2 refine the derivation, Remark 3,4,5 discuss some special cases.

We studied the 1D case in order to get some simple but fundamental ideas about the general deformation approach, and in fact, most of the results here can be extended naturally to general case. We will discuss the general case later. In the next sections, a numerical algorithm and some numerical examples are presented to show the correctness and accuracy of *the deformation method(1D).*

Remark 1. *Solving differential equation (2.4) on* $\mathbb{R}$ *doesn't conflict with* $f(x,t)$ *is only defined on* Ω_t*. Because first, theoretically we can smoothly extend* $f(x,t)$ *from* Ω_t *to* $\mathbb{R}$ *such that (2.4) is solvable on* $\mathbb{R}$*. Second, later we know* $\phi(\xi,t)$ *maps* Ω_0 *onto* Ω_t *and only* $u(x,t) \in \Omega_t$ *is used in (2.5). So we only need to solve (2.4) on* Ω_t*.*

Remark 2. *The boundary condition in (2.4) guarantees the range of* $\phi(\xi,t)$*, and it can be substituted with* $u(b(t),t) = \frac{b'(t)}{f(b(t),t)}$*, namely the 2 boundary conditions are equivalent to each other. A brief proof is given here.*

Proof of Remark 2. Suppose $x = x(\xi,t)$ is any C^1 map from Ω_0 to Ω_t. By change of variables, we have

$$\begin{aligned}
0 = \frac{d}{dt}\int_{\Omega_t} \frac{1}{f(x,t)}dx &= \frac{d}{dt}\int_{\Omega_0} \frac{1}{f(x,t)}x_\xi d\xi = \int_{\Omega_0} \frac{d}{dt}(\frac{x_\xi}{f(x,t)})d\xi \\
&= \int_{\Omega_0} x_\xi \frac{d}{dt}(\frac{1}{f(x,t)})d\xi + \int_{\Omega_0} \frac{1}{f(x,t)}\frac{d}{dt}(x_\xi)d\xi \\
&= -\int_{\Omega_0} \frac{x_\xi(f_x x_t + f_t)}{f^2(x,t)}d\xi + \int_{\Omega_0} \frac{(x_t)_\xi}{f(x,t)}d\xi \\
&= -\int_{\Omega_0} \frac{x_\xi f_t}{f^2(x,t)}d\xi - \int_{\Omega_0} \frac{x_\xi f_x x_t}{f^2(x,t)}dx + \int_{\Omega_0} \frac{(x_t)_\xi}{f(x,t)}d\xi \\
&= \int_{\Omega_t} \frac{\partial}{\partial t}(\frac{1}{f(x,t)})dx + \int_{\Omega_0} (\frac{x_t}{f(x,t)})_\xi d\xi.
\end{aligned}$$

Which leads to $\int_{\Omega_0}(\frac{x_t}{f(x,t)})_\xi d\xi = -\int_{\Omega_t}\frac{\partial}{\partial t}(\frac{1}{f(x,t)})dx = \int_{\Omega_t} u_x dx$, and

$$\begin{aligned}
&\Rightarrow \frac{x_t}{f(x,t)}\Big|_a^b = \int_{a(t)}^{b(t)} u_x dx = u\Big|_{a(t)}^{b(t)},\\
&\Rightarrow \frac{b'(t)}{f(b(t),t)} - \frac{a'(t)}{f(a(t),t)} = u(b(t)) - u(a(t)),\\
&\Rightarrow u(b(t)) = \frac{b'(t)}{f(b(t),t)} \Longleftrightarrow u(a(t)) = \frac{a'(t)}{f(a(t),t)}.
\end{aligned}$$

□

Remark 3. *One special case in practice is letting $\Omega_t \equiv \Omega_0$, namely the fixed domain problem. In this special case, $f(x,t) > 0$ should be given on $\Omega_0 \times [0,1]$, such that*

$$f(x,0) = 1, \tag{2.1*}$$

$$\int_{\Omega_0}\frac{1}{f(x,t)}dx = |\Omega_0| = b - a. \tag{2.2*}$$

After solving differential equations (2.4) and (2.5), we can get $\phi(\xi,t) : \Omega_0 \to \Omega_0$ which satisfies (2.3).

Remark 4. *Another special case is $f(x,t) = \frac{b(t)-a(t)}{b-a}$, where now we can get $\phi(\xi,t) = \frac{b(t)-a(t)}{b-a}(\xi - a) + a(t)$ after solving differential equations (2.4) and (2.5). As expected, $\phi(\xi,t)$ is a linear transformation from Ω_0 to Ω_t. A brief proof is also given here.*

Proof of Remark 4. For short, let $g(t) = b(t) - a(t), c = b - a$, so $\frac{1}{f(x,t)} = \frac{c}{g(t)}$. (2.4) is now

$$\begin{cases} u_x = \dfrac{cg'(t)}{g^2(t)}, \\ u(a(t),t) = \dfrac{ca'(t)}{g(t)}. \end{cases}$$

Simply integrate it to get

$$u(x,t) = \frac{cg'(t)}{g^2(t)}x - \frac{cg'(t)a(t)}{g^2(t)} + \frac{ca'(t)}{g(t)}.$$

Plugging it into (2.5), we have

$$\phi_t = \frac{g'(t)}{g(t)}\phi - \frac{g'(t)a(t)}{g(t)} + a'(t),$$

which is

$$(\frac{\phi}{g(t)})' = (\frac{a(t)}{g(t)})'.$$

Therefore

$$\frac{\phi}{g(t)} = \frac{a(t)}{g(t)} + h(\xi) \Longrightarrow \phi(\xi, t) = a(t) + g(t)h(\xi).$$

Considering the boundary condition $\phi(\xi, 0) = \xi$, we can get $h(\xi) = \frac{\xi - a}{b - a}$. Finally,

$$\phi(\xi, t) = a(t) + \frac{b(t) - a(t)}{b - a}(\xi - a).$$

□

Remark 5. *A simple way of constructing $\phi(\xi, t)$ in 1D case is to directly solve the ordinary differential equation on Ω_0 for any fixed t:*

$$\begin{cases} \phi_\xi(\xi, t) = f(\phi(\xi, t), t), \\ \phi(a, t) = a(t). \end{cases} \tag{2.6}$$

It looks like in 1D case, the deformation method(1D) may not have advantages over directly solving ODE (2.6). However, in general 2D and 3D cases, where we can not directly solve the ODE, the deformation method still works in the construction of diffeomorphisms. Please check the following sections for details.

2.2.3 Numerical implementation and examples

2.2.3.1 Algorithm implementation

We implemented *the deformation method(1D)* by the following 4-step algorithm, which bases on a scheme of multiple time steps:

- **Step 1: Initialize.** Starts from $t = 0$, $\phi(\xi, 0) = \xi$, $f(x, 0) = 1$.
- **Step 2: ODE (2.4).** Compute $-\frac{\partial}{\partial t}(\frac{1}{f(x,t)})$, then **solve** (2.4) to get $u(x, t)$ on Ω_t.

- **Step 3: ODE (2.5).** Update $\phi(\xi, t)$ from Ω_t to Ω_{t+dt} by (2.5).
- **Step 4: Next time step.** Move to next time step $t = t + dt$, **back to Step 2** till $t = 1$.

2.2.3.2 Precision analysis

When using numerical methods to implement the above algorithm of constructing $\phi(\xi, t)$, the precision relies on the following parts:

1. Evaluating $-\frac{\partial}{\partial t}(\frac{1}{f(x,t)})$, with error $O(h_t)$ or higher order.
2. Solving ODE (2.4), with error $O(h_x)$ or higher order.
3. Solving ODE (2.5), with error $O(h_t)$ or higher order.
4. Potentially evaluating $f(\phi(\xi, t), t)$ by interpolation, with error $O(h_x)$ or higher order.

2.2.3.3 Numerical Example 1: moving domains with $f(x, t)$ normalized already

Let $t \in [0, 1], \Omega_t = [0, 1 - 0.9t]$, and $f(x, t) = 1 - 0.9t$. The analytic solution to this problem is

$$\phi(\xi, t) = (1 - 0.9t)\xi.$$

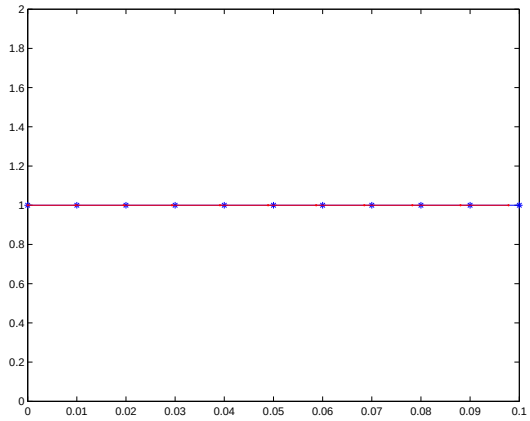
We use 11 equally spaced nodes on $[0, 1]$ and 11 equally distributed time steps from 0 to 1, i.e. $h_x = h_t = 0.1$. Apply second order numerical methods for each part, we get the numerical solutions as follows:

Nodes	*1*	*2*	*3*	*4*	*5*	*6*
analytic solution	0	0.055	0.110	0.165	0.2200	0.2750
numerical solution	0	0.05489	0.10979	0.16468	0.21958	0.27447
error	0	1.05e-4	2.1e-4	3.15e-4	4.2e-4	5.25e-4
Nodes	*7*	*8*	*9*	*10*	*11*	
analytic solution	0.3300	0.3850	0.4400	0.4950	0.5500	
numerical solution	0.32937	0.38426	0.43916	0.49405	0.54895	
error	6.3e-4	7.35e-4	8.39e-4	9.44e-4	10.49e-4	

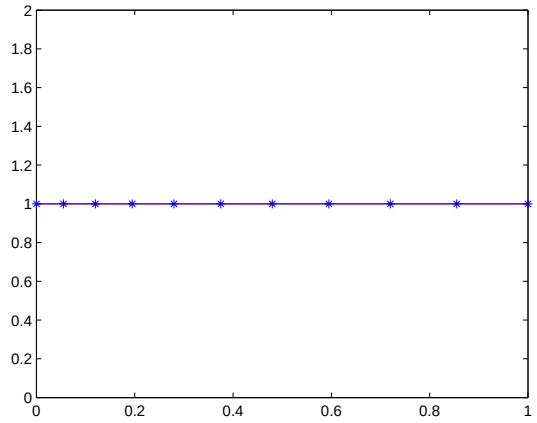
Table 2.1. Comparison of example 1: $\phi(\xi, 0.5)$.

Nodes	*1*	*2*	*3*	*4*	*5*	*6*
analytic solution	0	0.01	0.02	0.03	0.04	0.05
numerical solution	0	0.00978	0.01956	0.02934	0.03913	0.04891
error	0	2.18e-4	4.35e-4	6.53e-4	8.7e-4	10.88e-4
Nodes	*7*	*8*	*9*	*10*	*11*	
analytic solution	0.06	0.07	0.08	0.09	0.1	
numerical solution	0.05869	0.06847	0.07826	0.08804	0.09782	
error	13.06e-4	15.23e-4	17.41e-4	19.58e-4	21.76e-4	

Table 2.2. Comparison of example 1: $\phi(\xi, 1)$.



(a) example 1



(b) example 2

Figure 2.2. Illustration of examples 1 and 2: $\phi(\xi, 1)$. Red dots represent numerical solution, blue stars represent analytic solution.

2.2.3.4 Example 2: fixed domain

Let $t \in [0,1], \Omega_0 = \Omega_t = [0,1]$, $f(x,t) = 1 - t + t\sqrt{2x + \frac{1}{4}}$ and be normalized on Ω_t by $f(x,t) = f(x,t) \int_{\Omega_t} \frac{1}{f(x,t)} dx$. The analytic solution at $t = 1$ to the problem is

$$\phi(\xi, 1) = \frac{1}{2}(\xi^2 + \xi).$$

Similar to example 1, use 11 equally spaced nodes on $[0, 1]$ and 11 equally distributed time steps from 0 to 1, i.e. $h_x = h_t = 0.1$. Apply second order numerical methods for each part, we get the numerical solutions as follows:

Nodes	*1*	*2*	*3*	*4*	*5*	*6*
analytic solution	0	0.0550	0.1200	0.1950	0.2800	0.3750
numerical solution	0	0.0567	0.1198	0.1935	0.2782	0.3731
error	0	0.0017	-0.0002	-0.0015	-0.0018	-0.0019
Nodes	*7*	*8*	*9*	*10*	*11*	
analytic solution	0.4800	0.5950	0.7200	0.8550	1.0000	
numerical solution	0.4782	0.5935	0.7190	0.8545	1.0000	
error	-0.0018	-0.0015	-0.0010	-0.0005	0	

Table 2.3. Comparison of example 2: $\phi(\xi, 1)$.

2.2.3.5 Example 3: moving domains with $f(x,t) = \frac{b(t)-a(t)}{b-a}$

Let $t \in [0,1], \Omega_t = [1 + t\sin t, 2 + \frac{1}{2}t^2]$, $f(x,t) = \frac{b(t)-a(t)}{b-a} = 1 + \frac{1}{2}t^2 - t\sin t$. As mentioned in Remark 4, the analytic solution is

$$\phi(\xi, t) = (1 + \frac{1}{2}t^2 - t\sin t)(\xi - 1) + 1 + t\sin t.$$

By using 11 equally spaced nodes on $[1, 2]$ and 11 equally distributed time steps from 0 to 1, we get the numerical solutions as follows:

Nodes	*1*	*2*	*3*	*4*	*5*	*6*
analytic solution	1.8415	1.9073	1.9732	2.0390	2.1049	2.1707
numerical solution	1.8415	1.9073	1.9731	2.0389	2.1047	2.1705
error	0	3.83e-5	7.66e-5	11.49e-5	15.32e-5	19.15e-5
Nodes	*7*	*8*	*9*	*10*	*11*	
analytic solution	2.2366	2.3024	2.3683	2.4341	2.5000	
numerical solution	2.2364	2.3022	2.3680	2.4338	2.5000	
error	22.98e-5	26.81e-5	30.63e-5	34.46e-5	0	

Table 2.4. Comparison of example 3: $\phi(\xi, 1)$.

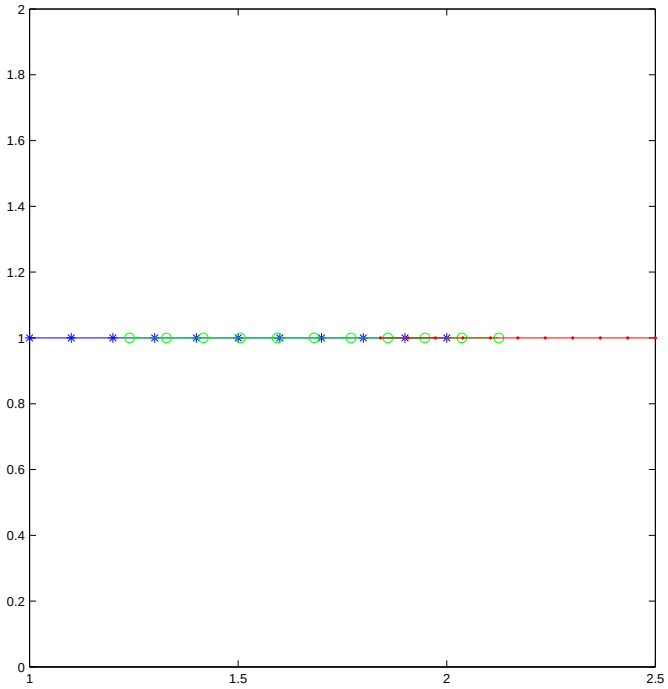


Figure 2.3. Illustration of example 3: blue stars for $\phi(\xi, 0)$, green circles for $\phi(\xi, 0.5)$, red dots for $\phi(\xi, 1)$.

2.2.3.6 Example 4: moving domains

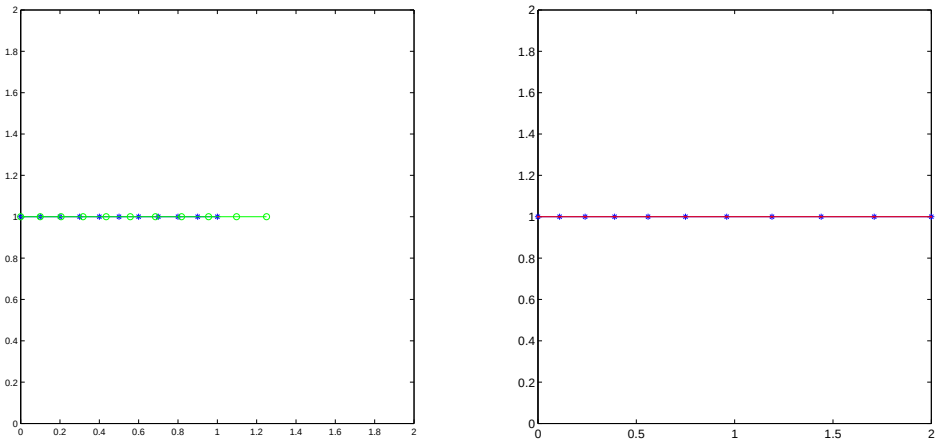
Let $t \in [0,1], \Omega_t = [0, 1+t^2]$, $f(x,t) = 1 - t + 2t\sqrt{x + \frac{1}{4}}$ and be normalized on Ω_t by $f(x,t) = f(x,t) \int_{\Omega_t} \frac{1}{f(x,t)} dx$. The analytic solution at $t = 1$ to the problem is

$$\phi(\xi, 1) = \xi^2 + \xi.$$

Similarly, the numerical solutions:

Nodes	*1*	*2*	*3*	*4*	*5*	*6*
analytic solution	0	0.1100	0.2400	0.3900	0.5600	0.7500
numerical solution	0	0.1097	0.2395	0.3893	0.5591	0.7489
error	0	3.14e-4	5.30e-4	7.11e-4	8.95e-4	11.12e-4
Nodes	*7*	*8*	*9*	*10*	*11*	
analytic solution	0.9600	1.1900	1.4400	1.7100	2.0000	
numerical solution	0.9586	1.1883	1.4379	1.7073	2.0000	
error	13.82e-4	17.21e-4	21.41e-4	26.53e-4	0	

Table 2.5. Comparison of example 4: $\phi(\xi, 1)$.



(a) Green dots for $\phi(\xi, 0.5)$, blue stars for $\phi(\xi, 0)$.

(b) $\phi(\xi, 1)$, red dots for numerical solution, blue stars for analytic solution.

Figure 2.4. Illustration of example 4.

2.2.3.7 Conclusion

We can see clearly that all examples get satisfied results. And further studies by theoretical analysis and numerical examples indicate that the precision of evaluating $-\frac{\partial}{\partial t}(\frac{1}{f(x,t)})$ effects the numerical outcomes most. As mentioned in Remark 5, *the deformation method(1D)* is not our final object. We shall end the discussion in 1D case here, and move to general cases with the same basic ideas.

2.3 Two- and Three-dimensional case

2.3.1 Problem formulation

Now Let $\Omega_t \subset \mathbb{R}^n (n = 2, 3)$ be moving domains, $\boldsymbol{v}(\boldsymbol{x}, t)$ be the velocity field on the boundary $\partial\Omega_t$. Given a scalar function $f(\boldsymbol{x}, t) \in C^1(\boldsymbol{x}, t) > 0$ on domain of $(\boldsymbol{x}, t) : \Omega_t \times [0, 1]$ such that

$$f(\boldsymbol{x}, 0) \quad = 1, \tag{2.7}$$

$$\int_{\Omega_t} \frac{1}{f(\boldsymbol{x},t)} d\boldsymbol{x} \quad = |\Omega_0|. \tag{2.8}$$

A diffeomorphism

$$\boldsymbol{\phi}(\boldsymbol{\xi}, t) : \Omega_0 \to \Omega_t$$

such that $\forall t \geq 0$

$$J(\boldsymbol{\phi}(\boldsymbol{\xi}, t)) = f(\boldsymbol{\phi}(\boldsymbol{\xi}, t), t) \tag{2.9}$$

can be constructed by solving the following 2 differential equations (2.10) and (2.11). (Note: Here $J(\boldsymbol{\phi}(\boldsymbol{\xi}, t)) = \det\nabla\boldsymbol{\phi}(\boldsymbol{\xi}, t)$ is the Jacobian determinant of $\boldsymbol{\phi}(\boldsymbol{\xi}, t)$.)

- First, determine $\boldsymbol{u}(\boldsymbol{x},t)$ on $\mathbb{R}^n$ by solving the *div-curl-system*:

$$\begin{cases} \text{div } \boldsymbol{u}(\boldsymbol{x},t) = -\dfrac{\partial}{\partial t}(\dfrac{1}{f(\boldsymbol{x},t)}), \\ \text{curl } \boldsymbol{u}(\boldsymbol{x},t) = \boldsymbol{0}, \\ \boldsymbol{u}(\boldsymbol{x},t) = \dfrac{\boldsymbol{v}(\boldsymbol{x},t)}{f(\boldsymbol{x},t)} \qquad \text{on } \partial\Omega_t. \end{cases} \tag{2.10}$$

- Second, determine $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ on Ω_0 by solving:

$$\begin{cases} \dfrac{\partial \boldsymbol{\phi}}{\partial t}(\boldsymbol{\xi},t) = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t), \\ \boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{\xi}. \end{cases} \tag{2.11}$$

We name this method *the deformation method*, and (2.10) the *div-curl-system*.

Remark 6. *Comparing with the deformation method(1D), here in (2.10), we add one more curl equation. The purpose of adding this curl equation is just to make the div-curl-system have one unique solution* $\boldsymbol{u}$ *[19]. It will not effect the property (2.9).*

2.3.2 Theoretical derivation

Similarly, in order to claim the constructed $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ is the desired diffeomorphism, we shall prove the following statements:

S4. $\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)$ exists and (2.9) is satisfied;

S5. $\frac{\partial \boldsymbol{\phi}(\boldsymbol{\xi},t)}{\partial t} = \boldsymbol{v}(\boldsymbol{\xi},t)$ on $\partial\Omega_0$, namely $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ does map Ω_0 onto Ω_t.

Before proving S4., we need to recall a theorem here[20].

Theorem 2 (Liouville's Theorem or Abel-Jacobi-Liouville Identity)**.** *If* $A \in C(\mathbb{R}, \mathbb{R}^{n\times n})$ *is a* $n \times n$ *matrix, and* $X(t)$ *is a matrix solution of* $X'(t) = A(t)X(t)$*, then*

$$\mathrm{detX(t) = detX(t_0)e^{\int_{t_0}^{t} TrA(s)ds}},$$

furthermore,

$$\frac{d}{dt}\mathrm{detX(t) = TrA(t)detX(t)}.$$

Proof of S4. The existence of $\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)$ follows the same smooth dependence on initial conditions of ODE Theorem 1.

To prove (2.9), we define $H(\boldsymbol{\xi},t)=\frac{J(\boldsymbol{\phi}(\boldsymbol{\xi},t))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}$ on $\Omega_0\times[0,1]$, consequently $H(\boldsymbol{\xi},0)=1$. Our goal here is still to prove $\frac{dH(\boldsymbol{\xi},t)}{dt}\equiv 0$, then $H(\boldsymbol{\xi},t)\equiv H(\boldsymbol{\xi},0)=1$.

Start with

$$\begin{aligned}\frac{dH(\boldsymbol{\xi},t)}{dt}&=\frac{d}{dt}\frac{J(\boldsymbol{\phi}(\boldsymbol{\xi},t))}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}=\frac{d}{dt}\frac{\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}\\&=[\frac{d}{dt}\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)]\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}+\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)[\frac{d}{dt}\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}].\end{aligned}\tag{$*$}$$

To be clear here, let $n=3$ and

$$\boldsymbol{\phi}=\begin{pmatrix}\phi_1\\ \phi_2\\ \phi_3\end{pmatrix}_{3\times1},\boldsymbol{u}=\begin{pmatrix}u_1\\ u_2\\ u_3\end{pmatrix}_{3\times1},\nabla_{\boldsymbol{\phi}}f=\begin{pmatrix}f_{\phi_1},f_{\phi_2},f_{\phi_3}\end{pmatrix}_{1\times3}.$$

Consider $\frac{d}{dt}\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)$,

$$\begin{aligned}&\frac{d}{dt}\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)=\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}_t(\boldsymbol{\xi},t)=\nabla_{\boldsymbol{\xi}}[f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\\&=\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)+f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\nabla_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)\\&=[\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)+f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\nabla_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\cdot\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t).\end{aligned}$$

By Theorem 2, we have

$$\begin{aligned}&\frac{d}{dt}\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)\\&=\mathrm{Tr}[\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)+f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\nabla_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)\\&=[\nabla_{\boldsymbol{\phi}}f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)+f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t).\end{aligned}$$

Plug it back into $(*)$,

$$\begin{aligned}
\frac{dH(\boldsymbol{\xi},t)}{dt} &= \{[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\
&\quad + [\frac{d}{dt}\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}]\}\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t) \\
&= \{[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\frac{1}{f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\
&\quad - \frac{\nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{\phi}_t(\boldsymbol{\xi},t) + f_t(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}{f^2(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)}\}\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t) \\
&= \{[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) \\
&\quad - \nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{\phi}_t(\boldsymbol{\xi},t) + f_t(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\}\frac{\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)}{f^2(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\
&= \{[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot \boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) \\
&\quad - \nabla_{\boldsymbol{\phi}} f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\cdot f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) + f_t(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\}\frac{\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)}{f^2(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\
&= [f^2(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\mathrm{div}_{\boldsymbol{\phi}}\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t) - f_t(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)]\frac{\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)}{f^2(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)} \\
&= [f^2(\boldsymbol{x},t)\mathrm{div}_{\boldsymbol{x}}\boldsymbol{u}(\boldsymbol{x},t) - f_t(\boldsymbol{x},t)]\frac{\det\nabla_{\boldsymbol{\xi}}\boldsymbol{\phi}(\boldsymbol{\xi},t)}{f^2(\boldsymbol{x},t)} \\
&= 0.
\end{aligned}$$

Hence, $H(\boldsymbol{\xi},t)\equiv 1 \Rightarrow J(\boldsymbol{\phi}(\boldsymbol{\xi},t)) = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)$ for $\forall t$. □

Proof of S5. The proof is similar to the Proof of S3. Let $\forall \boldsymbol{\xi_0}\in\partial\Omega_0$ in (2.11), we get

$$\begin{cases} \dfrac{\partial\boldsymbol{\phi}}{\partial t}(\boldsymbol{\xi_0},t) = f(\boldsymbol{\phi}(\boldsymbol{\xi_0},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi_0},t),t), \\ \boldsymbol{\phi}(\boldsymbol{\xi_0},0) = \boldsymbol{\xi_0}. \end{cases}$$

From the boundary condition of (2.10), we have

$$\begin{cases} \dfrac{d\boldsymbol{x}}{dt} = \boldsymbol{v}(\boldsymbol{x},t) = f(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t), \\ \boldsymbol{x}(\boldsymbol{\xi_0},0) = \boldsymbol{\xi_0}. \end{cases}$$

By uniqueness of ODE initial value problem, we claim that $\boldsymbol{\phi}(\boldsymbol{\xi_0},t) = \boldsymbol{x}(\boldsymbol{\xi_0},t)$ for $\forall\boldsymbol{\xi_0}\in\partial\Omega_0$. Namely, $\boldsymbol{\phi}(\boldsymbol{\xi},t) = \boldsymbol{x}(\boldsymbol{\xi},t)$ on $\partial\Omega_0$. Moreover, $\frac{\partial\boldsymbol{\phi}(\boldsymbol{\xi},t)}{\partial t} = \boldsymbol{v}(\boldsymbol{\xi},t)$ on $\partial\Omega_0$. □

Remark 7. *For the same reason in Remark 1, solving differential equation (2.10) on* $\mathbb{R}^n$ *doesn't conflict with the condition that* $f(\boldsymbol{x},t)$ *is only defined on* Ω_t.

Remark 8. *One special case in practice is* $\Omega_t \equiv \Omega_0$, *namely the fixed domain problem. In this special case,* $f(\boldsymbol{x},t) > 0$ *should be given on* $\Omega_0 \times [0,1]$, *such that*

$$f(\boldsymbol{x},0) = 1, \tag{2.7*}$$

$$\int_{\Omega_0} \frac{1}{f(\boldsymbol{x},t)} d\boldsymbol{x} = |\Omega_0|. \tag{2.8*}$$

After solving differential equations (2.10) and (2.11), we can get $\boldsymbol{\phi}(\boldsymbol{\xi},t) : \Omega_0 \to \Omega_0$ *which satisfies (2.9).*

Remark 9. *Another special case in practice is* $f(\boldsymbol{x},t) = \frac{|\Omega_t|}{|\Omega_0|}$, *we can get a uniform map* $\boldsymbol{\phi}(\boldsymbol{\xi},t) : \Omega_0 \to \Omega_t$.

Before ending the theoretical discussion, we want to make one more Remark.

Remark 10. *Recall the deformation method, a diffeomorphism is constructed from the identity map, i.e.* $\boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{id}(\boldsymbol{\xi})$. *It works well both in theory and in practice. However, we could ask the question naturally: can we start the construction from any given map* $\boldsymbol{\phi}_0(\boldsymbol{\xi},0)$? *The answer is of course yes, and the explanation is simple. Just consider any* $t_0 > 0$, *then what happens in* $[t_0,1]$ *is exactly what we discuss here.*
So we can re-address the deformation method one more time as:
Keep the same assumptions about Ω_t *and* $\boldsymbol{v}(\boldsymbol{x},t)$. *Given a map* $\boldsymbol{\phi}_0(\boldsymbol{\xi}) : \Omega_0 \to \Omega_0$ *and a scalar function* $f(\boldsymbol{x},t) \in C^1(\boldsymbol{x},t) > 0$ *on domain of* $(\boldsymbol{x},t) : \Omega_t \times [0,1]$ *such that*

$$f(\boldsymbol{\phi}_0,0) \quad = J(\boldsymbol{\phi}_0), \tag{2.12}$$

$$\textstyle\int_{\Omega_t} \frac{1}{f(\boldsymbol{x},t)} d\boldsymbol{x} \quad = |\Omega_0|. \tag{2.13}$$

We construct a diffeomorphism

$$\boldsymbol{\phi}(\boldsymbol{\xi},t) : \Omega_0 \to \Omega_t$$

such that $J(\boldsymbol{\phi}(\boldsymbol{\xi},t)) = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)$*, by solving the following 2 differential equations (2.14) and (2.15).*

- *First, determine* $\boldsymbol{u}(\boldsymbol{x},t)$ *on* $\mathbb{R}^n$ *by solving the div-curl-system:*

$$\begin{cases} div\ \boldsymbol{u}(\boldsymbol{x},t) = -\dfrac{\partial}{\partial t}(\dfrac{1}{f(\boldsymbol{x},t)}), \\ curl\ \boldsymbol{u}(\boldsymbol{x},t) = \mathbf{0}, \\ \boldsymbol{u}(\boldsymbol{x},t) = \dfrac{\boldsymbol{v}(\boldsymbol{x},t)}{f(\boldsymbol{x},t)} \qquad on\ \partial\Omega_t. \end{cases} \tag{2.14}$$

- *Second, determine* $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ *on* Ω_0 *by solving:*

$$\begin{cases} \dfrac{\partial \boldsymbol{\phi}}{\partial t}(\boldsymbol{\xi},t) = f(\boldsymbol{\phi}(\boldsymbol{\xi},t),t)\boldsymbol{u}(\boldsymbol{\phi}(\boldsymbol{\xi},t),t), \\ \boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{\phi}_0(\boldsymbol{\xi}). \end{cases} \tag{2.15}$$

A numerical example in the next section will use this Remark to show an interesting problem.

Now, we finish the theoretical derivation of the general *deformation method*. It is a natural extension of *the deformation method(1D)*, a new *div-curl-system* (2.10) is introduced. Let's move on to the numerical implementation part, where the LSFEM method [19] is used to solve the *div-curl-system* (2.10).

2.3.3 Numerical implementation and examples

2.3.3.1 Algorithm implementation

We can still implement *the deformation method* by the same 4-step algorithm:

- **Step 1: Initialize.** Starts from $t = 0$, $\boldsymbol{\phi}(\boldsymbol{\xi},0) = \boldsymbol{\xi}$, $f(\boldsymbol{x},0) = 1$.
- **Step 2: Div-curl system (2.10).** Compute $-\frac{\partial}{\partial t}(\frac{1}{f(\boldsymbol{x},t)})$, then use LSFEM to **solve** (2.10), get $\boldsymbol{u}(\boldsymbol{x},t)$ on Ω_t.
- **Step 3: ODE (2.11).** Update $\boldsymbol{\phi}(\boldsymbol{\xi},t)$ from Ω_t to Ω_{t+dt} by (2.11).

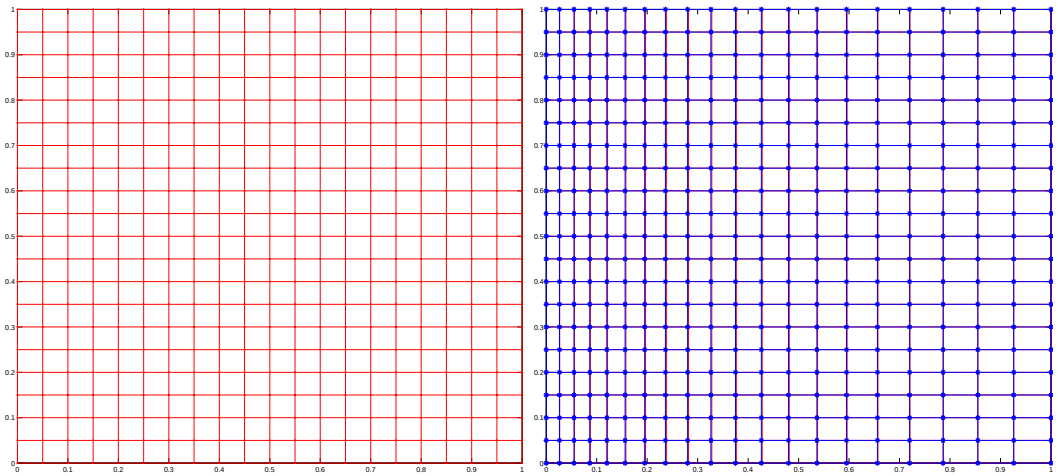
- **Step 4: Next time step.** Move to next time step $t = t + dt$, **back to Step 2** till $t = 1$.

2.3.3.2 Numerical Example 1: 2D fixed domain

Our first example considers a fixed domain $\Omega_t \equiv [0,1] \times [0,1]$. Let $f(x,y,t) = 1 - t^2 + t^2\sqrt{2x + \frac{1}{4}}$, and be normalized on Ω_t by $f(x,y,t) = f(x,y,t)\int_{\Omega_t} \frac{1}{f(x,y,t)} dxdy$. The analytic solution at $t = 1$ to the problem is

$$\boldsymbol{\phi}(\xi,\eta,1) = (\frac{\xi^2 + \xi}{2}, \eta).$$

The numerical solutions of $\boldsymbol{\phi}(\xi,\eta,0)$ and $\boldsymbol{\phi}(\xi,\eta,1)$ are showed in Figure 2.5. At $t = 1$, $max(\| \boldsymbol{\phi}_{analytic}(\xi,\eta,1) - \boldsymbol{\phi}_{numerical}(\xi,\eta,1) \|) = 0.0018$.



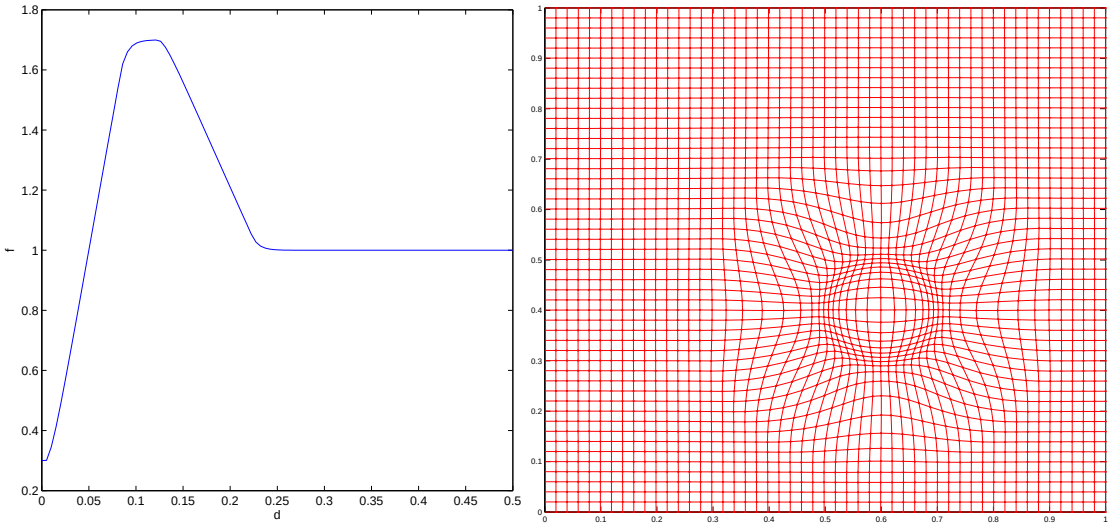
(a) $\boldsymbol{\phi}(\xi,\eta,0)$ (b) $\boldsymbol{\phi}(\xi,\eta,1)$

Figure 2.5. Illustration of example 1, red dots represent numerical solution, blue stars represent analytic solution.

2.3.3.3 Example 2: 2D fixed domain

For a general $f(\boldsymbol{x},t)$, we don't have a analytic solution $\boldsymbol{\phi}_{analytic}$. But we can examine the difference between $J(\boldsymbol{\phi})$ and $f(\boldsymbol{\phi})$, which is expected to be 0.

Let $d = \mid \sqrt{(x-0.4)^2+(y-0.6)^2} - 0.1 \mid$, i.e. the distance to the circle $(x-0.4)^2 + (y-0.6)^2 = 0.1^2$. Define $f(\boldsymbol{x})$ based on the function $f(d)$ showed in Figure 2.6(a). $f(\boldsymbol{x}, t) = 1 - \sin t + \sin t f(\boldsymbol{x})$ and normalized on a fixed domain $\Omega_t \equiv [0,1] \times [0,1]$. Figure 2.5(b) shows the constructed diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$. Table 2.6 shows the L^2-norm between $J(\boldsymbol{\phi})$ and $f(\boldsymbol{\phi})$ at each time step t from 0 to 1, with $dt = 0.1$ and $dt = 0.02$. We can clearly check the accuracy.



(a) $f(d)$ (b) $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$

Figure 2.6. Example 2, (a) shows a function $f(d)$ used to define $f(\boldsymbol{x}, t)$. (b) shows the constructed diffeomorphism $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$.

Time t	$\| J(\boldsymbol{\phi}) - f(\boldsymbol{\phi}) \|_2$, $dt = 0.1$	$\| J(\boldsymbol{\phi}) - f(\boldsymbol{\phi}) \|_2$ $dt = 0.02$
0.1	1.6827e-005	1.0807e-005
0.2	1.0734e-004	4.2828e-005
0.3	3.2271e-004	9.5412e-005
0.4	6.5313e-004	1.7480e-004
0.5	0.0011	2.7650e-004
0.6	0.0017	3.9221e-004
0.7	0.0024	4.6594e-004
0.8	0.0034	5.9361e-004
0.9	0.0051	8.8787e-004
1.0	0.0066	0.0011

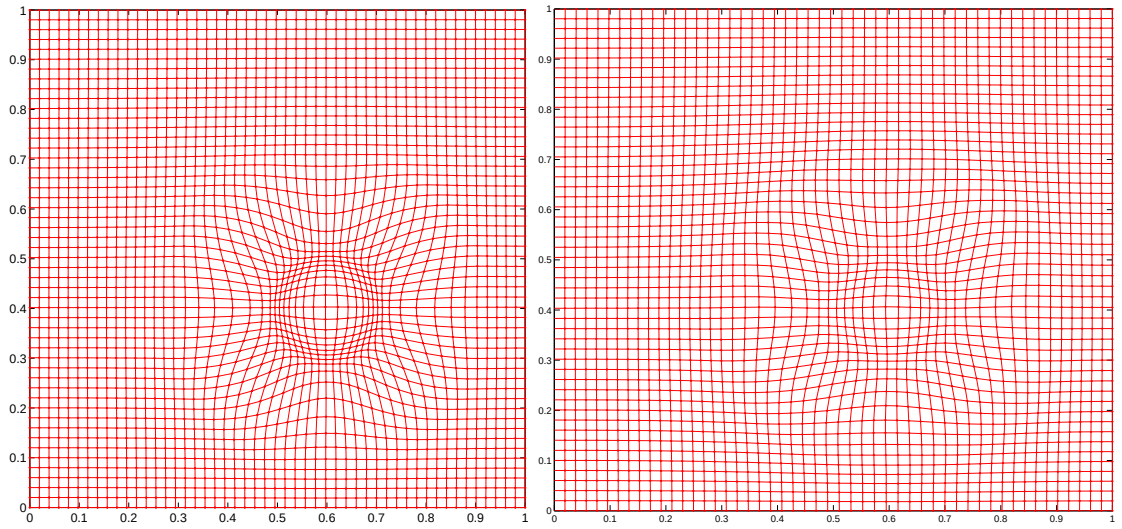
Table 2.6. L^2-norm of the difference $J(\boldsymbol{\phi})$ and $f(\boldsymbol{\phi})$ at each time step t

Note: Professor S.Turek's group studied the precision problem with *the Poisson based deformation method*[21], they also applied *the deformation method* to simulation of multiple falling balls in water[22].

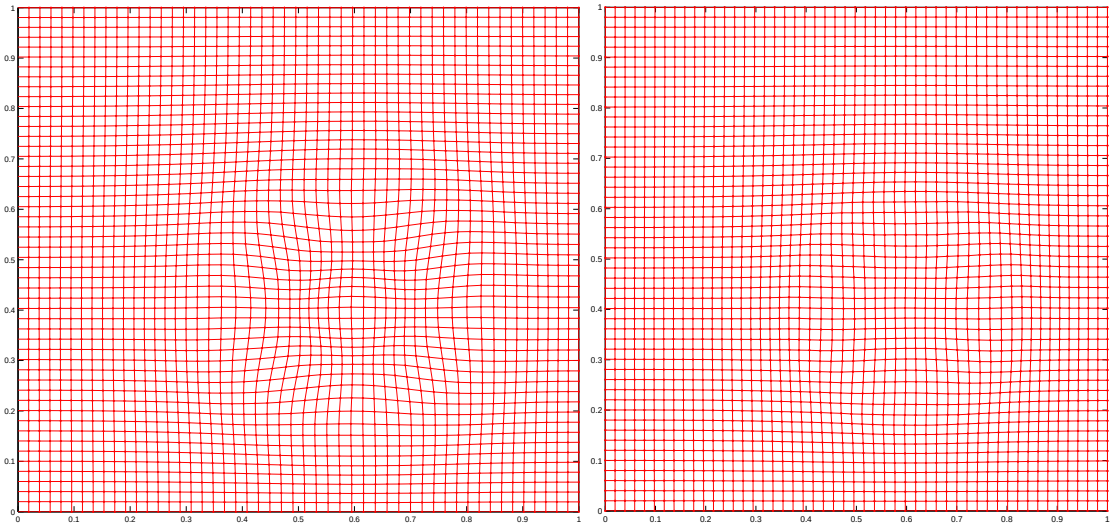
2.3.3.4 Example 3: Back to identity map

The third example deals with an interesting problem mentioned in Remark 10. We want to construct diffeomorphisms from some $\boldsymbol{\phi}_0$ back to the identity map $\boldsymbol{id}(\boldsymbol{\xi})$. Given a known diffeomorphism $\boldsymbol{\phi}_0$, numerically we can find $f(\boldsymbol{x})$ such that $f(\boldsymbol{\phi}_0) = J(\boldsymbol{\phi}_0)$. Define $f(\boldsymbol{x}, t) = t + (1-t)f(\boldsymbol{x})$ and normalized on Ω_t, then in theory, apply *the deformation method*, we shall expect $\boldsymbol{\phi}(\boldsymbol{\xi}, 1) = \boldsymbol{\xi}$.

Take the constructed $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$ in the previous example 2 as $\boldsymbol{\phi}_0$ here. The results are shown in Figure 2.7, $max \parallel \boldsymbol{\phi}(\boldsymbol{\xi}, 1) - \boldsymbol{\xi} \parallel = 0.0150$.



(a) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0) = \boldsymbol{\phi}_0$ (b) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0.4)$



(c) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0.6)$ (d) $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$

Figure 2.7. Example 3, from a known map back to identity map.

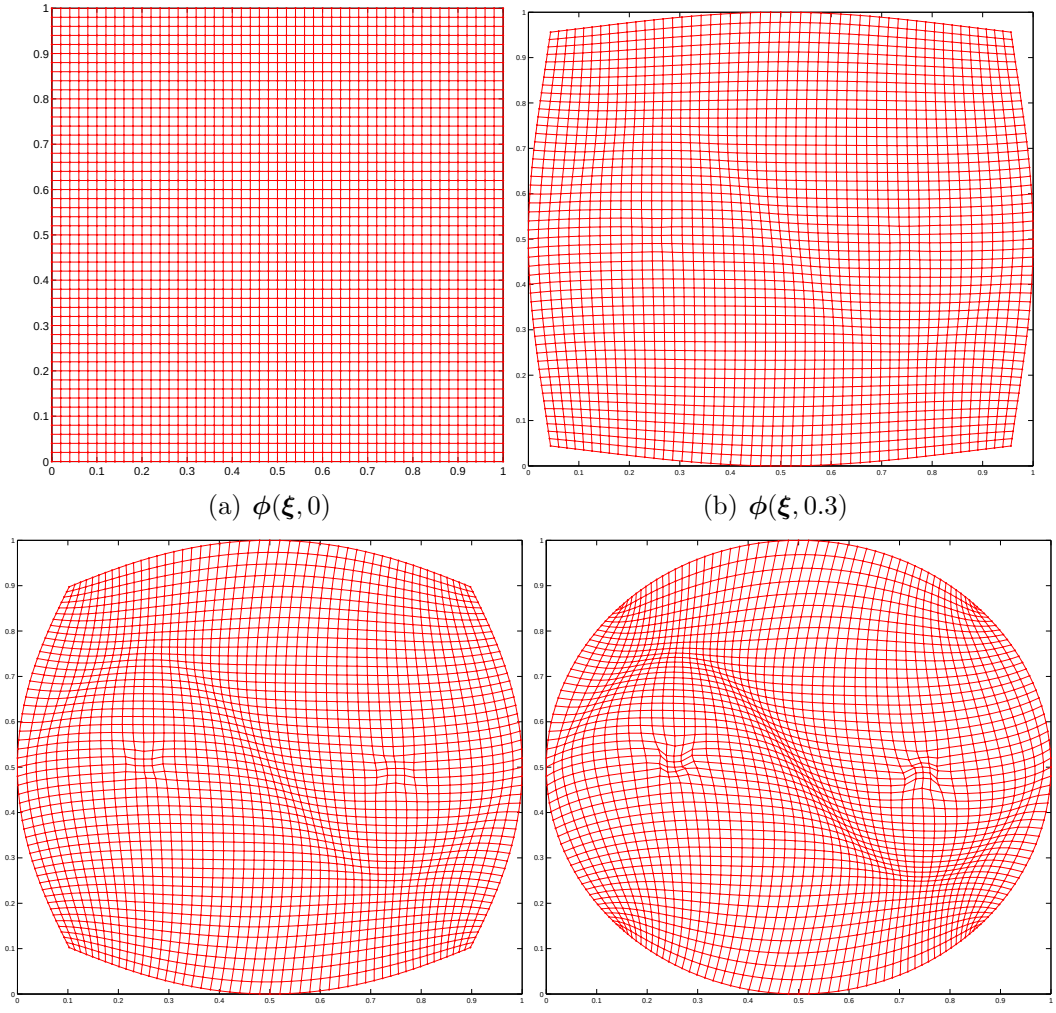
2.3.3.5 Example 4: 2D moving domain

In this example, Ω_0 is set to be a unit square, Ω_1 be a unit circle, and Ω_t be a series of intermediate domains determined by the boundary nodes. Let $d_1 =$

$\sqrt{(x-\frac{1}{2})^2+(y-\frac{1}{4})^2}, d_2 = \sqrt{(x-\frac{1}{2})^2+(y-\frac{3}{4})^2}, d_3 = \mid x-\frac{1}{2}-\frac{1}{4}\sin 2\pi y \mid$. Define $f(x,y)$ as:

$$f(x,y) = \begin{cases} 1 - C_1 e^{-C_2 min(d_1,d_2)} & d_1 \le p \ or \ d_2 \le p, \\ 1 - C_1 e^{-C_2 d_3} & else, \end{cases}$$

C_1, C_2, p are parameters. $f(x,y,t) = 1 - t + tf(x,y)$ and normalized on Ω_t.



(a) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0)$ (b) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0.3)$

(c) $\boldsymbol{\phi}(\boldsymbol{\xi}, 0.7)$ (d) $\boldsymbol{\phi}(\boldsymbol{\xi}, 1)$

Figure 2.8. Illustration of example 4, here $C_1 = 0.8, C_2 = 5, p = 0.03$.

CHAPTER 3

NEW VARIATIONAL METHOD WITH PRESCRIBED JACOBIAN DETERMINANT AND CURL VECTOR

3.1 Introduction

In *the deformation method*, we first determine a velocity field $\boldsymbol{u}$, then use this velocity field $\boldsymbol{u}$ to construct $\boldsymbol{\phi}$. The ultimate criterion is actually the Jacobian determinant of $\boldsymbol{\phi}$. So we could ask, since what we want is the Jacobian determinant of a diffeomorphism $\boldsymbol{\phi}$, can we just control it directly, without the help of velocity field $\boldsymbol{u}$? We explored this idea of controlling the Jacobian determinant directly, and developed a new approach based on an optimization scheme. The core procedure is to define a cost functional, derive the variational gradient, apply gradient descend method to minimize the cost functional, and construct $\boldsymbol{\phi}$ iteratively. Because the highlight of this approach is the derivation of the variational gradient ,we named it *new variational method with prescribed Jacobian determinant*.

Later, inspired from the *div-curl system*, we learnt it is for sure that with the same Jacobian determinant, different curl vectors can lead different diffeomorphisms [23]. This fact reminds us that the curl vector of $\boldsymbol{\phi}$ is also needed in the cost functional. Therefore, we extended the new approach to *new variational method with prescribed Jacobian determinant and curl vector*.

In this chapter, different versions of the *new variational method* are proposed. Theoretical derivations of the variational gradient are discussed in detail. Also, numerical algorithms and examples are presented to show the satisfactory results in the numerical construction of diffeomorphisms.

More over, in the next chapter, we will use this method to study a potential uniqueness problem in differential geometry.

First, let's look at the *new variational method with prescribed Jacobian determinant*.

3.2 New Variational Method: Version 1

3.2.1 Problem formulation

Given a domain $\Omega \subset \mathbb{R}^n$, $n = 1, 2, 3$, and a scalar function $f_0(\boldsymbol{x}) > 0$ defined on Ω with

$$\int_\Omega f_0(\boldsymbol{x}) d\boldsymbol{x} = \mid \Omega \mid . \tag{3.1}$$

Define a cost functional *ssd* to be:

$$ssd = \frac{1}{2} \int_\Omega (J(\boldsymbol{\phi}(\boldsymbol{x})) - f_0(\boldsymbol{x}))^2 d\boldsymbol{x}. \tag{3.2}$$

A transformation $\boldsymbol{\phi}(\boldsymbol{x}) : \Omega \to \Omega$ such that $J(\boldsymbol{\phi}(\boldsymbol{x})) = f_0(\boldsymbol{x})$ can be numerically constructed by minimizing *ssd* with respect to a control function $f(\boldsymbol{x})$ under the following 2 constraints (3.3) and (3.4).

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}_{old}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}) dt, \tag{3.3}$$

where dt is an artificial time step, and $\boldsymbol{u}(\boldsymbol{x})$ is a velocity vector satisfying:

$$\begin{cases} \text{div } \boldsymbol{u} &= f & \text{in } \Omega, \\ \text{curl } \boldsymbol{u} &= 0 & \text{in } \Omega, \\ \boldsymbol{u} &= \mathbf{0} & \text{on } \partial\Omega. \end{cases} \tag{3.4}$$

We name this method *new variational method with prescribed Jacobian determinant*, and call $f_0(\boldsymbol{x})$ the prescribed Jacobian determinant monitor function.

3.2.2 Theoretical derivation

In order to apply the gradient descend method to minimize *ssd*, we shall derive the variational derivative $\frac{\partial ssd}{\partial f}$ by variational calculus. Without loss of generality, we

show the theoretical derivation in two-dimensional case.

Now $\boldsymbol{\phi} = (\phi_1, \phi_2), \boldsymbol{u} = (u_1, u_2)$. Notice that (3.4) leads to the poisson's equation:

$$\begin{cases} \Delta u_1 &= f_x, \\ \Delta u_2 &= f_y. \end{cases} \tag{3.5}$$

Let δf be any given variation of f which vanishes on the boundary, we have:

$$\begin{aligned} \delta ssd &= \int (J(\boldsymbol{\phi}(\boldsymbol{x})) - f_0(\boldsymbol{x}))\delta J(\boldsymbol{\phi}(\boldsymbol{x}))d\boldsymbol{x} \\ &= \int (J(\boldsymbol{\phi}) - f_0)\delta J(\boldsymbol{\phi})d\boldsymbol{x} \\ &= \int (J(\boldsymbol{\phi}) - f_0)\delta(\phi_{1x}\phi_{2y} - \phi_{2x}\phi_{1y})d\boldsymbol{x} \\ &= \int (J(\boldsymbol{\phi}) - f_0)(\delta\phi_{1x}\phi_{2y} + \phi_{1x}\delta\phi_{2y} - \delta\phi_{2x}\phi_{1y} - \phi_{2x}\delta\phi_{1y})d\boldsymbol{x} \\ &= \int (J(\boldsymbol{\phi}) - f_0)(\delta u_{1x}dt\phi_{2y} + \phi_{1x}\delta u_{2y}dt - \delta u_{2x}dt\phi_{1y} - \phi_{2x}\delta u_{1y}dt)d\boldsymbol{x}. \end{aligned}$$

The last line above follows $\delta\boldsymbol{\phi} = \delta\boldsymbol{u}dt$ from (3.3).

Let $P = (J(\boldsymbol{\phi}) - f_0)dt$, we can continue as

$$\begin{aligned} \delta ssd &= \int P(\delta u_{1x}\phi_{2y} + \phi_{1x}\delta u_{2y} - \delta u_{2x}\phi_{1y} - \phi_{2x}\delta u_{1y})d\boldsymbol{x} \\ &= \int P[(\phi_{2y}, -\phi_{2x}) \cdot \nabla\delta u_1 + (-\phi_{1y}, \phi_{1x}) \cdot \nabla\delta u_2]d\boldsymbol{x}. \end{aligned}$$

Define $\boldsymbol{a_1} = -P(\phi_{2y}, -\phi_{2x}), \boldsymbol{a_2} = -P(-\phi_{1y}, \phi_{1x})$, and introduce g_1, g_2 satisfying

$$\begin{cases} \Delta g_1 &= \nabla \cdot \boldsymbol{a_1}, \\ \Delta g_2 &= \nabla \cdot \boldsymbol{a_2}. \end{cases} \tag{3.6}$$

Consequently,

$$\delta ssd = \int (-\boldsymbol{a_1} \cdot \nabla\delta u_1 - \boldsymbol{a_2} \cdot \nabla\delta u_2)d\boldsymbol{x}.$$

Before continuing, we need to recall two Corollaries of divergence theorem here.

Corollary 1. *By Applying the divergence theorem to the product of a scalar function v and a vector field $\boldsymbol{u}$ on domain Ω, we have*

$$\int_\Omega (\boldsymbol{u} \cdot \nabla v + v(\nabla \cdot \boldsymbol{u}))dV = \int_{\partial\Omega} (v\boldsymbol{u} \cdot \boldsymbol{n})dS,$$

where $\boldsymbol{n}$ is the outward unit normal vector.

Corollary 2. *By Applying the divergence theorem to the product of a scalar function v and a vector field ∇w on domain Ω, we have*

$$\int_{\Omega}(\nabla w \cdot \nabla v + v(\Delta w))dV = \int_{\partial\Omega}(v\nabla w \cdot \boldsymbol{n})dS,$$

where $\boldsymbol{n}$ is the outward unit normal vector.

Apply Corollary 1 ($*$) and Corollary 2 ($**$):

$$\begin{aligned}
\delta ssd &= \int(-\boldsymbol{a_1} \cdot \nabla\delta u_1 - \boldsymbol{a_2} \cdot \nabla\delta u_2)d\boldsymbol{x} \\
&= \int(\nabla \cdot \boldsymbol{a_1}\delta u_1 + \nabla \cdot \boldsymbol{a_2}\delta u_2)d\boldsymbol{x} \qquad (*) \\
&= \int(\Delta g_1\delta u_1 + \Delta g_2\delta u_2)d\boldsymbol{x} \\
&= \int(g_1\delta\Delta u_1 + g_2\delta\Delta u_2)d\boldsymbol{x} \qquad (**) \\
&= \int(g_1\delta f_x + g_2\delta f_y)d\boldsymbol{x} \\
&= \int((g_1, g_2) \cdot \nabla\delta f)d\boldsymbol{x} \\
&= \int(-\nabla \cdot (g_1, g_2))\delta f d\boldsymbol{x}.
\end{aligned}$$

Finally, we obtain the variational derivative

$$\frac{\partial ssd}{\partial f} = -\nabla \cdot (g_1, g_2) = -(g_{1x} + g_{2y}).$$

3.2.3 Algorithm implementation

The gradient descend optimization algorithm can be briefly described as follows:

1. **Initialize** $\boldsymbol{\phi}_{old} = \boldsymbol{id}$, $f = 0$.
2. **Compute** $P, \boldsymbol{a}_1, \boldsymbol{a}_2, \nabla \cdot \boldsymbol{a_1}, \nabla \cdot \boldsymbol{a_2}$.
3. **Solve** Poisson's equations (3.6) to get g_1, g_2, and then $\frac{\delta ssd}{\delta f}$.

4. **Update** f by $f_{new} = f_{old} - \frac{\delta ssd}{\delta f} \times tstepdt$, where $tstep$ is an optimization parameter.
5. **Solve** Poisson's equations (3.5) to get u_1, u_2.
6. **Update** $\boldsymbol{\phi}$ by (3.3).
7. **Back** to 2, keep iterating until a preset tolerance or a preset number of iteration steps is reached.

3.3 Development of New Variational Method: Version 2

3.3.1 Discussion and improvements

Later, we made some improvements as follows:

1. The Jacobian determinant of a transformation alone usually can not determine the transformation itself uniquely. Namely, the nonlinear equation $\det\nabla\boldsymbol{\phi} = f_0$ is not guaranteed to have an unique solution. Inspired by the *div-curl-system* in *the deformation method*, we added the curl vector into the cost functional, therefore extended the similarity measure (3.2) to

$$ssd = \frac{1}{2}\int_\Omega [(J(\boldsymbol{\phi}(\boldsymbol{x})) - f_0(\boldsymbol{x}))^2 + \alpha(\mathrm{curl}(\boldsymbol{\phi}(\boldsymbol{x})) - \boldsymbol{g_0}(\boldsymbol{x}))^2]d\boldsymbol{x}. \tag{3.7}$$

Here $\alpha \geq 0$ is a weight parameter, f_0 is the prescribed Jacobian determinant monitor function, and $\boldsymbol{g_0}$ is the prescribed curl vector monitor function.

So the problem we study now is *to construct numerically a transformation* $\boldsymbol{\phi} : \Omega \to \Omega$*, such that, the Jacobian determinant of* $\boldsymbol{\phi}$*, i.e.* $J(\boldsymbol{\phi})$*, equals a prescribed monitor function* f_0*, and the curl of* $\boldsymbol{\phi}$*, i.e.* $curl(\boldsymbol{\phi})$*, equals a prescribed monitor function* $\boldsymbol{g_0}$.

2. Removed the artificial time step in (3.3), directly let

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}). \tag{3.8}$$

3. The constraint (3.4) were generalized to

$$\begin{cases} \text{div } \boldsymbol{u} & = f & \text{in } \Omega, \\ \text{curl } \boldsymbol{u} & = \boldsymbol{g} & \text{in } \Omega, \\ \boldsymbol{u} & = \boldsymbol{0} & \text{on } \partial\Omega. \end{cases} \tag{3.9}$$

Which leads to (in 2D)

$$\begin{cases} \Delta u_1 & = f_x - g_y := f_1, \\ \Delta u_2 & = f_y + g_x := f_2. \end{cases} \tag{3.10}$$

We then used f_1 and f_2 as the control functions and directly set:

$$\Delta \boldsymbol{u} = \boldsymbol{f} = (f_1, f_2) \tag{3.11}$$

with fixed boundary conditions.

Combining (3.7), (3.8) and (3.11), we name the new version *new variational method with prescribed Jacobian determinant and curl vector*. Next, let's derive the variational gradient for the new set-up.

3.3.2 Derivation of the gradient: 2D

Let $\boldsymbol{\phi} = (\phi_1, \phi_2), \boldsymbol{u} = (u_1, u_2), \boldsymbol{f} = (f_1, f_2)$, and take $\alpha = 1$ for the simplicity of presentation (See Remark 11 at the end of this section for the general cases). The new variational method: version 2 is to minimize

$$ssd = \frac{1}{2} \iint_{\Omega} [(J(\boldsymbol{\phi}) - f_0)^2 + (\text{curl}(\boldsymbol{\phi}) - g_0)^2] dA$$

subject to the constraints that the displacement $\boldsymbol{u}$ satisfies:

$$\begin{cases} \Delta u_1 & = f_1, \\ \Delta u_2 & = f_2 & \text{in } \Omega, \\ u_1, u_2 & = 0 & \text{on } \partial\Omega. \end{cases} \tag{3.12}$$

For arbitrary δf_1 and δf_2 that vanish on $\partial\Omega$,

$$\begin{aligned}\delta ssd &= \iint_\Omega (J(\boldsymbol{\phi}) - f_0)\delta J(\boldsymbol{\phi}) + (\text{curl}(\boldsymbol{\phi}) - g_0)\delta\text{curl}(\boldsymbol{\phi})dA \\ &= \iint_\Omega [(J(\boldsymbol{\phi}) - f_0)\delta(\phi_{1x}\phi_{2y} - \phi_{1y}\phi_{2x}) + (\text{curl}(\phi) - g_0)\delta(\phi_{2x} - \phi_{1y})]dA \\ &= \iint_\Omega [(J(\boldsymbol{\phi}) - f_0)(\delta\phi_{1x}\phi_{2y} + \phi_{1x}\delta\phi_{2y} - \delta\phi_{1y}\phi_{2x} - \phi_{1y}\delta\phi_{2x}) \\ &+ (\text{curl}(\boldsymbol{\phi}) - g_0)(\delta\phi_{2x} - \delta\phi_{1y})]dA.\end{aligned}$$

From (3.8), we know $\delta\boldsymbol{\phi} = \delta\boldsymbol{u}$, so $\delta\phi_{ix} = \delta u_{ix}, \delta\phi_{iy} = \delta u_{iy}, i = 1, 2$. Plug it back in the above equation, we get

$$\begin{aligned}\delta ssd = \iint_\Omega [&(J(\boldsymbol{\phi}) - f_0)(\delta u_{1x}\phi_{2y} + \phi_{1x}\delta u_{2y} - \delta u_{1y}\phi_{2x} - \phi_{1y}\delta u_{2x}) \\ &+ (\text{curl}(\boldsymbol{\phi}) - g_0)(\delta u_{2x} - \delta u_{1y})]dA.\end{aligned}$$

Let $P = (J(\boldsymbol{\phi}) - f_0), Q = (\text{curl}(\boldsymbol{\phi}) - g_0)$, we have

$$\delta ssd = \iint_\Omega [P(\delta u_{1x}\phi_{2y} + \phi_{1x}\delta u_{2y} - \delta u_{1y}\phi_{2x} - \phi_{1y}\delta u_{2x}) + Q(\delta u_{2x} - \delta u_{1y})]dA.$$

Notice that $\nabla\delta u_i = (\delta u_{ix}, \delta u_{iy}), i = 1, 2$, so

$$\begin{aligned}\delta ssd = \iint_\Omega [&P((\phi_{2y}, -\phi_{2x}) \cdot \nabla\delta u_1 + (-\phi_{1y}, \phi_{1x}) \cdot \nabla\delta u_2) \\ &+ Q((0, -1) \cdot \nabla\delta u_1 + (1, 0) \cdot \nabla\delta u_2)]dA.\end{aligned}$$

Define

$$\begin{cases} -\boldsymbol{a}_1 &= P(\phi_{2y}, -\phi_{2x}) + Q(0, -1), \\ -\boldsymbol{a}_2 &= P(-\phi_{1y}, \phi_{1x}) + Q(1, 0). \end{cases} \tag{3.13}$$

We now get

$$\delta ssd = \iint_\Omega (-\boldsymbol{a}_1 \cdot \nabla\delta u_1 - \boldsymbol{a}_2 \cdot \nabla\delta u_2)dA.$$

Recall that $u_i = 0$ on $\partial\Omega$, so $\delta u_i = 0$ on $\partial\Omega$. By Corollary 1,

$$\begin{aligned}\delta ssd &= \iint_\Omega (-\boldsymbol{a}_1 \cdot \nabla\delta u_1 - \boldsymbol{a}_2 \cdot \nabla\delta u_2)dA \\ &= \iint_\Omega ((\nabla \cdot \boldsymbol{a}_1)\delta u_1 + (\nabla \cdot \boldsymbol{a}_2)\delta u_2)dA - \int_{\partial\Omega} (\delta u_1(\boldsymbol{a}_1 \cdot \boldsymbol{n}) + \delta u_2(\boldsymbol{a}_2 \cdot \boldsymbol{n}))dS \\ &= \iint_\Omega ((\nabla \cdot \boldsymbol{a}_1)\delta u_1 + (\nabla \cdot \boldsymbol{a}_2)\delta u_2)dA.\end{aligned}$$

Now Define $\boldsymbol{g} = (g_1, g_2)$ as

$$\begin{cases} \Delta g_1 & = \nabla \cdot \boldsymbol{a}_1, \\ \Delta g_2 & = \nabla \cdot \boldsymbol{a}_2, \\ g_1, g_2 & = 0 \qquad on\ \partial\Omega. \end{cases} \tag{3.14}$$

Then,

$$\delta ssd = \iint_{\Omega} (\Delta g_1 \delta u_1 + \Delta g_2 \delta u_2) dA.$$

By the Corollary 2, for $i = 1, 2$, we get

$$\begin{cases} \iint_{\Omega} (\nabla g_i \cdot \nabla \delta u_i + \delta u_i \Delta g_i) dA & = \int_{\partial\Omega} (\delta u_i \nabla g_i \cdot \boldsymbol{n}) dS & = 0, \\ \iint_{\Omega} (\nabla g_i \cdot \nabla \delta u_i + g_i \Delta \delta u_i) dA & = \int_{\partial\Omega} (g_i \nabla \delta u_i \cdot \boldsymbol{n}) dS & = 0. \end{cases}$$

Finally,

$$\begin{aligned} \delta ssd &= \iint_{\Omega} (\Delta g_1 \delta u_1 + \Delta g_2 \delta u_2) dA \\ &= \iint_{\Omega} (g_1 \delta(\Delta u_1) + g_2 \delta(\Delta u_2)) dA \\ &= \iint_{\Omega} (g_1 \delta f_1 + g_2 \delta f_2) dA, \end{aligned}$$

which is

$$\frac{\partial ssd}{\partial f_i} = g_i, i = 1, 2. \tag{3.15}$$

Remark 11. *At the end of derivation, we want to point out that if we change the identity map $\boldsymbol{x}$ to any given transformation $\boldsymbol{\phi}^*(\boldsymbol{x})$ in (3.8), the derivation above is still right. Generally speaking, (3.8) can be replaced by*

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi}^*(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}), \tag{3.8*}$$

where $\boldsymbol{\phi}^*(\boldsymbol{x})$ is any given transformation. This means the construction can start with any given transformation, the identity transformation is just a special case.

Remark 12. *For arbitrary weight parameter α in (3.7), the derivation above is still right just by changing $Q = \alpha(curl(\boldsymbol{\phi}) - g_0)$.*

Remark 13. *The fixed boundary conditions in (3.12) and (3.14) ($\boldsymbol{u} = \mathbf{0}$ and $\boldsymbol{g} = \mathbf{0}$ on $\partial\Omega$) are actually needed by the 2 Corollaries to move the gradient operator ($\nabla\cdot$) or the laplace operator (Δ). The fixed boundary conditions are trivial ones that can make the 2 Corollaries work. It is clear that the Neumann boundary conditions $\nabla u_i \cdot \boldsymbol{n} = 0$ and $\nabla g_i \cdot \boldsymbol{n} = 0$ can make Corollary 2 work, but may fail Corollary 1. Generally speaking, in theory, it seems the only correct boundary condition is the fixed boundary condition. However, in numerical practice, Neumann boundary conditions can also lead satisfied results. Related discussion continues at the end of this chapter.*

3.3.3 Derivation of the gradient: 3D

The general derivation is similar for three-dimensional case. The only major difference is the component of $\boldsymbol{a}_i$ in (3.13).

Let $\boldsymbol{\phi} = (\phi_1, \phi_2, \phi_3), \boldsymbol{u} = (u_1, u_2, u_3), \boldsymbol{f} = (f_1, f_2, f_3), \boldsymbol{g_0} = (g_{01}, g_{02}, g_{03})$, and take $\alpha = 1$.

The cost functional is

$$ssd = \frac{1}{2}\iiint_{\Omega}[(J(\boldsymbol{\phi}) - f_0)^2 + (\mathrm{curl}(\boldsymbol{\phi}) - \boldsymbol{g_0})^2]dV,$$

subject to

$$\begin{cases} \Delta u_1 & = f_1, \\ \Delta u_2 & = f_2, \\ \Delta u_3 & = f_3 \qquad in\ \Omega, \\ u_1, u_2, u_3 & = 0 \qquad on\ \partial\Omega. \end{cases} \tag{3.16}$$

Now,

$$J(\boldsymbol{\phi}) = \phi_{1x}(\phi_{2y}\phi_{3z} - \phi_{3y}\phi_{2z}) - \phi_{2x}(\phi_{1y}\phi_{3z} - \phi_{3y}\phi_{1z}) + \phi_{3x}(\phi_{1y}\phi_{2z} - \phi_{2y}\phi_{1z}),$$

and

$$\mathrm{curl}(\boldsymbol{\phi}) = (\phi_{3y} - \phi_{2z}, \phi_{1z} - \phi_{3x}, \phi_{2x} - \phi_{1y}).$$

Let $P = (J(\boldsymbol{\phi}) - f_0), Q_1 = (\phi_{3y} - \phi_{2z} - g_{01}), Q_2 = (\phi_{1z} - \phi_{3x} - g_{02}), Q_3 = (\phi_{2x} - \phi_{1y} - g_{03})$.

For arbitrary $\delta f_i (i = 1, 2, 3)$ that vanish on $\partial\Omega$, we have

$$
\begin{aligned}
\delta ssd &= \int_\Omega (J(\boldsymbol{\phi}) - f_0)\delta J(\boldsymbol{\phi}) + (\mathrm{curl}(\boldsymbol{\phi}) - \boldsymbol{g_0}) \cdot \delta \mathrm{curl}(\boldsymbol{\phi}) dV \\
&= \int_\Omega [P\delta(\phi_{1x}(\phi_{2y}\phi_{3z} - \phi_{2y}\phi_{3z}) - \phi_{2x}(\phi_{1y}\phi_{3z} - \phi_{3y}\phi_{1z}) + \phi_{3x}(\phi_{1y}\phi_{2z} - \phi_{2y}\phi_{1z})) \\
&+ (Q_1, Q_2, Q_3) \cdot \delta(\phi_{3y} - \phi_{2z}, \phi_{1z} - \phi_{3x}, \phi_{2x} - \phi_{1y})] dV \\
&= \int_\Omega [P(\delta u_{1x}\phi_{2y}\phi_{3z} + \delta u_{2y}\phi_{1x}\phi_{3z} + \delta u_{3z}\phi_{1x}\phi_{2y} - \delta u_{1x}\phi_{3y}\phi_{2z} - \delta u_{2z}\phi_{1x}\phi_{3y} \\
&- \delta u_{3y}\phi_{1x}\phi_{2z} - \delta u_{1y}\phi_{2x}\phi_{3z} - \delta u_{2x}\phi_{1y}\phi_{3z} - \delta u_{3z}\phi_{2x}\phi_{1y} + \delta u_{1z}\phi_{2x}\phi_{3y} + \delta u_{2x}\phi_{3y}\phi_{1z} \\
&+ \delta u_{3y}\phi_{2x}\phi_{1z} + \delta u_{1y}\phi_{3x}\phi_{2z} + \delta u_{2z}\phi_{3x}\phi_{1y} + \delta u_{3x}\phi_{1y}\phi_{2z} - \delta u_{1z}\phi_{3x}\phi_{2y} - \delta u_{2y}\phi_{3x}\phi_{1z} \\
&- \delta u_{3x}\phi_{2y}\phi_{1z}) + Q_1(\delta u_{3y} - \delta\phi_{2z}) + Q_2(\delta u_{1z} - \delta\phi_{3x}) + Q_3(\delta u_{2x} - \delta\phi_{1y})] dV \\
&= \int_\Omega [P(\delta u_{1x}, \delta u_{1y}, \delta u_{1z}) \cdot (\phi_{2y}\phi_{3z} - \phi_{3y}\phi_{2z}, \phi_{3x}\phi_{2z} - \phi_{2x}\phi_{3z}, \phi_{2x}\phi_{3y} - \phi_{2y}\phi_{3x}) \\
&+ P(\delta u_{2x}, \delta u_{2y}, \delta u_{2z}) \cdot (\phi_{3y}\phi_{1z} - \phi_{1y}\phi_{3z}, \phi_{1x}\phi_{3z} - \phi_{1z}\phi_{3x}, \phi_{3x}\phi_{1y} - \phi_{1x}\phi_{3y}) \\
&+ P(\delta u_{3x}, \delta u_{3y}, \delta u_{3z}) \cdot (\phi_{1y}\phi_{2z} - \phi_{2y}\phi_{1z}, \phi_{2x}\phi_{1z} - \phi_{1x}\phi_{2z}, \phi_{1x}\phi_{2y} - \phi_{2x}\phi_{1y}) \\
&+ (\delta u_{1x}, \delta u_{1y}, \delta u_{1z}) \cdot (0, -Q_3, Q_2) + (\delta u_{2x}, \delta u_{2y}, \delta u_{2z}) \cdot (Q_3, 0, -Q_1) \\
&+ (\delta u_{3x}, \delta u_{3y}, \delta u_{3z}) \cdot (-Q_2, Q_1, 0)] dV.
\end{aligned}
$$

Define

$$
\begin{cases}
-\boldsymbol{a}_1 &= P(\phi_{2y}\phi_{3z} - \phi_{3y}\phi_{2z}, \phi_{3x}\phi_{2z} - \phi_{2x}\phi_{3z}, \phi_{2x}\phi_{3y} - \phi_{2y}\phi_{3x}) \\
&\quad + (0, -Q_3, Q_2), \\
-\boldsymbol{a}_2 &= P(\phi_{3y}\phi_{1z} - \phi_{1y}\phi_{3z}, \phi_{1x}\phi_{3z} - \phi_{1z}\phi_{3x}, \phi_{3x}\phi_{1y} - \phi_{1x}\phi_{3y}) \\
&\quad + (Q_3, 0, -Q_1), \\
-\boldsymbol{a}_3 &= P(\phi_{1y}\phi_{2z} - \phi_{2y}\phi_{1z}, \phi_{2x}\phi_{1z} - \phi_{1x}\phi_{2z}, \phi_{1x}\phi_{2y} - \phi_{2x}\phi_{1y}) \\
&\quad + (-Q_2, Q_1, 0),
\end{cases}
\tag{3.17}
$$

and $\boldsymbol{g} = (g_1, g_2, g_3)$ satisfies

$$
\begin{cases}
\Delta g_1 &= \nabla \cdot \boldsymbol{a}_1, \\
\Delta g_2 &= \nabla \cdot \boldsymbol{a}_2, \\
\Delta g_3 &= \nabla \cdot \boldsymbol{a}_3, \\
g_1, g_2, g_3 &= 0 \qquad on\ \partial\Omega.
\end{cases}
\tag{3.18}
$$

We can get

$$\begin{aligned}\delta ssd &= \int_\Omega(-\boldsymbol{a}_1\cdot\nabla\delta u_1-\boldsymbol{a}_2\cdot\nabla\delta u_2-\boldsymbol{a}_3\cdot\nabla\delta u_3)dV\\ &=\dots\\ &=\int_\Omega(g_1\delta f_1+g_2\delta f_2+g_3\delta f_3)dV.\end{aligned}$$

Exactly,

$$\frac{\partial ssd}{\partial f_i}=g_i, i=1,2,3. \tag{3.19}$$

3.3.4 Algorithm implementation

We can implement this new version by similar gradient descent optimization scheme as follows:

1. **Initialize** $\boldsymbol{\phi}=\boldsymbol{id}, \boldsymbol{u}=\mathbf{0}, f=\mathbf{0}$.
2. **Compute** $\boldsymbol{a}_i$ by (3.13 or 3.17), then **solve** Poisson's equation (3.14 or 3.18) to get g_i.
3. **Update** f by $f_{i,new}=f_{i,old}-g_i\times tstep$, where $tstep$ is an optimization parameter.
4. **Solve** Poisson's equation (3.12 or 3.16) to get u_i.
5. **Update** $\boldsymbol{\phi}$ by (3.8).
6. **Back** to 2, keep iterating until a preset tolerance or a preset number of iteration steps is reached.

3.4 Development of New Variational Method: Version 3

The last version considers $f_0(\boldsymbol{\phi}(\boldsymbol{x}))$ and $\boldsymbol{g_0}(\boldsymbol{\phi}(\boldsymbol{x}))$ instead of $f_0(\boldsymbol{x})$ and $\boldsymbol{g_0}(\boldsymbol{x})$ in the cost functional. Namely define ssd as:

$$ssd=\frac{1}{2}\int_\Omega[(J(\boldsymbol{\phi})-f_0(\boldsymbol{\phi}))^2+\alpha(\mathrm{curl}(\boldsymbol{\phi})-\boldsymbol{g_0}(\boldsymbol{\phi}))^2]d\boldsymbol{x}. \tag{3.20}$$

In 2D case, keep the same constraints (3.12) and definition (3.13), and let $P = (J(\boldsymbol{\phi}) - f_0(\boldsymbol{\phi})), Q = (\text{curl}(\boldsymbol{\phi}) - g_0(\boldsymbol{\phi}))$ correspondingly. We can make the similar derivation:

$$\begin{aligned}
\delta ssd &= \int_\Omega [P(\delta J(\boldsymbol{\phi}) - \delta f_0(\boldsymbol{\phi})) + Q(\delta \text{curl}(\boldsymbol{\phi}) - \delta g_0(\boldsymbol{\phi}))] d\boldsymbol{x} \\
&= \int_\Omega [P\delta J(\boldsymbol{\phi}) + Q\delta \text{curl}(\boldsymbol{\phi}) - P\delta f_0(\boldsymbol{\phi}) - Q\delta g_0(\boldsymbol{\phi})] d\boldsymbol{x} \\
&= \int_\Omega [(\nabla \cdot \boldsymbol{a}_1)\delta u_1 + (\nabla \cdot \boldsymbol{a}_2)\delta u_2 - P\delta f_0(\boldsymbol{\phi}) - Q\delta g_0(\boldsymbol{\phi})] d\boldsymbol{x} \\
&= \int_\Omega [(\nabla \cdot \boldsymbol{a}_1)\delta u_1 + (\nabla \cdot \boldsymbol{a}_2)\delta u_2 \\
&- P\nabla f_0(\boldsymbol{\phi}) \cdot (\delta u_1, \delta u_2) - Q\nabla g_0(\boldsymbol{\phi}) \cdot (\delta u_1, \delta u_2)] d\boldsymbol{x} \\
&= \int_\Omega [(\nabla \cdot \boldsymbol{a}_1 - Pf_{0x}(\boldsymbol{\phi}) - Qg_{0x}(\boldsymbol{\phi}))\delta u_1 + (\nabla \cdot \boldsymbol{a}_2 - Pf_{0y}(\boldsymbol{\phi}) - Qg_{0y}(\boldsymbol{\phi}))\delta u_2] d\boldsymbol{x}.
\end{aligned}$$

Now define $\boldsymbol{g} = (g_1, g_2)$ by

$$\begin{cases}
\Delta g_1 &= \nabla \cdot \boldsymbol{a}_1 - Pf_{0x}(\boldsymbol{\phi}) - Qg_{0x}(\boldsymbol{\phi}), \\
\Delta g_2 &= \nabla \cdot \boldsymbol{a}_2 - Pf_{0y}(\boldsymbol{\phi}) - Qg_{0y}(\boldsymbol{\phi}), \\
g_1, g_2 &= 0 \qquad on\ \partial\Omega.
\end{cases} \tag{3.21}$$

Again,

$$\frac{\partial ssd}{\partial f_i} = g_i, i = 1, 2.$$

3.5 Numerical examples

Several numerical examples of constructing transformations are illustrated in this section. The first 4 examples in 2D take $g_0 = 0, \alpha = 1$ and corresponding prescribed Jacobian determinant monitor functions f_0.

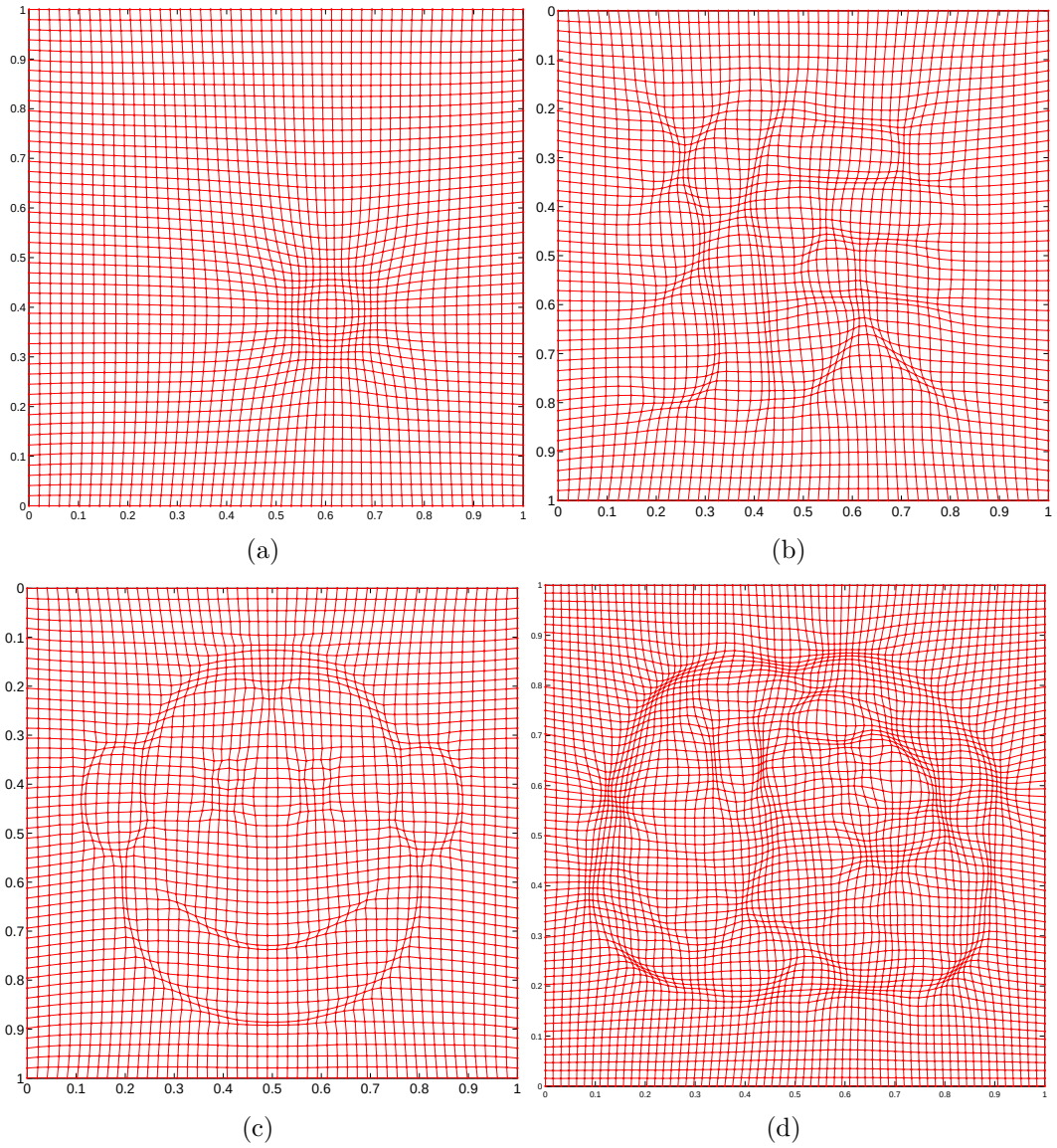


Figure 3.1. Illustration of some 2D numerical transformations constructed by the new variational method with prescribed Jacobian determinant and prescribed curl vector.

The next examples illustrate two 3D transformations, iso-surfaces of the Jacobian determinant (i.e. the volume size) are used to show the interior structure of the 3D transformations. In both examples, set $\alpha = 0.01, \boldsymbol{g}_0 = \mathbf{0}$. Figure 3.2 shows a trans-

formation which concentrates toward a annulus, Figure 3.3 shows a transformation which moves toward a sphere's surface.

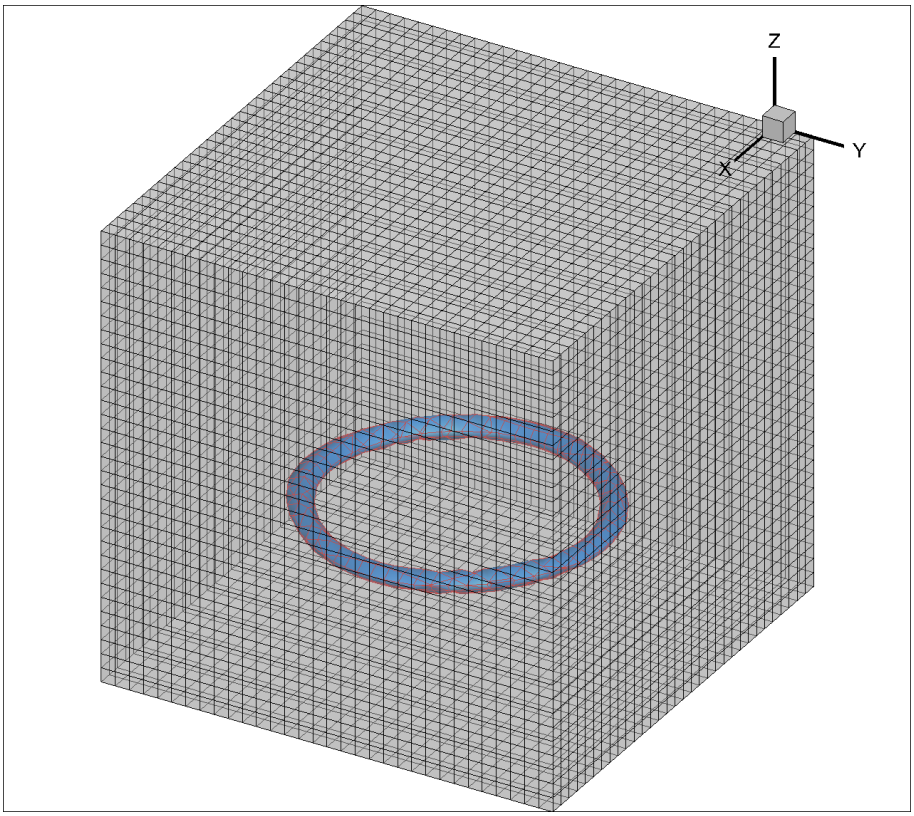


Figure 3.2. 3D transformation concentrates toward a annulus constructed by the new variational method with prescribed Jacobian determinant and prescribed curl vector.

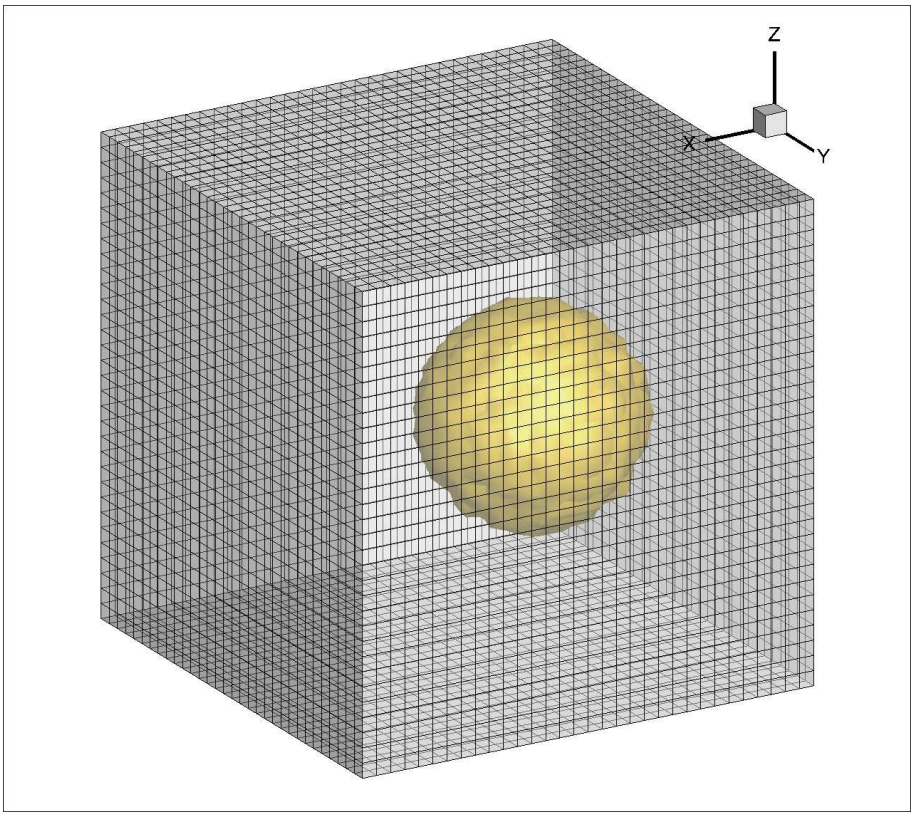


Figure 3.3. 3D transformation moves toward a sphere's surface constructed by the new variational method with prescribed Jacobian determinant and prescribed curl vector.

All examples here are with fixed boundary conditions, as discussed in Remark 13. But it doesn't mean this method can not deal with moving boundary problems, we have at least the following 3 approaches:

1. *Boundary match.* By Remark 11, we can use a transformation $\boldsymbol{\phi}^*(\boldsymbol{x})$ to move the boundary first, then it will be a fixed boundary problem.
2. *Domain padding.* We can pad the domain Ω a little bit to Ω^*, do a fixed boundary problem on Ω^*, then map back to Ω.
3. *The deformation method.* We can combine with *the deformation method* to deal with the boundary or even domain issues.

3.6 Conclusion

The above examples show the construction of transformations by *the new variational method with prescribed Jacobian determinant and prescribed curl vector* is successful in practice. However, we need to admit that, unlike *the deformation method*, *the new variational method* is a numerical method other than a theoretical solution. We can not claim that the constructed transformation $\boldsymbol{\phi}$ is definitely a diffeomorphism. Although, $J(\boldsymbol{\phi}) > 0$ and fixed boundary condition does imply that $\boldsymbol{\phi}$ is a diffeomorphism in theory [24, 25]. We also leave out the convergence analysis of the gradient descend optimization scheme.

At this point, the two methods of constructing diffeomorphisms/transfromations have been presented in detail. At last, we want to emphasize the importance of the function $f(\boldsymbol{x}, t)$ in *the deformation method* and $f_0(\boldsymbol{x})$ in *the variational method*. The quality of the transformation constructed highly depends on the quality of these 2 functions. Besides their normalization conditions (2.7,2.8) and (3.1), more importantly, they have to be reasonable. Namely, there should exist a diffeomorphism $\boldsymbol{\phi}$ such that $J(\boldsymbol{\phi}) = f(\boldsymbol{\phi}, t)$, or a transformation $\boldsymbol{\phi}$ such that $J(\boldsymbol{\phi}) = f$ before we try to construct it. In other words, if given a f, but there is no such a transformation such that $J(\boldsymbol{\phi}) = f$, of course we can not construct a transformation successfully. Again, how to define a function $f(\boldsymbol{x}, t)$ or $f_0(\boldsymbol{x})$ properly is the key issue in the applications of the 2 methods to grid generation problems as well as other potential areas.

3.7 Graphical User Interface: GridPanel

We also designed a graphical user interface named "GridPanel" to demonstrate our two methods. It can generate a grid based on the lines you draw automatically and instantaneously. The next figures show some works by GridPanel.

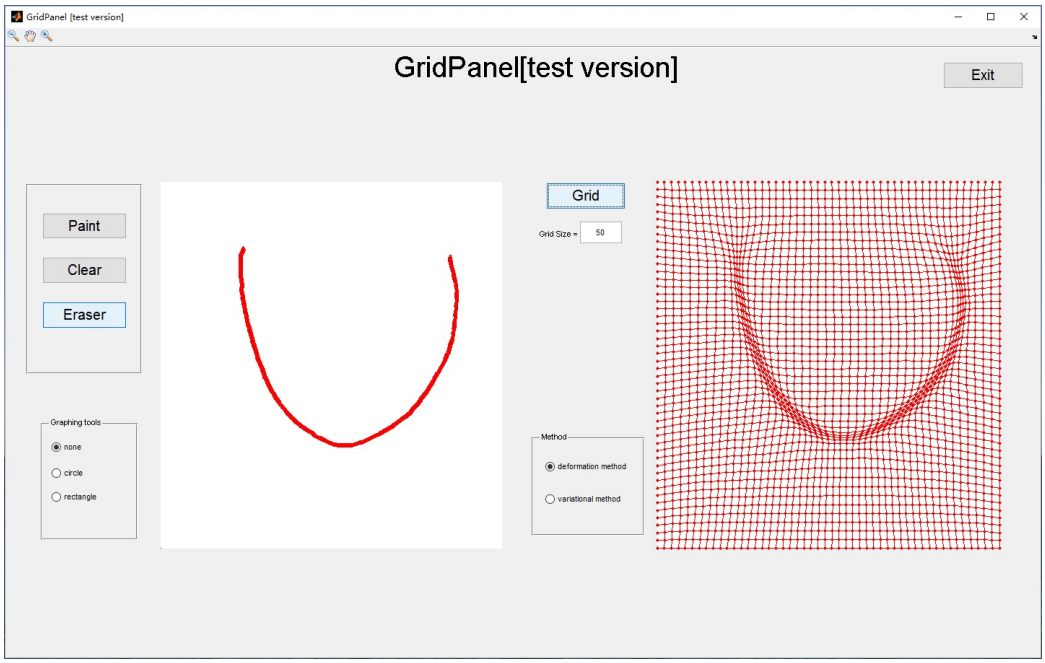


Figure 3.4. GridPanel example.

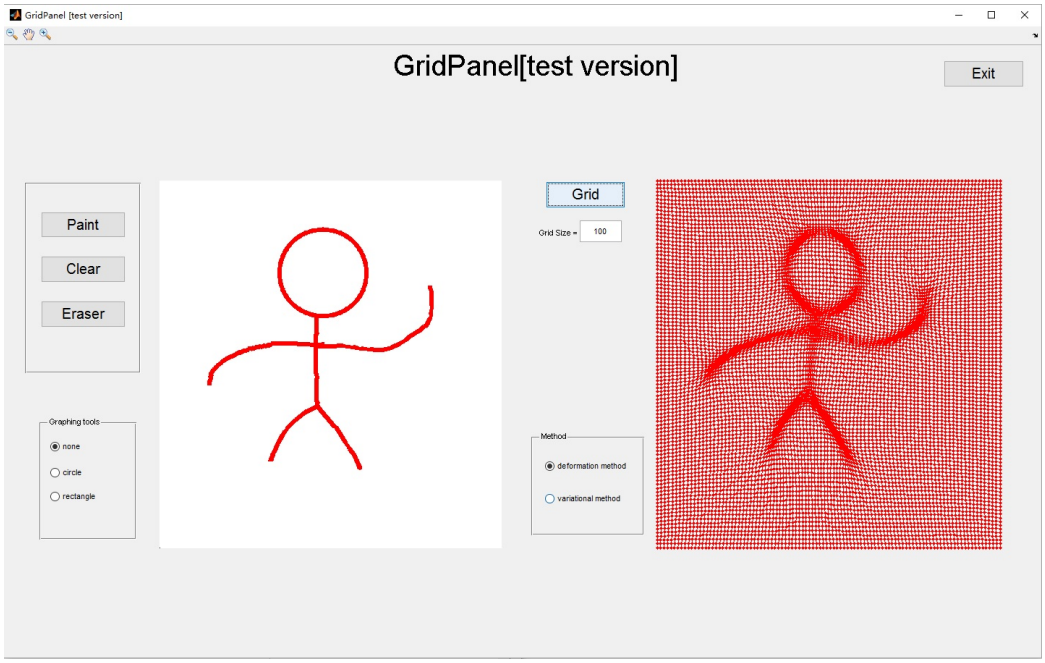


Figure 3.5. GridPanel example.

CHAPTER 4

STUDY ON THE UNIQUENESS OF TRANSFORMATION WITH JACOBIAN DETERMINANT AND CURL VECTOR

4.1 Introduction

In the study of the two transformation construction methods, we deeply learnt the importance of the curl vector curlϕ in the form of a transformation. A natural question is how important it is? In the theory of differential geometry, we already know that the divergence $\nabla\phi$ and the curl vector $\nabla \times \phi$ together can determine a transformation ϕ uniquely under certain boundary condtions. Recall *the variational method with prescribed jacobian determinant and curl vector*, we use $J(\phi)$ and curlϕ to construct a transformation ϕ, so is there a uniqueness statement also? Namely, can the Jacobian determinant and the curl vector together determine a transformation uniquely under a general condition? If not, are there any stronger conditions such that the uniqueness is guaranteed? Intuitively, the Jacobian determinant characterizes the local volume size, and the curl vector characterizes the local rotation, it seems these two determine the local behavior of a transformation.

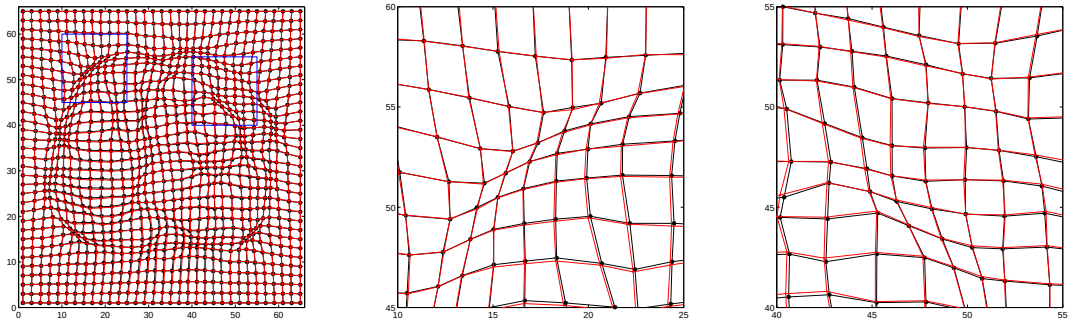
In order to study this uniqueness problem, we first designed a recovering experiment, and achieved positive observations from numerical tests. These numerical results inspired us to further explore the problem in Mathematics. Later, we proved a related statement under a strong condition in 2D. All these will be discussed in this chapter. And Let's start with the recovering experiment.

4.2 Experiments of Recovering Transformations

Recall *the variational method*, the constructed $\boldsymbol{\phi}$ should satisfy $J(\boldsymbol{\phi}) = f_0(x)$ and $\text{curl}(\boldsymbol{\phi}) = \boldsymbol{g}_0(x)$. Now, if we take a known transformation $\boldsymbol{\phi}_0 : \Omega \to \Omega$, compute its Jacobian determinant as our f_0 and curl vector as $\boldsymbol{g}_0$, namely set $f_0(x) = J(\boldsymbol{\phi}_0(x)), \boldsymbol{g}_0(x) = \text{curl}(\boldsymbol{\phi}_0(x))$, then apply *the variational method*. What will we get? Will the constructed $\boldsymbol{\phi}$ undoubtedly be the known transformation $\boldsymbol{\phi}_0$? If it is true, does it mean the transformation with the Jacobian determinant f_0 and curl vector $\boldsymbol{g}_0$ is unique as $\boldsymbol{\phi}_0$? Consequently, we carried out this procedure by numerical examples, compared the constructed $\boldsymbol{\phi}$ and $\boldsymbol{\phi}_0$ in details. This is our experiment of recovering transformations.

The following figures and tables illustrate corresponding results of two experiments. In each experiment, we tried different values of $\alpha : 0, 1, 0.1, 10$. Notice that if $\alpha = 0$, only the Jacobian determinant is used, curl vector is not used.

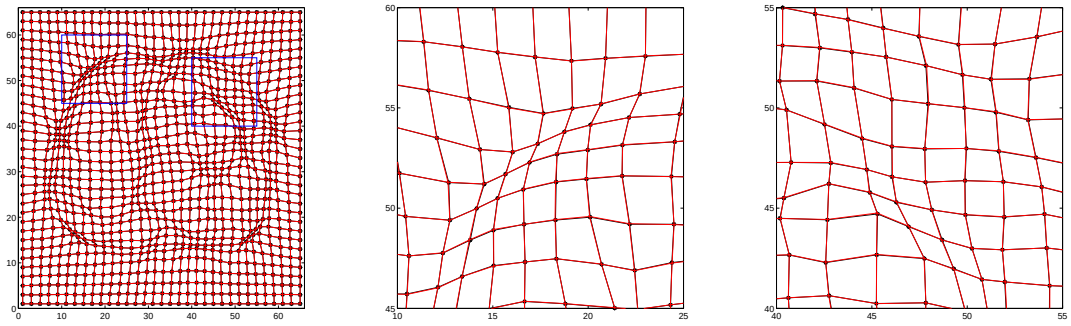
Our first experiment uses a "brain-like" 2D transformation. Random gaussian noise is added to get $\boldsymbol{\phi}_0$. Figure 4.1,4.2 and table 4.1 show the results and comparison.



(a) $\alpha = 0$, curl not used

(b) $\alpha = 0$, enlarged view of left rectangle

(c) $\alpha = 0$, enlarged view of right rectangle

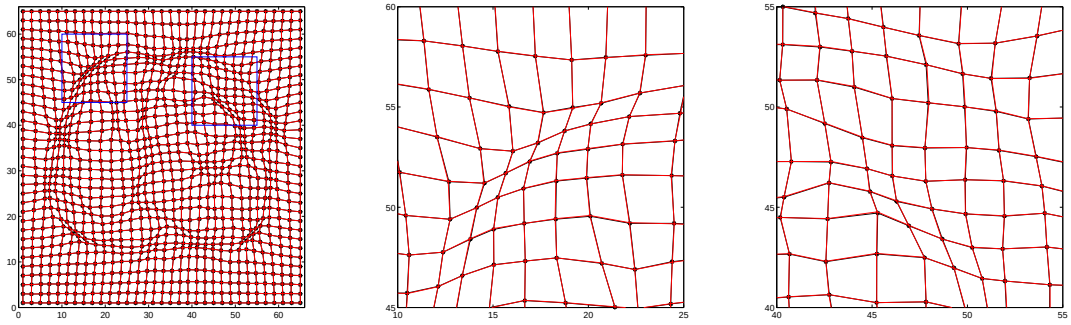


(d) $\alpha = 1$

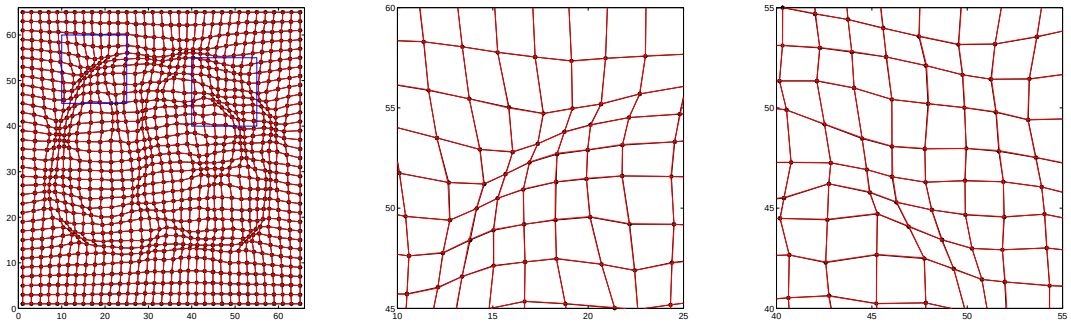
(e) $\alpha = 1$, enlarged view of left rectangle

(f) $\alpha = 1$, enlarged view of right rectangle

Figure 4.1. Experiment 1, 65×65 grid nodes. The black star dots $*$ represent $\boldsymbol{\phi}_0$, and red dots $\cdot$ represent constructed $\boldsymbol{\phi}$.



(a) $\alpha = 0.1$ (b) $\alpha = 0.1$, enlarged view of left rectangle (c) $\alpha = 0.1$, enlarged view of right rectangle



(d) $\alpha = 10$ (e) $\alpha = 10$, enlarged view of left rectangle (f) $\alpha = 10$, enlarged view of right rectangle

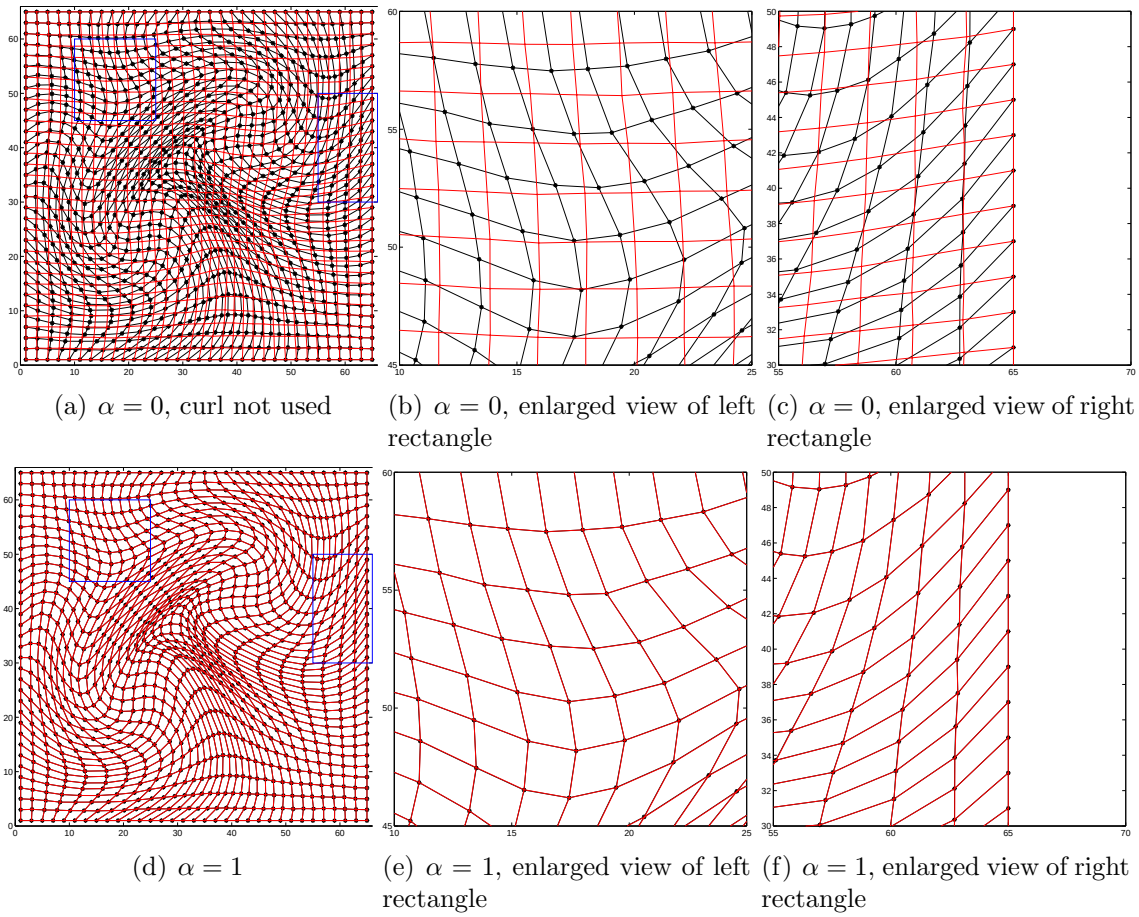
Figure 4.2. Experiment 1-continued, 65×65 grid nodes. The black star dots $*$ represent $\boldsymbol{\phi}_0$, and red dots $\cdot$ represent constructed $\boldsymbol{\phi}$.

α	$\alpha = 0$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 10$
ssd	4.8134	0.6377	1.6869	1.8762
maximal distance[1]	0.3444	0.0706	0.0757	0.0797
average distance	0.1109	0.0177	0.0198	0.0203
maximal angle difference[2]	25.8343	15.7230	16.5480	16.6378
average angle difference	2.6132	1.7481	1.8057	1.9603

Computational domain is $[1, 65] \times [1, 65]$, [1]distance means $\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}_0(\boldsymbol{x})\|_2, \boldsymbol{x} \in \Omega$, [2]angle difference means the corresponding angle differences in every corresponding quadrilateral.

Table 4.1. Comparison of Experiment 1

The second experiment uses a 2D transformation with significant curl vectors, which provides more convincing results. Also a noise is added. Figure 4.3,4.4 and table 4.4 show the results and comparison.



(a) $\alpha = 0$, curl not used

(b) $\alpha = 0$, enlarged view of left rectangle

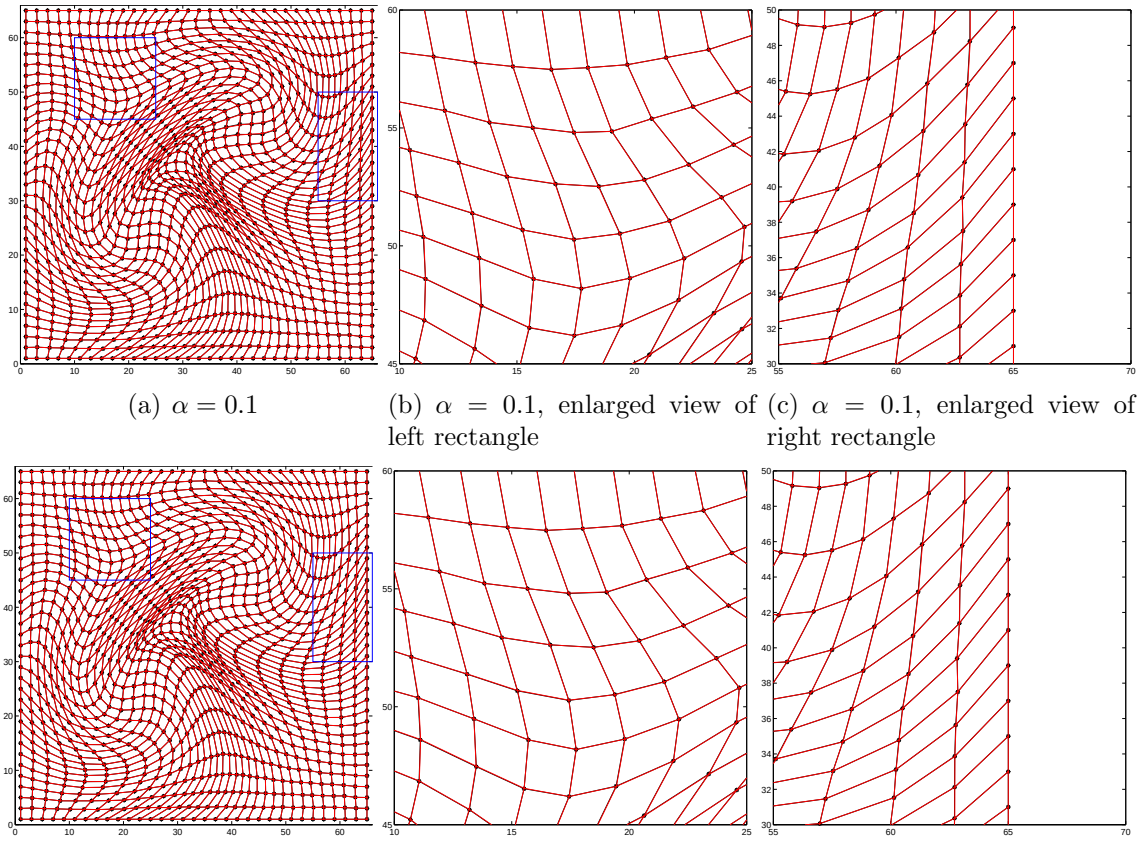
(c) $\alpha = 0$, enlarged view of right rectangle

(d) $\alpha = 1$

(e) $\alpha = 1$, enlarged view of left rectangle

(f) $\alpha = 1$, enlarged view of right rectangle

Figure 4.3. Experiment 2, 65×65 grid nodes. The black star dots $*$ represent $\boldsymbol{\phi}_0$, and red dots $\cdot$ represent constructed $\boldsymbol{\phi}$.



(a) $\alpha = 0.1$ (b) $\alpha = 0.1$, enlarged view of left rectangle (c) $\alpha = 0.1$, enlarged view of right rectangle

(d) $\alpha = 10$ (e) $\alpha = 10$, enlarged view of left rectangle (f) $\alpha = 10$, enlarged view of right rectangle

Figure 4.4. Experiment 2-continued, 65×65 grid nodes. The black star dots $*$ represent $\boldsymbol{\phi}_0$, and red dots $\cdot$ represent constructed $\boldsymbol{\phi}$.

α	$\alpha = 0$	$\alpha = 1$	$\alpha = 0.1$	$\alpha = 10$
ssd	2.9607e+003	0.4004	0.5845	0.4924
maximal distance[1]	13.9553	0.1311	0.1738	0.1434
average distance	4.6372	0.0129	0.0162	0.0141
maximal angle difference[2]	94.6738	13.5459	14.7015	13.5859
average angle difference	28.4427	1.0460	1.1987	1.1026

Computational domain is $[1, 65] \times [1, 65]$, [1]distance means $\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}_0(\boldsymbol{x})\|_2, \boldsymbol{x} \in \Omega$, [2]angle difference means the corresponding angle differences in every corresponding quadrilateral.

Table 4.2. Comparison of Experiment 1

Comparing the cases of $\alpha = 0$ and $\alpha \neq 0$, it is obvious that the curl vector plays an important role in transformations. With both the Jacobian determinant and curl vector, we can recover a transformation quite precisely. On one hand, the numerical results encourage us to think about the uniqueness problem in mathematical theory, which will be discussed next. On the other hand, these experiments show again *the variational method* is reliable and accurate in transformation construction. By the way, we can also observe that there is no significant difference among different values of α.

4.3 Theoretical analysis

First we describe the problem as: suppose there are 2 diffeomorphisms Φ and Ψ from Ω to Ω, such that

$$J(\Psi) = J(\Phi), \tag{4.1}$$

$$\text{curl}\Psi = \text{curl}\Phi, \tag{4.2}$$

$$\Psi = \Phi \quad on \quad \partial\Omega. \tag{4.3}$$

Is it true that $\Psi \equiv \Phi$ on Ω?

We start our study by the simplest case in 2D, where one of the diffeomorphism is the identity map.

4.3.1 One diffeomorphism is the identity map

Let Φ be the identity map, u is the difference between Φ and Ψ, bounded on Ω and zero on $\partial\Omega$. More specifically,

$$\Phi = identity, \tag{4.4}$$

$$\Psi - \Phi = u, \tag{4.5}$$

$$\|u\|_{H_o^2(\Omega)} = O(\epsilon), \tag{4.6}$$

$$u = 0 \quad on \quad \partial\Omega. \tag{4.7}$$

Then $\Phi(x,y) = (x,y)$, by (4.4), $J(\Phi) = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$.

$\Psi(x,y) = (x + u_1(x,y), y + u_2(x,y))$, so

$$J(\Psi) = \begin{vmatrix} 1 + u_{1x} & 0 + u_{2x} \\ 0 + u_{1y} & 1 + u_{2y} \end{vmatrix} = 1 + u_{1x} + u_{2y} + u_{1x}u_{2y} - u_{1y}u_{2x}$$

$$= 1 + u_{1x} + u_{2y} + J(u) = 1 + \text{div } u + J(u).$$

By (4.1), we must have div $u + J(u) = 0$, and by (4.2), curl $u = 0$, i.e.

$$\begin{cases} \text{div } u &= -J(u), \\ \text{curl } u &= 0. \end{cases} \tag{4.8}$$

Which leads to the following equations:

$$\begin{cases} \Delta u_1 &= (-J(u))_x &= O(\epsilon^2), \\ \Delta u_2 &= (-J(u))_y &= O(\epsilon^2). \end{cases} \tag{4.9}$$

For the first and second derivative of u is $O(\epsilon)$ (4.6), and $J(u) = u_{1x}u_{2y} - u_{1y}u_{2x}$. Note that $u = (u_1, u_2)$, $\Delta u = (\Delta u_1, \Delta u_2)$, we have

$$\|\Delta u\|_{L^2} = (\iint_\Omega \mid \Delta u \mid^2)^{\frac{1}{2}} = O(\epsilon^2). \tag{4.10}$$

By *Green's first identity* and (4.7), we have

$$\iint_\Omega \mid \nabla u \mid^2 = \mid \iint_\Omega u \cdot \Delta u \mid . \tag{4.11}$$

By *Cauchy-Schwarz inequality*, the RHS of (4.11):

$$\mid \iint_\Omega u \cdot \Delta u \mid \leq \iint_\Omega \mid u \cdot \Delta u \mid \leq \|u\|_{L^2} \|\Delta u\|_{L^2}.$$

By *Poincaré's inequality*, the LHS of (4.11):

$$\iint_\Omega \mid \nabla u \mid^2 \geq C \|u\|_{L^2}^2,$$

with C is a optimal Constant.

Combine those two,

$$C\|u\|_{L^2}^2 \leq \iint_\Omega \mid \nabla u \mid^2 = \mid \iint_\Omega u \cdot \Delta u \mid \leq \|u\|_{L^2} \|\Delta u\|_{L^2}. \tag{4.12}$$

First, we get

$$\|u\|_{L^2} \leq \|\Delta u\|_{L^2} \leq O(\epsilon^2).$$

Plug it back into (4.11), second we get $\|\nabla u\|_{L^2}^2 = \iint_\Omega \mid \nabla u \mid^2 \leq \|u\|_{L^2} O(\epsilon^2) \leq O(\epsilon^4)$.

Namely,

$$\begin{aligned} \|u\|_{L^2} &\leq O(\epsilon^2), \\ \|\nabla u\|_{L^2} &\leq O(\epsilon^2). \end{aligned} \tag{4.13}$$

Last, $\Delta u_1 = (-J(u))_x = -(u_{1x}u_{2y} - u_{1y}u_{2x})_x = -u_{1xx}u_{2y} - u_{1x}u_{2yx} + u_{1yx}u_{2x} + u_{1y}u_{2xx} \Rightarrow \mid \Delta u_1 \mid \leq O(\epsilon^4)$. Same with Δu_2.

We can apply the same procedure (4.11-4.13) again under the new condition $\|\Delta u\|_{L^2} \leq O(\epsilon^4)$, and achieve $\|u\|_{L^2} \leq O(\epsilon^4)$ at the end.

It is clear to see this iterative work will achieve results as $\|u\|_{L^2} \leq O(\epsilon^2), O(\epsilon^4), O(\epsilon^6)$....

Hence, $\|u\|_{L^2} = 0$, namely $u \equiv 0$, $\Psi = \Phi$.

4.3.2 Two diffeomorphisms close to the identity map

Next, we suppose Φ and Ψ both differs a small diffeomorphism with identity map, i.e.:

$$\Phi = identity + u \tag{4.14}$$

$$\Psi = identity + v \tag{4.15}$$

$$\|u\|_{H^2_\sigma(\Omega)} = O(\epsilon) \quad \|v\|_{H^2_\sigma(\Omega)} = O(\epsilon) \tag{4.16}$$

$$u = v = 0 \quad on \quad \partial\Omega. \tag{4.17}$$

By (4.1,4.2), we have

$$\begin{cases} J(\Phi) = 1 + \text{div } u + J(u) = J(\Psi) = 1 + \text{div } v + J(v), \\ \text{curl } u = \text{curl } v, \end{cases}$$

which equals

$$\begin{cases} \text{div } (u - v) &= -(J(u) - J(v)), \\ \text{curl } (u - v) &= 0. \end{cases} \tag{4.18}$$

Similarly,

$$\begin{cases} \Delta(u - v)_1 &= -(J(u) - J(v))_x &= O(\epsilon^2) \\ \Delta(u - v)_2 &= -(J(u) - J(v))_y &= O(\epsilon^2). \end{cases} \tag{4.19}$$

We can carry out the similar iterative procedure, achieve $\|u-v\|_{L^2} \leq O(\epsilon^2), O(\epsilon^4), O(\epsilon^6)$.... Hence, $\|u - v\|_{L^2} = 0$, namely $u \equiv v$, $\Psi = \Phi$.

4.3.3 Conclusion

Overall, we just get a very weak statement. If two diffeomorphisms are "very close" to the identity map, with fixed boundary condition, same Jacobian determinant and same curl vector imply they are the same map. It is quite far from a general conclusion. And we couldn't even figure out the uniqueness for the following simple

problem: If $J(\phi) = 1, \text{curl}\phi = 0, \phi \mid_{\partial\Omega} = identity$, is $\phi : \Omega \to \Omega$ the identity map? Despite this, we are still optimistic about the potential uniqueness.

CHAPTER 5

APPLICATION TO NON-RIGID IMAGE REGISTRATION

5.1 Image registration and transformations

A image is actually a scalar function $I(x, y)$ in 2D, $I(x, y, z)$ in 3D. So for 2 different images $I_1(x, y, z)$ and $I_2(x, y, z)$, Image registration process is to find a transformation Φ such that $I_1(\Phi) = I_2$ or $I_2(\Phi) = I_1$. Hence, our methods of construction of diffeomorphism can be potentially applied in this area.

5.2 Optimal control approach

The first attempt[16] is based on one version of *the deformation method*. Let f be a positive function such that $\iint_\Omega (f-1)dx = 0$. The following two steps will generate a transformation Φ with $J(\Phi) = f(\Phi)$.

- First, to generate a vector field $\boldsymbol{u}$ from the *div-curl system*:

$$\begin{cases} \text{div } \boldsymbol{u} & = f - 1, \\ \text{curl } \boldsymbol{u} & = \boldsymbol{g} \ \textit{on } \Omega, \\ \boldsymbol{u} & = 0 \ \textit{on } \partial\Omega. \end{cases} \tag{5.1}$$

- Second, to form a grid node velocity field for s in [0,1]:

$$\boldsymbol{v} = \boldsymbol{u}/(s + (1-s)f).$$

  Then find the transformation $\Phi(x) = \Phi(x, 1)$, where $\Phi(x, s)$ is determined from the differential equation

$$\partial\Phi(x, s)/\partial s = \boldsymbol{v}(\Phi, s).$$

The main idea of the optimal control approach in image registration is to find Φ by optimizing a similarity measure, subject to the above equations. Previously, we use f and $\boldsymbol{g}$ as the control functions. By adjusting f and $\boldsymbol{g}$, we use the above equations to form the largest possible search space which consists of all smooth, invertible transformations.

But, the normalization requirement of f (i.e. $\iint_\Omega (f-1)dx = 0$) and the positivity requirement $f > 0$ are not easy to maintain during the optimization process. Moreover, the calculation of variational gradient is not based on solid mathematical derivation.

5.3 New algorithm of nonrigid image registration

Next, a new algorithm based on variational method is developed[17]. Given the template image T and the reference image R in 2D, we determine a registration transformation Φ by iteratively minimizing

$$SSD(\Phi) = \frac{1}{2}\iint_\Omega (T(\Phi(x)) - R(x))^2 d\boldsymbol{x} \tag{5.2}$$

under constraints:

1. The registration transformation Φ is updated from the current transformation by solving one-step forward finite difference method. Namely:

$$\Phi(x) = \Phi_{old}(x) + \frac{\boldsymbol{u}(x)}{1 + div\ \boldsymbol{u}}\Delta t. \tag{5.3}$$

2. $\boldsymbol{u}$ satisfies,

$$\Delta \boldsymbol{u} = \boldsymbol{F} = (f_1, f_2), \boldsymbol{u} = 0\ on\ \partial\Omega. \tag{5.4}$$

3. Control functions f_1 and f_2 are the defined by

$$f_1 = f_{x_1} - g_{x_2}\ and\ f_2 = f_{x_2} + g_{x_1}, \tag{5.5}$$

where f, g are the same ones in (5.1).

We can derive that

$$\partial SSD/\partial \boldsymbol{F} = (\boldsymbol{a} + \nabla b)\Delta t.$$

Vector $\boldsymbol{a}$ is determined by solving the Poisson equations

$$\begin{aligned} &\Delta \boldsymbol{a} = \boldsymbol{w} \ on\ \Omega, \\ &\boldsymbol{a} = \mathbf{0} \ on\ \partial\Omega, \\ with \quad &\boldsymbol{w} = \tfrac{T(\Phi(x))-R(x)}{1+div\boldsymbol{u}} \nabla T(\Phi(x)), \end{aligned} \tag{5.6}$$

and function b is determined by solving the Poisson equation

$$\begin{aligned} &\Delta b = h \ on\ \Omega, \\ &b = 0 \ on\ \partial\Omega, \\ with \quad &h = \tfrac{T(\Phi(x))-R(x)}{(1+div\boldsymbol{u})^2} \nabla T \cdot \boldsymbol{u}. \end{aligned} \tag{5.7}$$

The derivation of $\partial SSD/\partial \boldsymbol{F}$ is briefly showed as follows.

Let $\mathbf{F} = (f_1, f_2)$, and $\delta\mathbf{F} = (\delta f_1, \delta f_2)$ be variations that vanish at the boundary. Then $\Delta\delta\mathbf{u} = \delta\mathbf{F} = (\delta f_1, \delta f_2)$ from (5.4). We have

$$\delta SSD = \iint_\Omega (T - R)\delta T d\boldsymbol{x} = \iint_\Omega (T - R)\nabla T \cdot \delta\boldsymbol{\Phi} d\boldsymbol{x}.$$

Since $\delta\boldsymbol{\Phi} = \delta(\frac{\boldsymbol{u}}{1+div\ \boldsymbol{u}})\Delta t = \frac{\delta\boldsymbol{u}}{1+div\ \boldsymbol{u}}\Delta t - \frac{\boldsymbol{u}(\boldsymbol{div}(\delta\boldsymbol{u}))}{(1+div\ \boldsymbol{u})^2}\Delta t$, we get

$$\delta SSD = \iint_\Omega \frac{(T - R)\nabla T \cdot \delta\boldsymbol{u}}{1 + div\ \boldsymbol{u}}\Delta t d\boldsymbol{x} - \iint_\Omega \frac{(T - R)(\nabla T \cdot \boldsymbol{u})}{(1 + div\ \boldsymbol{u})^2} div(\delta\boldsymbol{u})\ \Delta t d\boldsymbol{x}.$$

Let $\boldsymbol{a}$ and b be the solutions to the Poisson equations (5.6) and (5.7) respectively. Then,

$$\begin{aligned} \delta SSD &= \iint_\Omega \boldsymbol{w} \cdot \delta\boldsymbol{u}\Delta t d\boldsymbol{x} - \iint_\Omega h\ div(\delta\boldsymbol{u})\Delta t d\boldsymbol{x} \\ &= \iint_\Omega (\Delta\boldsymbol{a} \cdot \delta\boldsymbol{u} - \Delta b\ div(\delta\boldsymbol{u}))\Delta t d\boldsymbol{x} \\ &= \iint_\Omega (\Delta\boldsymbol{a} \cdot \delta\boldsymbol{u} - b\ \Delta(div(\delta\boldsymbol{u})))\Delta t d\boldsymbol{x} \\ &= \iint_\Omega (\Delta\boldsymbol{a} \cdot \delta\boldsymbol{u} - b\ div(\Delta\delta\boldsymbol{u}))\Delta t d\boldsymbol{x} \\ &= \iint_\Omega (\boldsymbol{a} \cdot \Delta\delta\boldsymbol{u} + \nabla b \cdot \Delta\delta\boldsymbol{u})\Delta t d\boldsymbol{x} \\ &= \iint_\Omega ((\boldsymbol{a} + \nabla b)\delta\boldsymbol{F})\Delta t d\boldsymbol{x}. \end{aligned} \tag{5.8}$$

Hence $\partial SSD/\partial\boldsymbol{F} = (\boldsymbol{a} + \nabla b)\Delta t$.

An optimization scheme by gradient descend method can be applied now.

5.4 Development of non-rigid image registration algorithm

5.4.1 Simplification

Similar to what we did in *the variational method*:version 2, we can change the constraints (5.3,5.4,5.5) to

$$\Phi(x) = x + \boldsymbol{u}(x), \tag{5.9}$$

and

$$\Delta\boldsymbol{u} = \boldsymbol{F} = (f_1, f_2), \boldsymbol{u} = 0\ on\ \partial\Omega, \tag{5.10}$$

where f_1 and f_2 are two independent control functions, no longer we need (5.5).

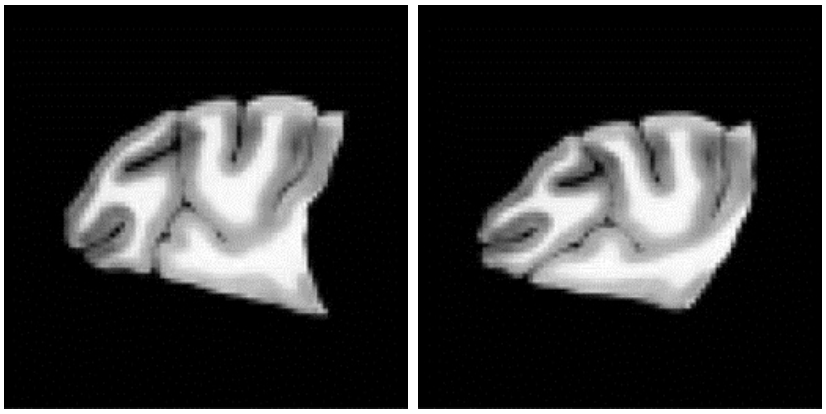
Now, we can derive that

$$\partial SSD/\partial\boldsymbol{F} = \boldsymbol{a},$$

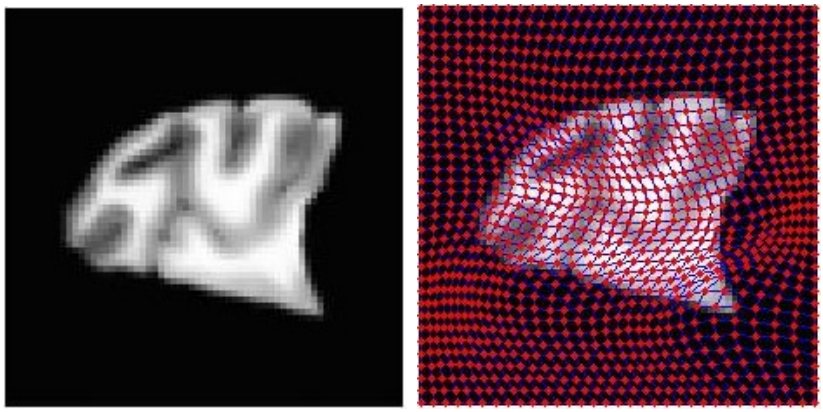
where vector $\boldsymbol{a}$ is determined by solving the Poisson equations

$$\begin{cases} \Delta\boldsymbol{a} &= (T(\Phi(x)) - R(x))\nabla T(\Phi(x))\ on\ \Omega, \\ \boldsymbol{a} &= \boldsymbol{0}\ on\ \partial\Omega. \end{cases} \tag{5.11}$$

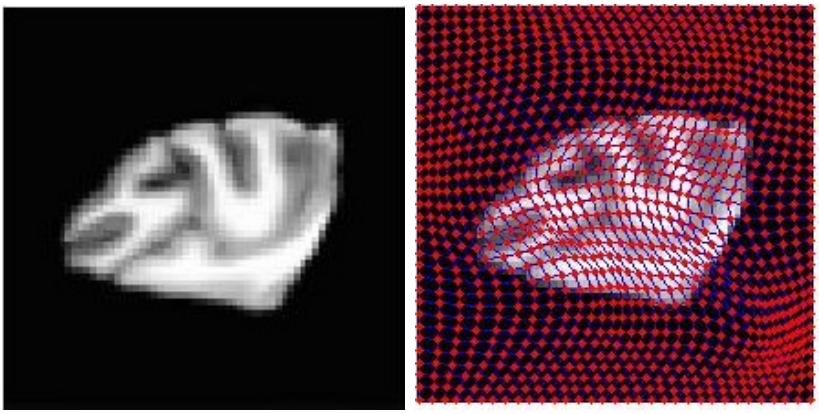
We shall just present some image registration results here to demonstrate the capacity of our algorithms.



(a) Reference Image R (b) Template Image T

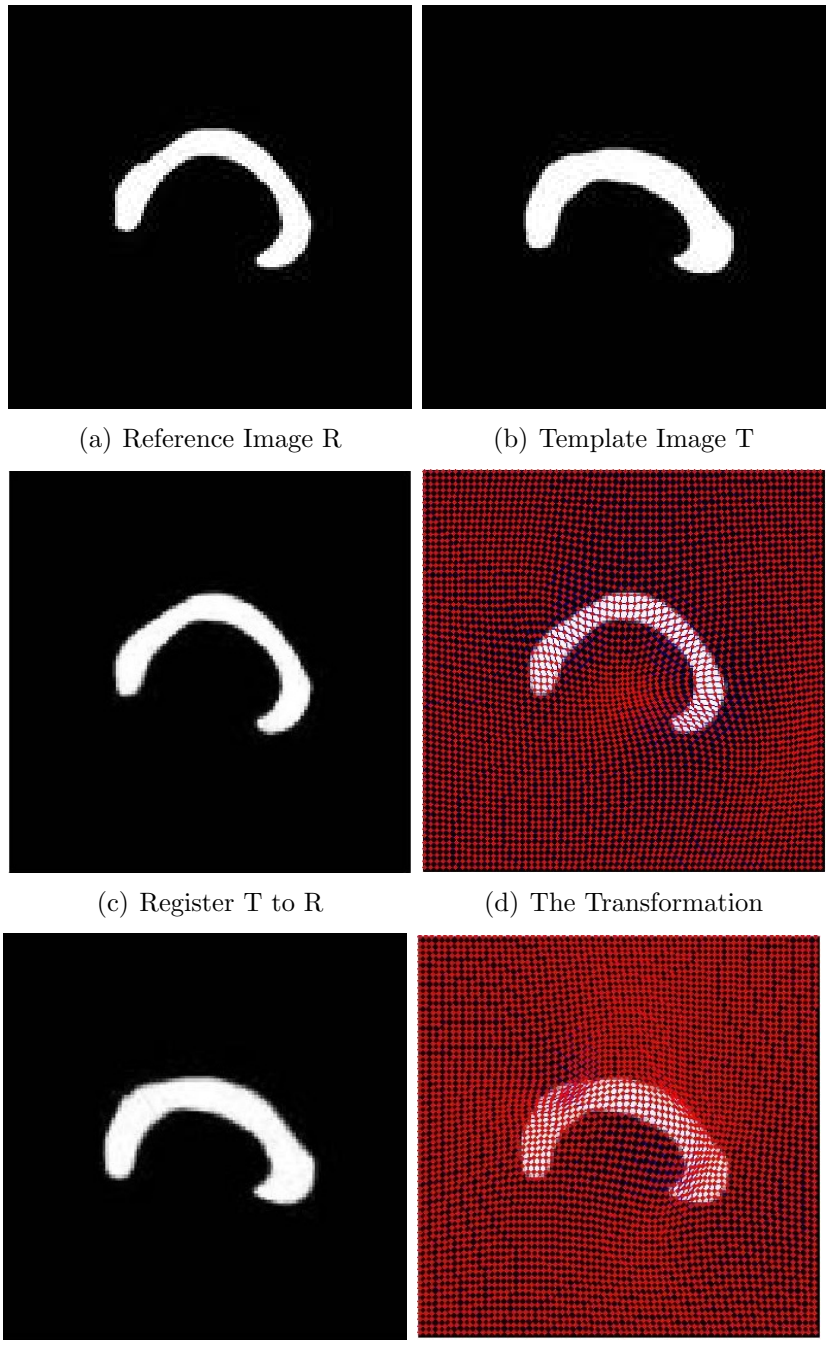


(c) Register T to R (d) The Transformation



(e) Register R to T (f) The Transformation

Figure 5.1. Image Registration Example 1.



(a) Reference Image R (b) Template Image T

(c) Register T to R (d) The Transformation

(e) Register R to T (f) The Transformation

Figure 5.2. Image Registration Example 2.

5.4.2 Symmetric scheme

Invertibility and Transitivity are two important properties of transformation. These two properties are not certainly implied in the algorithms above. In [26] and [27], a *Symmetric Scheme* is mentioned, so we tried to extend our algorithm in the same way:

Suppose there exists a implicit middle image M, and we register both the reference image R and the template image T simultaneously to this M. Namely to construct two transformations h_R and h_T, such that

$$SSD(h_T, h_R) = \frac{1}{2}\iint_{\Omega}(T(h_T(x)) - R(h_R(x)))^2 d\boldsymbol{x}$$

is minimized.

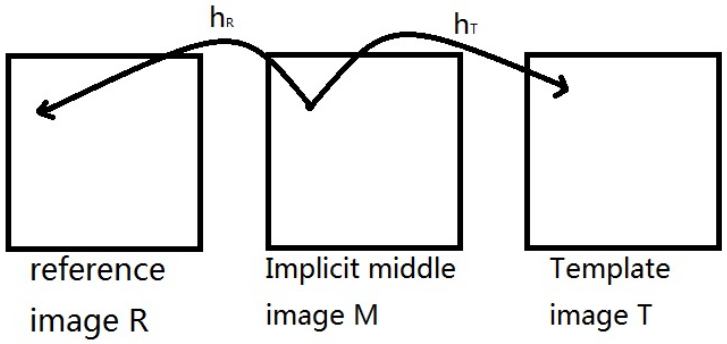


Figure 5.3. illustration of symmetric scheme.

Similarly, we shall derive the gradient $\frac{\partial SSD}{\partial \boldsymbol{f}}$ and $\frac{\partial SSD}{\partial \boldsymbol{g}}$ based on the following assumptions:

$$\begin{aligned} h_R(\boldsymbol{x}) &= \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}), \\ h_T(\boldsymbol{x}) &= \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x}). \end{aligned} \tag{5.12}$$

With the displacement $\boldsymbol{u}$ and $\boldsymbol{v}$ satisfy the Poisson's equations

$$\begin{cases} \Delta \boldsymbol{u} = \boldsymbol{f}, \\ \Delta \boldsymbol{v} = \boldsymbol{g}, \\ \boldsymbol{u}, \boldsymbol{v} \qquad = 0 \qquad on\ \partial\Omega. \end{cases} \tag{5.13}$$

Let $\boldsymbol{f} = (f_1, f_2), \boldsymbol{g} = (g_1, g_2)$, and $\delta\boldsymbol{f} = (\delta f_1, \delta f_2), \delta\boldsymbol{g} = (\delta g_1, \delta g_2)$ be variations that vanish at the boundary. And from (5.13), $\Delta\delta\boldsymbol{u} = \delta\boldsymbol{f}, \Delta\delta\boldsymbol{v} = \delta\boldsymbol{g}$.

By variational calculus,

$$\begin{aligned} \delta SSD &= \iint_\Omega (T(h_T(x)) - R(h_R(x)))(\delta T(h_T(x)) - \delta R(h_R(x)))d\boldsymbol{x} \\ &= \iint_\Omega (T(h_T(x)) - R(h_R(x)))(\nabla T \cdot \delta h_T(x) - \nabla R \cdot \delta h_R(x))d\boldsymbol{x}. \end{aligned}$$

Since $\delta h_T(x) = \delta\boldsymbol{v}(x), \delta h_R(x) = \delta\boldsymbol{u}(x)$, we get

$$\delta SSD = \iint_\Omega (T(h_T(x)) - R(h_R(x)))(\nabla T \cdot \delta\boldsymbol{v} - \nabla R \cdot \delta\boldsymbol{u})d\boldsymbol{x}.$$

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be the solutions to the Poisson equations:

$$\begin{cases} \Delta\boldsymbol{b} &= (T(h_T(x)) - R(h_R(x)))\nabla T, \\ \Delta\boldsymbol{a} &= -(T(h_T(x)) - R(h_R(x)))\nabla R, \\ \boldsymbol{a}, \boldsymbol{b} &= 0 \qquad on\ \partial\Omega. \end{cases} \tag{5.14}$$

Now,

$$\begin{aligned} \delta SSD &= \iint_\Omega (\Delta\boldsymbol{a} \cdot \delta\boldsymbol{u} + \Delta\boldsymbol{b} \cdot \delta\boldsymbol{v})d\boldsymbol{x} \\ &= \iint_\Omega (\Delta a_1\delta u_1 + \Delta a_2\delta u_2 + \Delta b_1\delta v_1 + \Delta b_2\delta v_2)d\boldsymbol{x} \\ &= \iint_\Omega (a_1\Delta\delta u_1 + a_2\Delta\delta u_2 + b_1\Delta\delta v_1 + b_2\Delta\delta v_2)d\boldsymbol{x} \\ &= \iint_\Omega (\boldsymbol{a} \cdot \delta\Delta\boldsymbol{u} + \boldsymbol{b} \cdot \delta\Delta\boldsymbol{v})d\boldsymbol{x} \\ &= \iint_\Omega (\boldsymbol{a} \cdot \delta\boldsymbol{f} + \boldsymbol{b} \cdot \delta\boldsymbol{g})d\boldsymbol{x}. \end{aligned}$$

Finally, we obtain

$$\begin{aligned} \frac{\partial SSD}{\partial \boldsymbol{f}} &= \boldsymbol{a} \\ \frac{\partial SSD}{\partial \boldsymbol{g}} &= \boldsymbol{b}. \end{aligned} \tag{5.15}$$

After finding h_R, h_T, $h_T \circ h_R^{-1}$ is the transformation from R to T, $h_R \circ h_T^{-1}$ is the transformation from T to R, and they are invertible and transitive.

5.5 Conclusion

Overall, Image registration is a complicated task, it requires many other works such as pre-processing, post-processing, segmentation etc. There are three components provide a common classification schema for registration methods: *the transformation model, the similarity(or correspondence) measures, and the optimization strategy*. Many methods with different characteristic features are invented in this area. We want to point out that the major features of our non-rigid image registration method are: it bases on a solid but simple mathematical foundation, and it can produce a smooth transformation between images.

5.6 Graphical User Interface:

We shall end this chapter with another graphical user interface we designed for image registration: "Image Registration Tools". It allows anyone to input two images as template image and reference image, then implement our registration algorithm, and provide the corresponding transformation. The next two figures illustrate works done by "Image Registration Tools".

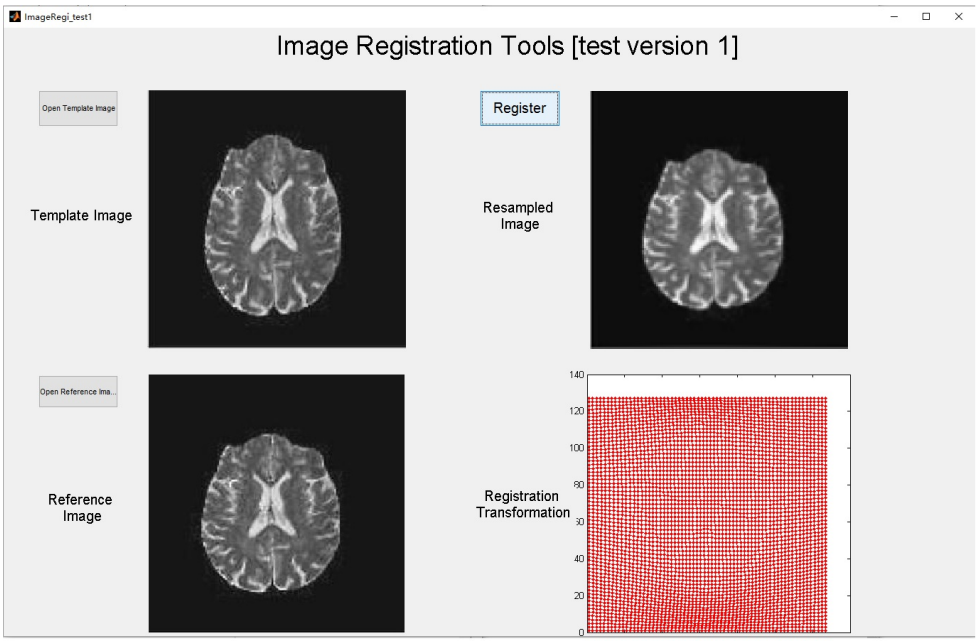


Figure 5.4. Image Registration Tools example 1.

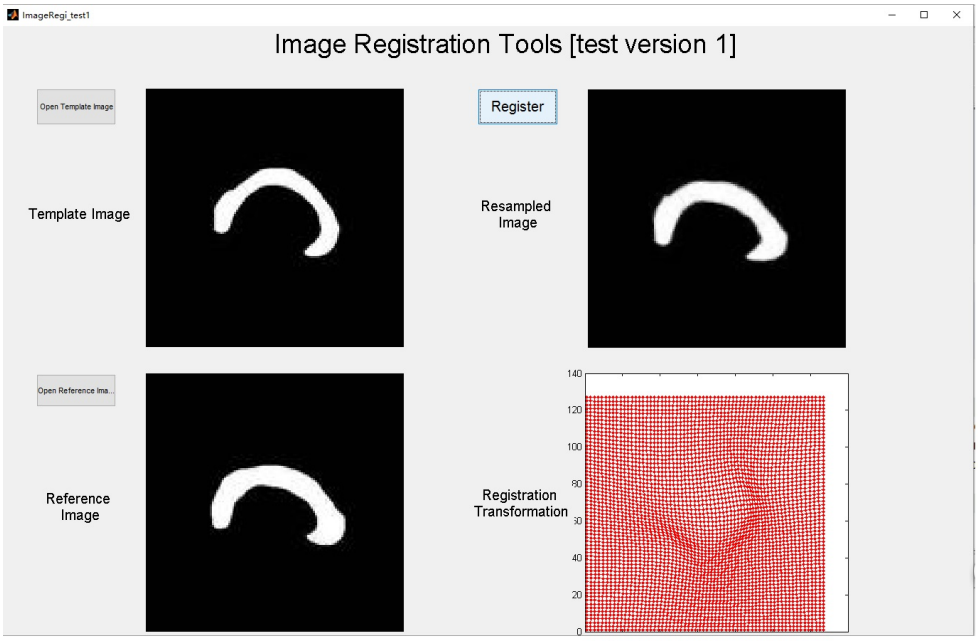


Figure 5.5. Image Registration Tools example 2.

REFERENCES

[1] B. Dacorogna and J. Moser, "On a partial differetial equation involving the jacobian determinant," *Ann. Inst H Poincare*, vol. 7, no. 1, pp. 1–26, 1990.

[2] G. Liao and D. Anderson, "A new approach to grid generation," *Applicable Analysis: An International Journal*, vol. 44, no. 3-4, pp. 285–298, 1992.

[3] G. Liao and J. Su, "Grid generation via deformation," *Applied Mathematics Letters*, vol. 5, no. 3, pp. 27–29, 1992.

[4] G. Liao and J. Su, "A direct method in dacorogna-moser's approach of grid generation problems," *Applicable Analysis: An International Journal*, vol. 49, no. 1-2, pp. 73–84, 1993.

[5] G. Liao and H.Liu, "Existence and $c_{0,\alpha}$ regularity of minima of a functional related to the grid-generation problem," *Numerical Methods for Partial Differential Equations*, vol. 9, no. 3, pp. 261–264, 1993.

[6] F. Liu, S. Ji, and G. Liao, "An adaptive grid method and its application to steady euler flow calculations," *SIAM J. Sci. Comput*, vol. 20, no. 3, pp. 811–825, 1998.

[7] G. Liao, F. Liu, G. C. de la Pena, D. Peng, and S. Osher, "Level-set-based deformation methods for adaptive grids," *Journal of Computational Physics*, vol. 159, pp. 103–122, 2000.

[8] D. L. Fleitas, *Dissertation: The least-squares finite element method for grid deformation and meshfree applications*. 2005.

[9] P. Bochev, G. Liao, and G. C. de la Pena, "Analysis and computation of adaptive moving grids by deformation," *Numerical Methods for Partial Differential Equations*, vol. 12, no. 4, pp. 489–506, 1996.

[10] G. Liao, J. Su, Z. Lei, G. G. de la Pena, and D. Anderson, "A moving finite difference method for partial differential equations," *Studia Universitatis Babeş-Bolyai Mathematica*, vol. 69, no. 2, 2004.

[11] G. Liao and B. Semper, "A moving grid finite-element method using grid deformation," *Numerical Methods for Partial Differential Equations*, vol. 11, no. 6, pp. 603–615, 1995.

[12] G. Liao, T.-W. Pan, and J. Su, "Numerical grid generator based on moser's deformation method," *Numerical Methods for Partial Differential Equations*, vol. 10, no. 1, pp. 21–31, 1994.

[13] G. Liao and J. Su, "A moving grid method for (1 + 1) dimension," *Applied Mathematics Letters*, vol. 8, no. 4, pp. 47–49, 1995.

[14] X. xin Cai, D. Fleitas, B. Jiang, and G. Liao, "Adaptive grid generation based on the least-squares finite-element method," *Computers and Mathematics with Applications*, vol. 48, no. 7-8, pp. 1077–1085, 2004.

[15] X. Chen and G. Liao, "New variational method of grid generation with prescribed jacobian determinant and prescribed curl," *arxiv.org/pdf/1507.03715*, 2015.

[16] G. G. Liao, X. Cai, D. Fleitas, X. Luo, J. Wang, and J. Xue, "Optimal control approach to data set alignment," *Applied Mathematics Letters*, vol. 21, no. 9, pp. 898–905, 2008.

[17] H.-Y. Hsiao, C.-Y. Hsieh, X. Chen, Y. Gong, X. Luo, and G. Liao, "New development of nonrigid registration," *The ANZIAM Journal*, vol. 55, no. 03, pp. 289–297, 2014.

[18] T. C. Sideris, *Ordinary Differential Equations and Dynamical Systems, Class Notes 2009-2010, Department of Mathematics, UC santa barbara.*

[19] B. Jiang, *The Least-Squares Finite Element Method: Theory and Applications in Computational Fluid Dynamics and Electromagnetics.*

[20] E. A. Coddington and R. Carlson, *Linear Ordinary Differential Equations*.

[21] M. Grajewski, M. Köster, and S. Turek, "A new multilevel grid deformation method," 2008.

[22] D. Wan and S. Turek, "Direct numerical simulation of particulate flow via multigrid fem techniques and the fictitious boundary method," *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS*, vol. 51, p. 531C566, 2006.

[23] G. G. Liao, X. Cai, J. Liu, X. Luo, J. Wang, and J. Xue, "Construction of differentiable transformations," *Applied Mathematics Letters*, vol. 22, no. 10, pp. 1543–1548, 2009.

[24] G. H. Meisters and C. Olech, "Locally one-to-one mappings and a classical theorem on schlicht functions," *Duke Mathematical Journal*, vol. 30, no. 1, pp. 63–80, 1963.

[25] M. Gurtin, "on the nonlinear theory of elasticity," *Contemporary Developments in Continuum Mechanics and Partial Differential Equations: Proceedings of the International Symposium on Continuum Mechanics and Partial Differential Equations, Rio de Janeiro, August 1977*.

[26] B. B. Avants, M. Grossman, and J. C. Gee, *Symmetric diffeomorphic image registration: Evaluating automated labeling of elderly and neurodegenerative cortex and frontal lobe*. 2006.

[27] X. Geng, H. Gu, W. Shin, T. Ross, and Y. Yang, "Unbiased group-wise image registration: Applications in brain fiber tract atlas construction and functional connectivity analysis," *Journal of Medical Systems*, vol. 35, no. 5.

BIOGRAPHICAL STATEMENT

Xi Chen was born in Hefei, Anhui, China, in 1990. He received his B.S. degree from the special class for the gifted young, University of Science and Technology of China, in 2010, his Ph.D. degree from The University of Texas at Arlington in 2016, all in Mathematics and Applied Mathematics. During his 5 years doctoral studies, he also served as a Graduate Teaching Assistant in the department of mathematics, taught multiple undergraduate courses as instructor. His research interest is in the areas of numerical grid generation and image registration. He received Outstanding Graduate Research Award in 2016.