Numerical Methods for Partial Differential Equations from Geometry Related Problems

by

Hao Liu

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Mathematics

August 31, 2024, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Hao Liu

Numerical Methods for Partial Differential Equations from Geometry Related Problems

by

Hao Liu

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Shingyu Leung

Prof. J.R. Chasnov , Acting Head of Department

Department of Mathematics

July 2018

To my parents

Acknowledgment

I would like to express my deep gratitude to my supervisor, Prof. Shingyu Leung. During my four year PhD study in HKUST, he gave me lots of valuable advices and opportunities. I have deep impression on his enthusiasm in research, teaching and mentoring students. In our meetings, he always had many ideas that I had not considered before. I benefit a lot from him. I feel so fortunate having him as my supervisor. I want to show my great appreciation to Prof. Roland Glowinski. I learned a lot from him. His wealth of ideas, not satisfied of current results and rigorous academic attitude affects me a lot. It is my great pleasure to work with and learn from him.

I am grateful to the committee members in my PhD qualify exam and my PhD thesis: Prof. Yang Xiang, Prof. Xiaoping Wang, Prof. Jianfeng Cai, Prof. Albert C. S. Chung, Prof. Ronald Lok Ming Lui, and Prof. Tiezheng Qian.

I want to show great thanks to Prof. Zhigang Yao and Prof. Tony F. Chan for the discussion on our level set method for dimension reduction. I appreciate Prof. Tony F. Chan in his meetings where I am impressed with his interests and ideas on problems in various areas. I also want to thank Prof. Jianliang Qian for his suggestions on our projects on the Monge-Ampère equation.

I want to shown my deep gratitude to Prof. Tieyong Zeng for his continuous encouragements and helps. He is one of my first teachers in research. I also want to thank Prof. Tao Tang, Prof. Rolf Jeltsch, Prof. Yutian Li and Prof. Xiaonan Wu for their generous help in my study and research in my undergraduate time.

There are so many friends who helped me and enriched my life in the past several years. I want to thank Prof. Shingyu Leung's group: Meng Wang, Boxi Xu, Ningchen Ying, Kawah Wong, Wingfai Kwan, Yukeung Ng, Kwunlun Chu, Haiyi Tan and Siyang Wei for their presentations, discussions and their helps in

my research. I want to thank my friends in HKUST: Haohan Li, Xiaoyu Wei, Dong Wang, Haoyun Tang, Shidong Cui, Rui She, Ting Li, Xianshi Yu, Ke Wei, Guiyu Cao, Xing Ji, Yuqi Zhao, Diwei Li and Yisu Lo. Because of them, my time in HKUST is colourful and rich.

I would like to thank my friends in HKBU: Yugen Li, Xiangpeng Meng, Ronggang Li, Yuchen Kang, Yuchen Li, Jianchen Zhu, Zecheng Zhang, Xu Han, Fan Zhang, Ran Tao, Wenye Xie, Fenruo Wang and everyone in Dreama. I spent my first four years in Hong Kong with them and had many wonderful memories. I would like to thank my friends in my hometown: Guangcan Gao, Bing Li, Bing Yu, Bin Yu, Fangfang Hou, Qiannan Ye, Yanxin Fei, Chaozheng Liu, Xiaodong Zhao, Rui Zhou, Muchuan Shen, Shichao Wang, Jinfeng Duan and Shao Li for their support and accompany in these years. I am grateful to everyone in G9 and WHCC. They gave me lots of help and care in Hong Kong and Houston. I want to thank uncle and aunty in my host family in Hong Kong for their support and help.

I would like to thank the Department of Mathematics for providing me this opportunity with postgraduate studentship award so I have the valuable opportunity to study here.

I thank my family especially my parents for their constant encouragement and support.

Contents

List of Figures

List of Tables

Numerical Methods for Partial Differential Equations from Geometry Related Problems

Hao Liu

Department of Mathematics

Abstract

In this dissertation, we apply the level set method and the operator-splitting method to solve geometry related problems. In the first part we use the level set method for dimension reduction on a Riemannian manifold. We design an energy based on the distance from the solution to the data set and the principal direction generated from the data set. The functional is minimized by gradient decent. This algorithm is implemented using level set functions where the solution is represented implicitly. So we do not need any a priori knowledge of the structure of the solution. Our algorithm is very robust to noise and outliers. In the second part we apply operator-splitting method to solve some Monge-Ampère equation related problems. The Monge-Ampère equation originates from the Minkowski problem in differential geometry which asks to reconstruct a convex surface from a prescribed Gaussian curvature. In these problem, the solution is constructed by solving the Monge-Ampère equation. Operator-splitting method for two dimensional Monge-Ampère equation with the Dirichlet problem is first discussed. We rewrite the equation in an elliptic divergence form and then solve an initial value problem to steady state. For problem with classical solution, a two-stage strategy is introduced to accelerate the convergence rate. Then the algorithm is modified to solve various related problems.

Chapter 1

Introduction

Many geometry related problems are prescribed by partial differential equations (PDE's). Therefore it is desired to develop efficient and accurate numerical methods to obtain an approximated solutions. In this thesis, we explore the level set method [94] and some operator-splitting methods [47] for these geometric problems. Level set method was originally proposed to model the evolution of interface. The formulation is implicit and can naturally handle topology changes. It has been applied in various areas [94]. Operator-splitting method [47] is another widely used numerical method. For complicated problems, operator-splitting method decompose the problem into several relatively simple problems that can be solved easily. Operator-splitting method has induced many famous numerical methods: alternating implicit method (ADI), alternating explicit method (ADE), alternating direction methods of multipliers (ADMM) and etc. In this dissertation, we apply level set method to solve dimension reduction problem and apply operator-splitting method to solve Monge-Ampère related problems.

In Chapter 2, we use the level set method for dimension reduction on Riemannian manifolds. Given some data set on a manifold, we want to fit a surface or a curve to the data. This goal is achieved using the level set method. With our method,

no a priori knowledge of the desired curve is needed. The solution is represented implicitly. For the solution containing several closed patterns, we only need to initialize a large curve enclosing all data set. It can then automatically split into multiple patterns. Examples with real earthquake location data are tested.

In Chapter 3, we apply the operator-splitting method to solve the canonical two dimensional Monge-Ampère equation with the Dirichlet boundary condition. We take advantage of the equivalence between the Mogne-Ampère equation and a divergence form to get an initial value problem. The equation is then time discretized by the opetrator-splitting method and space discretized by a mixed finite element method. If classical solution exists, we introduce a two-stage strategy to accelerate our algorithm.

Chapter 4 to Chapter 8 are extensions of the algorithm in Chapter 3. In these chapters, we use operator-splitting method to solve the Minkowski problem, the obstacle Monge-Ampère equation, the degenerate Monge-Ampère equation, the Monge-Ampère equation with the Neumann boundary condition and the three dimensional Monge-Ampère equation. In Chapter 4, we discuss the operator-splitting method for the Minkowksi problem, the one from differential geometry that induces the Monge-Ampère equation. In our algorithm, the nonlinear term on the right hand side are taken explicitly by finite element method. In Chapter 5, the operator-splitting method for the obstacle problem of the Monge-Ampère equation is designed. Our algorithm takes advantage of the subdifferential representation of the characteristic function. Two different operator-splitting schemes are tested. In Chapter 6, we talk about an algorithm for a class of degenerate Monge-Ampère equation. We consider an obstacle problem obtained as the limit of a Mong-Ampère problemin Chapter 5. In Chapter 7, an algorithm for the Monge-Ampère equation with the Nuemann boundary condition is derived. Due to the divergence formulation of the Monge-Ampère operator, it is not trivial to incorporate the Neumann boundary condition in our algorithm. A prey-predator

strategy is used. Our algorithms have been tested on various examples including the moving mesh problem. In Chapter 8, based on the three dimensional divergence formula, we extend our algorithm to solve three dimensional Monge-Ampère equation.

Chapter 2

Level Set Method for Dimension Reduction on Riemannian Manifold

2.1 Introduction

Principal component analysis (PCA) is a linear technique for reducing the dimension of a set of high-dimensional data in the standard Euclidean space. The idea is to determine the linear projection to the data that maximizes the variance in the projected space. In many applications, however, we have data constrained on a low-dimensional manifold embedded in some high-dimensional space. One simple example is when the collected data are landmarks identifying the shape of certain objects. To better understand the shape deformation, one needs to map the data to the intrinsic manifold space rather than directly works with the measurements in the vector space.

In the past decade or so, there has been a vast variety of nonlinear approaches developed to better handle nonlinear complex data by primarily extending the standard PCA to the manifold scenario. Strictly speaking, the majority of these approaches have been focused on extending the PCA to the manifold cases in which the embedding is assumed to be known. Given such a known embedding or manifold, the principal geodesic analysis (PGA) [37, 55] defines the first principal component on the manifold to be the geodesic passing through the mean of the given data and with the minimum average distance to the sample points. In this sense, the PGA essentially replaces the line of the PCA in the Euclidean space with a geodesic on manifolds. Another method is the so-called tangent space PCA [51, 36] which first projects all the sample data onto the tangent space at the mean, and then interprets the first principal component as the direction of maximal variability on the tangent space. Finally, the method maps this component back to the manifold. Not restricted to a single geodesic or curve, the method introduced in [37] decomposes the data variation by finding a sequence of nested linear sub-manifolds locally around the mean. A special case of using a collection of nested spheres can be found in [57]. Principal curves/surfaces, introduced in [53] as a dimensionality reduction tool in Euclidean space, are self-consistent smooth curves passing through the middle of the data. By extending [53], the authors in [54] have recently showed that the principal geodesic is a classic principal curve on Riemannian manifolds. Some other approaches more or less aim to directly fit a curve to the given sampling data on the manifold [62, 60, 110]. We refer interested readers to [74, 65, 49, 120] for a more complete discussion and a review on the application to dimensionality reduction.

In a recent work, a principal flow [97] approach has been proposed to determine a curve on the manifold passing through the mean of the data such that the tangent vector along the curve fits locally the first eigenvector of a local tangent space PCA. An extension to higher dimensions has been recently proposed

in [122] by the concept of principal sub-manifold. This paper aims to incorporate the level set method [96, 105, 95] with the principal flow to determine a codimension-one surface Γ on the manifold that *best fits* the dataset. Instead of dealing with an explicit parametrization of a curve or a surface, the level set method was originally developed to simulate evolution of the interface by solving equations on a fixed Cartesian mesh. Because of the flexibility in modelling the topological change in the interface evolution, it has become a very popular tool in fields including multiphase flow simulation, shape optimization, computational geometry, computer graphics and etc [105, 95]. The resulting method proposed in this work will, therefore, be a fully implicit and Eulerian approach which does not require any explicit parametrization of the underlying Riemannian manifold nor any assumption on the structure of the low dimensional representation of the given data.

Mathematically, we consider a complete Riemannian manifold $\mathcal{M}$ of dimension r, embedded in $\mathbb{R}^m$ such that $\mathcal{M}$ is implicitly represented by $\{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = 0\}$ for a given function $\psi : \mathbb{R}^m \to \mathbb{R}^{m-r}$. Let $\mathcal{S} = \{\mathbf{x}_i \in \mathcal{M}, 1 \leq i \leq N\}$ be a configuration of N points on $\mathcal{M}$ such that there exists a connected open set $\mathcal{B} \subset \mathcal{M}$ covering $\mathcal{S}$. In this paper, even though we concentrate only on the case where $r = 2$ and $m = 3$, i.e. point data sampling a curve on a two dimensional manifold in the three dimensional space, the approach can be generalized to higher dimensional spaces in a rather straightforward way.

When $\mathcal{M}$ is simply the standard $\mathbb{R}^m$, the dimensional reduction problem becomes a problem of shape reconstruction if one assumes that the data points sample a codimension-one manifold. This is an important question in fields such as computer graphics or medical imaging [126, 125, 95]. The most widely used approach is to determine a triangulated representation of the surface based on techniques like the Voronoi diagrams or the Delaunay triangulation [1, 32]. However, when there is noise in the given data or when the topology of the underlying

manifold is not known *a priori*, such approach might not give a robust reconstruction. Another possibility is a variational approach developed in [126, 125] which incorporates the level set method [96, 105, 95]. Such approach can give an implicit representation of the codimension-one surface by an isosurface of a level set function. The method requires only an unsigned distance function to the data points. It has been shown to be robust to the noise in the given data and can also capture various topologies in the underlying surface. When the distance function in the method is replaced by the so-called edge detector, the method relates closely to the snake model and the geodesic active contour in the field of image segmentation [59, 18]. Based on this relationship and some recent convexification strategies [67, 19, 13], a convex functional for point cloud reconstruction has recently been proposed in [73] which can automatically lead to the global minimizer of the corresponding variational functional.

All these methods, however, have been developed solely for reconstructing data in the standard Euclidean space without specifying any constraint in the data. In this paper, we are going to follow a similar strategy but are proposing a variational approach for reconstructing data with an extra constraint which states that the given data lies on a submanifold in a Riemannian space. When $r = 2$ and $m = 3$, our method indeed shares some similarities with the method introduced in [8, 14, 23] for modeling evolution of curves in the three dimensional space. But we in this work focus mainly on the construction of the velocity field on the Riemannian manifold which drives the curve to the data.

2.2 Background

Our proposed approach is developed based on two recent ideas. The first one is a variational approach developed in [126] which aims to reconstruct a two-dimensional surface from point cloud data in $\mathbb{R}^3$ using the level set method [96].

The other one is the principal flow proposed in [97]. In this section, we summarize and review these two algorithms for data processing.

2.2.1 A variational method for Euclidean point cloud reconstruction using the level set method

Given an unconnected point cloud in an Euclidean space, the aim here is to reconstruct the underlying codimensional-one surface. The problem is very ill-conditioned in the sense that the solution might usually be non-unique. Some reconstruction algorithms might be too sensitive to the noise in the given data. To develop a robust algorithm for data in the Euclidean space, [126] has proposed a variational method based on the level set formulation. The main idea is to determine a surface Γ which minimizes the energy

$$E(\Gamma) = \left[\int_{\Gamma} d^p(\mathbf{x})\, ds \right]^{\frac{1}{p}}$$

where $d(\mathbf{x})$ is the distance function from $\mathbf{x}$ to data set given by

$$d(\mathbf{x}) = \min_{\mathbf{x}_i \in \mathcal{S}} \|\mathbf{x} - \mathbf{x}_i\|_2 \,.$$

A straight-forward way for the minimization process is to parameterize the curve or the surface and then to optimize the discretized problem using the gradient descent. Without any *a priori* knowledge on the topology of the curve, however, it might be tricky to numerically implement such approach. The main challenge is that one will need to re-mesh the triangulation by reconnecting the sampling points of the surface if the topology of the interface does change in the evolution. To handle such change in the topology, [126] has reformulated the problem using the level set formulation. Let $\phi(\mathbf{x})$ be a level set function defined in $\mathbb{R}^m$ such that Γ is represented by zero level set of ϕ, i.e. $\Gamma = \phi^{-1}(0)$ and

$$\begin{aligned} \phi(\mathbf{x}) &> 0 \quad \text{outside} \quad \Gamma \\ \phi(\mathbf{x}) &< 0 \quad \text{inside} \quad \Gamma\,. \end{aligned}$$

Now, the energy can then be expressed in term of the level set function ϕ,

$$E(\phi) = \left[\int_{\mathbb{R}^m} d^p(\mathbf{x}) \delta(\phi)\, d\mathbf{x} \right]^{\frac{1}{p}} ,$$

where $\delta(x)$ is the Dirac delta function. Finally the corresponding Euler-Lagrange equation from the optimality condition is given by

$$\frac{\partial \phi}{\partial t} = \frac{1}{p}\, \delta(\phi) \, \left[\int d^p(\mathbf{y}) \delta(\phi) \|\nabla \phi\| d\mathbf{y} \right]^{\frac{1}{p}-1} \nabla \cdot \left[d^p(\mathbf{x}) \frac{\nabla \phi}{\|\nabla \phi\|} \right] .$$

2.2.2 Principal flow

The principal flow extends the notion of the PCA to Riemannian manifolds [97]. On one hand, the flow on the manifold has the advantage of being principal in the sense that it follows the main direction of the data points locally. On the other hand, the method is able to accommodate the curve fitting characteristic on the manifold. Given a set of points $\mathcal{S} = \{\mathbf{x}_i \in \mathcal{M}, 1 \leq i \leq N\}$, a principal flow is parameterized as a one dimensional curve, Γ, over a class of flows parameterized as

$$\begin{aligned} \Gamma(\mathbf{x}, v) = \quad & \Big\{ \eta : [0, r] \to \mathcal{M}, \eta \in C^2(\mathcal{M}), \eta(s) \neq \eta(s') \text{ for } s \neq s', \\ & \eta(0) = x, \dot{\eta}(0) = v, l(\eta[0, t]) = t \text{ for all } 0 \leq t \leq r \leq 1 \Big\}, \quad (2.1) \end{aligned}$$

where $\mathbf{x}$ is a chosen starting point (e.g., the Fréchet mean $\bar{\mathbf{x}}$) and v is the initial tangent unit at $\mathbf{x}$. This can be seen as a type of flows passing through the center of the data cloud, and at any point on the flow it points to the main direction of data variation.

Technically, the principal flow incorporates two ingredients: a local covariance matrix and a vector field. In a neighborhood of the point $\mathbf{x}$ at the scale h, a local tangent covariance matrix is defined as

$$\Sigma_h(\mathbf{x}) = \frac{1}{\sum_i \kappa_h(\mathbf{x}_i, \mathbf{x})} \sum_i \mathbf{log}_{\mathbf{x}}(\mathbf{x}_i) \otimes \mathbf{log}_{\mathbf{x}}(\mathbf{x}_i) \kappa_h(\mathbf{x}_i, \mathbf{x})$$

where $\mathbf{y} \otimes \mathbf{y} := \mathbf{y}\mathbf{y}^T$, $\kappa_h(\mathbf{x}_i, \mathbf{x}) = K(h^{-1} \|\mathbf{log}_{\mathbf{x}}\mathbf{x}_i - \mathbf{x}\|)$ with a smooth non-increasing univariate kernel K on $[0, \infty)$.

Proposition 2.2.1. *(Differentiability of the vector field) The eigenvector $e(\mathbf{x}_0)$ (or eigenvalue $\lambda(\mathbf{x}_0)$) of the local covariance matrix $\Sigma_h(\mathbf{x}_0)$ can be extended to a vector field defined on the open neighborhood $\mathbf{x}_0(h)$ of $\mathbf{x}_0$ such that for any $\mathbf{x} \in \mathbf{x}_0(h)$,*

- *$W : \mathbf{x}_0(h) \to \mathbb{R}^m$ is a differentiable function,*
- *$\Sigma_h(\mathbf{x})W(\mathbf{x}) = \lambda(\mathbf{x})W(\mathbf{x})$.*

The recent work [97] determines the principal flow over the candidate flow set (2.1) by solving the following two variational problems

$$\begin{aligned} \eta_1 &= \arg\sup_{\eta \in \Gamma(\bar{\mathbf{x}}, v_1)} \int_0^{l(\eta)} \langle \dot{\eta}(t), W(\eta(t)) \rangle \, dt \\ \eta_2 &= \arg\inf_{\eta \in \Gamma(\bar{\mathbf{x}}, v_2)} \int_0^{l(\eta)} \langle \dot{\eta}(t), W(\eta(t)) \rangle \, dt \end{aligned}$$

where η_1 and η_2 are the two opposite curves corresponding to the solution of the variational problem, respectively; $v_1 = e_1(\bar{\mathbf{x}})$, $v_2 = -v_1$, and $l(\eta)$ is the length of the curve η. W defined in the Proposition 2.2.1 is the extension of the eigenvector $e_1(\bar{\mathbf{x}})$ to the neighborhood $\bar{\mathbf{x}}(h)$ of $\bar{\mathbf{x}}$. To obtain either η_1 or η_2, [97] has showed that it is necessary to find the critical point of the Lagrangian

variational problem of

$$\mathcal{L}_1(W,\eta) = \int_0^{l(\eta)} \langle \dot{\eta}(t), W(\eta(t)) \rangle \, dt \tag{2.2}$$

subject to boundary conditions under certain re-parameterization of the candidate set (2.1), which does not change the integration in (2.2) (see more details in [97]). The unique solution of (2.2) under mild conditions is then shown to reduce to an ODE problem via the Euler-Lagrange method, where a numerical algorithm has also been provided.

2.3 The proposed formulation

In this work, we develop a variational approach for point cloud reconstruction on manifold $\mathcal{M}$ by extending the variational level set method in [126] and also incorporating the principal flow idea in [97] as a data fidelity. There are two main ingredients in the formulation. One is to extend the definition of the distance function which computes the *unsigned* distance from a grid point to the data set. Since the data points are defined on a manifold, one has to first replace the Euclidean distance in the original definition by the geodesic distance on $\mathcal{M}$. This can be easily done using a recent embedding approach developed in [121]. In the original variation level set reconstruction of point cloud, one uses only the data point location for defining the mismatch in the functional. Because of the limitation in the information provided, the gradient descent approach can be easily got trapped in a local minimizer. To drive the reconstruction out from a local minimizer of the functional, we propose to smooth the information from the data point location. The second main ingredient of the proposed approach is to incorporate the information from the PCA to drive our result to better fit the data points.

Mathematically, we propose the following energy given by

$$E(\Gamma) = \left[\int_\Gamma d^p(\mathbf{x}(s))\, ds\right]^{\frac{1}{p}} + \lambda \left[\int_\Gamma |\mathbf{n}(\mathbf{x}(s)) \cdot \mathbf{p}(\mathbf{x}(s))|^2 \, ds\right]^{\frac{1}{2}}, \qquad (2.3)$$

with the constraint $\Gamma \in \mathcal{M}$. In the energy, $p \geq 1$ is an integer, $\mathbf{p}(\mathbf{x}(s))$ is the principal direction estimated from the data points in a small neighborhood of $\mathbf{x}(s)$ along the curve Γ parameterized by s. The vector $\mathbf{n}(\mathbf{x}(s))$ along Γ is the normal direction of the reconstructed curve on $\mathcal{M}$ and $\lambda \geq 0$ controls the weight of this directional fidelity term to the location of the interface itself.

The first term is similar to the one proposed in [126] which also tries to fit a surface to the given data points. If the curve Γ is close to the data, the first line integral will be small. However, the current model is more restrictive than the original one in [126] since we now have an extra nonlinear constraint imposed on Γ. In the original formulation [126], the function $d(\mathbf{x})$ is a data matching function which equals to zero if $\mathbf{x} \in \mathcal{S}$ is one of a given data set and is straightly positive for $\mathbf{x} \notin \mathcal{S}$. A simple candidate is to set $d(\mathbf{x}) = g(\mathbf{x})$ where $g(\mathbf{x}) = dist(\mathbf{x}, \mathcal{S}) : \mathcal{M} \to [0, \infty]$ being the geodesic distance function on $\mathcal{M}$ to the dataset $\mathcal{S}$. Mathematically, this function can be obtained by solving the following surface eikonal equation

$$\|\nabla_{\mathcal{M}} g\| = 1 \qquad (2.4)$$

together with the boundary condition $g(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathcal{S}$ where $\nabla_{\mathcal{M}}$ is the surface gradient operator defined on the manifold $\mathcal{M}$. To have a more robust algorithm, we propose to relax the above condition and define $d(\mathbf{x})$ to be the smoothed version of $g(\mathbf{x})$ such that $d(\mathbf{x}) = G_\sigma \star g(\mathbf{x})$ for some Gaussian kernel G_σ with standard deviation σ. We will discuss a numerical algorithm to find the geodesic distance $g(\mathbf{x})$ and the justification of such a choice of $d(\mathbf{x})$ in the next section.

The second term is a fidelity term which tries to enforce the estimated principal direction $\mathbf{p}(\mathbf{x})$ to be perpendicular to the normal vector along the curve Γ at $\mathbf{x}$. In the case when the data points are living in the three dimensional space, i.e. $m = 3$, with the dimension of the manifold $\mathcal{M}$ being two, we can obtain a vector on the tangent plane at $\mathbf{x}$ (denoted by $T_{\mathbf{x}}\mathcal{M}$) perpendicular to $\mathbf{p}(\mathbf{x})$. If we denote this vector by $\mathbf{p}^{\perp}(\mathbf{x})$ such that $\mathbf{p}^{\perp}(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}$ and $\mathbf{p}(\mathbf{x}) \cdot \mathbf{p}^{\perp}(\mathbf{x}) = 0$, the fidelity term can be rewritten as

$$\left[\int_{\Gamma} \left|1 - \mathbf{n}(s) \cdot \mathbf{p}^{\perp}(s)\right|^2 \, ds\right]^{\frac{1}{2}}$$

which is similar to some variational formulations for surface processing via normal map [113, 114]. For a general scenario where both m and the dimension of $\mathcal{M}$ are unknown, the normal direction of the reconstructed surface $\mathbf{p}^{\perp}$ might not be readily available. It is therefore more natural to directly incorporate our model with the vector $\mathbf{p}$ estimated immediately from the PCA, as introduced in this work.

For the factor λ, since the first term has the same unit as the geodesic distance, we suggest to choose λ of order of h, the space step in discretization form. λ here controls the weight between the two terms, larger λ gives more weight to the fidelity term. So for some situations where data points are missed in some area, we can choose larger λ so that in the missing data area our solution follows the trend of neighbour area with data points.

In the level set formulation, we represent the manifold $\mathcal{M}$ implicitly by a level set function $\psi(\mathbf{x})$ for $\forall \mathbf{x} \in \mathbb{R}^m$ such that $\|\nabla\psi\| = 1$. Moreover, following [8, 14, 23], we introduce a second level set function $\phi(\mathbf{x})$ so that the curve Γ on $\mathcal{M} = \psi^{-1}(0)$ is implicitly represented by the intersection of these level set functions, i.e.

$$\Gamma = \{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \phi(\mathbf{x}) = 0\}\,.$$

Therefore, the minimization problem (2.3) can now be rewritten as

$$\begin{aligned} E(\phi;\lambda) &= \left[\int_{\mathbb{R}^m} d^p(\mathbf{x})\delta(\phi)\delta(\psi)\|\nabla\phi\|\,d\mathbf{x}\right]^{\frac{1}{p}} + \lambda\left[\int_{\mathbb{R}^m} |\mathbf{n}(\mathbf{x})\cdot\mathbf{p}(\mathbf{x})|^2\,\delta(\phi)\delta(\psi)\|\nabla\phi\|\,d\mathbf{x}\right]^{\frac{1}{2}} \\ &= \left[\int_{\mathcal{M}_\epsilon} d^p(\mathbf{x})\delta(\phi)\delta(\psi)\|\nabla\phi\|\,d\mathbf{x}\right]^{\frac{1}{p}} + \lambda\left[\int_{\mathcal{M}_\epsilon} |\mathbf{n}(\mathbf{x})\cdot\mathbf{p}(\mathbf{x})|^2\,\delta(\phi)\delta(\psi)\|\nabla\phi\|\,d\mathbf{x}\right]^{\frac{1}{2}} \end{aligned} \tag{2.5}$$

where the integral along Γ is converted into an integral in the full m-dimensional space $\mathbb{R}^m$ or an ϵ-neighborhood of the manifold, denoted by $\mathcal{M}_\epsilon$, using the Dirac delta functions $\delta(\psi(\mathbf{x}))$ and $\delta(\phi(\mathbf{x}))$. The normal vector $\mathbf{n}(s)$ along Γ is now converted into the normal vector $\mathbf{n}(\mathbf{x})$ along each level curve of $\phi(\mathbf{x})$ and can be easily computed using $\nabla\phi/\|\nabla\phi\|$.

To minimize (2.5), we first derive the variation of E with respect to the level set equation ϕ in a direction h. Formally, we obtain

$$\left(\frac{\delta E}{\delta\phi}, h\right) = -\gamma_1 I_1 - \lambda\gamma_2\left(I_2 + I_3\right)\,,$$

where

$$\begin{aligned} I_1 &= \int_{\mathcal{M}_\epsilon} \delta(\phi)\nabla\cdot\left[d^p(\mathbf{x})\delta(\psi)\frac{\nabla\phi}{\|\nabla\phi\|}\right] h\,d\mathbf{x} \\ I_2 &= \int_{\mathcal{M}_\epsilon} \left\{2\left(\mathbf{p}\cdot\frac{\nabla\phi}{\|\nabla\phi\|}\right)\nabla\cdot[\delta(\phi)\delta(\psi)\mathcal{P}\mathbf{p}]\right\} h\,d\mathbf{x} \\ I_3 &= \int_{\mathcal{M}_\epsilon} \left\{\delta(\phi)\nabla\cdot\left[\delta(\psi)\left(\frac{\nabla\phi}{\|\nabla\phi\|}\otimes\frac{\nabla\phi}{\|\nabla\phi\|}\right)\mathbf{p}\right]\right\} h\,d\mathbf{x} \\ \gamma_1 &= \frac{1}{p}\left[\int_{\mathcal{M}_\epsilon} d^p(\mathbf{y})\delta(\phi)\delta(\psi)\|\nabla\phi\|d\mathbf{y}\right]^{\frac{1}{p}-1} > 0 \\ \gamma_2 &= \frac{1}{2}\left[\int_{\mathcal{M}_\epsilon} |\mathbf{n}(\mathbf{y})\cdot\mathbf{p}(\mathbf{y})|^2\,\delta(\phi)\delta(\psi)\|\nabla\phi\|\,d\mathbf{y}\right]^{-\frac{1}{2}} > 0\,, \end{aligned}$$

and $\mathcal{P} = (I - \mathbf{n}\otimes\mathbf{n})$ is the projection operator onto the direction orthogonal

to $\mathbf{n}$. Now, based on the gradient descent, we introduce an artificial time t and determine the steady state solution to the following time-dependent equation with some initial guess,

$$\frac{\partial \phi}{\partial t} = \delta(\phi)\nabla \cdot \left\{ \left[\gamma_1 \, d^p(\mathbf{x}) + \lambda\gamma_2 \left(\mathbf{n} \cdot \mathbf{p} \right)^2 \right] \mathbf{n} \, \delta(\psi) \right\} + \lambda\gamma_2 \nabla \cdot \left[2(\mathbf{n} \cdot \mathbf{p}) \mathcal{P}\mathbf{p} \, \delta(\phi)\delta(\psi) \right] . \tag{2.6}$$

Both terms on the right hand side have factors $\delta(\phi)$ and $\delta(\psi)$ which concentrate the computations only near the curve Γ on the manifold $\mathcal{M}$. Following [124], we can further replace $\delta(\phi)$ by $\|\nabla\phi\|$, and $\delta(\psi)$ by $\|\nabla\psi\| = 1$. If we reinitialize and orthogonalize the level set functions such that $\|\nabla\phi\| = 1$ and $\nabla\phi \cdot \nabla\psi = 0$, we finally obtain

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \nabla \cdot \left[\gamma_1 \, d^p(\mathbf{x}) \, \mathbf{n} + \lambda\gamma_2 (\mathbf{p} + \mathcal{P}\mathbf{p})(\mathbf{p} \cdot \mathbf{n}) \right] \\ &= \nabla \cdot \left[\gamma_1 \, d^p(\mathbf{x}) \, \nabla\phi + \lambda\gamma_2 (\mathbf{p} + \mathcal{P}\mathbf{p})(\mathbf{p} \cdot \nabla\phi) \right] . \end{aligned} \tag{2.7}$$

We then solve this partial differential equation up to the steady state. The zero level set of the solution gives an implicit representation of the reconstruction on the given manifold $\mathcal{M}$.

To end this section, we give the following properties and observations concerning the functional which help us to understand the performance of the algorithm.

Observation 2.3.1. *The two terms in the energy are of different units. The first term has the unit of length and it has a scale depending on the manifold, while the second term is an integral of a scalar function which has no unit. To balance these two terms, one can pick $\lambda = O(L)$ for the length scale L for the manifold.*

Observation 2.3.2. *Although the vector $\mathbf{p}$ estimated from the PCA is unique only up to the sign, the factor $(\mathbf{p} + \mathcal{P}\mathbf{p})(\mathbf{p} \cdot \nabla\phi)$ remains unchanged if we replace $\mathbf{p}$ by $-\mathbf{p}$. This implies that the sign of $\mathbf{p}$ will not affect our result.*

Observation 2.3.3. *Since* $\|\mathbf{p}\|^2 = (\mathbf{n}\cdot\mathbf{p})^2 + \|\mathcal{P}\mathbf{p}\|^2 = (\mathbf{n}\cdot\mathbf{p})^2 + \mathbf{p}\cdot\mathcal{P}\mathbf{p}$*, we have*

$$\begin{aligned}\|\mathbf{p}+\mathcal{P}\mathbf{p}\|^2 &= \|\mathbf{p}\|^2 + \|\mathcal{P}\mathbf{p}\|^2 + 2(\mathbf{p}\cdot\mathcal{P}\mathbf{p}) \\ &= 1 + \left[1-(\mathbf{n}\cdot\mathbf{p})^2\right] + 2\left[1-(\mathbf{n}\cdot\mathbf{p})^2\right] \\ &= 4 - 3(\mathbf{n}\cdot\mathbf{p})^2 \geq 1\,.\end{aligned}$$

Therefore, the term $\mathbf{p}+\mathcal{P}\mathbf{p}$ *can never be the zero vector. This implies that the second term in the divergence operator cannot be zero, except when the normal* $\mathbf{n} = \nabla\phi$ *is orthogonal to the directional vector* $\mathbf{p}$ *estimated from the PCA. In the evolution of* ϕ*, the component in the functional corresponding to the principal direction always contributes to the functional unless* Γ *is tangent to* $\mathbf{p}$*.*

2.4 Implementation details

In this section, we explain the implementation details. As mentioned before, the function $d(x)$ in the functional depends on the geodesic distance function $g(\mathbf{x})$. We will first introduce a fast sweeping algorithm to numerically compute $g(\mathbf{x})$. Then we will discuss the choice of the function $d(\mathbf{x})$. To improve the computational complexity, we will discuss an approximation to the typical PCA which can be efficiently determined. To improve the stability and accuracy of the overall algorithm, we follow the standard regularization procedure in the level set community using the level set reinitialization and orthogonalization. The numerical implementations of these two algorithms will be given at the end of this section.

2.4.1 The geodesic function $g(\mathbf{x})$

We determine the geodesic function $g(\mathbf{x})$ by solving the surface eikonal equation (2.4). In this work, we follow our recent embedding approach developed in [121] which extends and improves the algorithm in [81]. Since the solution of the surface eikonal equation on an implicit surface is the intrinsic weighted distance function, we approximate such distance function by the Euclidean weighted distance function defined in a tubular neighbourhood of the implicit surface. In other words, we replace the surface eikonal equation (2.4) by the typical eikonal equation $\|\nabla g\| = 1$ defined in the small neighborhood of $\mathcal{M}$, i.e. $\mathcal{M}_\epsilon$, which can be easily solved by the fast sweeping method developed based on the Lax-Friedrichs Hamiltonian or the Godunov Hamiltonians [58, 123].

The fundamental idea of the fast sweeping method is to design an upwind, monotone and consistent discretization for the nonlinear term $\|\nabla g\|$. Furthermore, if there is a simple local solver for the discretized system, one can then obtain an efficient numerical algorithm for solving the nonlinear partial differential equation. The term *sweeping* comes from the fact that all methods in this class can be easily incorporated with the symmetric Gauss-Seidel iterations which may lead to an $O(N)$ algorithm with N the total number of mesh points in the computational domain. In the two-dimensional space ($m = 2$), for example, the local solver for the Godunov Hamiltonian can be explicitly constructed and it leads to the iterative formula

$$g_{i,j} \leftarrow \begin{cases} \min(g^x_{\min}, g^y_{\min}) + \Delta x & \text{if } |g^x_{\min} - g^y_{\min}| > \Delta x \\ \frac{1}{2}\left[g^x_{\min} + g^y_{\min} + \sqrt{2\Delta x^2 - (g^x_{\min} - g^y_{\min})^2}\right] & \text{otherwise.} \end{cases} \tag{2.8}$$

The corresponding local solver for the LxF Hamiltonian is given by

$$g_{i,j} \leftarrow \frac{g_{i+1,j} + g_{i-1,j} + g_{i,j+1} + g_{i,j-1}}{4} + \frac{\Delta x}{2}\left[1 - \sqrt{\left(\frac{g_{i+1,j} - g_{i-1,j}}{2\Delta x}\right)^2 + \left(\frac{g_{i,j+1} - g_{i,j-1}}{2\Delta x}\right)^2}\right]. \quad (2.9)$$

To correctly impose the boundary condition on the boundary $\partial\mathcal{M}_\epsilon$ taking care of the causality, however, we have proposed to introduce an extra layer of computational tube

$$\partial\mathcal{M}_{\epsilon'} = \{\mathbf{x} \in \mathbb{R}^m : \epsilon < \psi(\mathbf{x}) < \epsilon'\}$$

and impose an extension equation in this computational domain. In two dimensions, for example, $\mathbf{v} \cdot \nabla g = (p, q) \cdot (g_x, g_y) = 0$, we can use the simple upwind differencing to discretize the equation and derive the update formula [68, 20, 71, 72]. At each point (x_i, y_j), we define $p^\pm = \frac{1}{2}(p_{i,j} \pm |p_{i,j}|)$ and, therefore, obtain the update formula given by

$$g_{i,j} \leftarrow \frac{p^+ g_{i-1,j} - p^- g_{i+1,j} + q^+ g_{i,j-1} - q^- g_{i,j+1}}{|p_{i,j}| + |q_{i,j}|}. \quad (2.10)$$

We summarize the overall fast sweeping method in Algorithm 1. For more details, we refer interested readers to [121].

2.4.2 The function $d(\mathbf{x})$

The data mismatch function $d(\mathbf{x})$ is crucial in the algorithm. Numerically, it might not be a good idea to directly use $d(\mathbf{x}) = g(\mathbf{x})$ since such choice will give $d(\mathbf{x}_i) = 0$ for any data $\mathbf{x}_i \in \mathcal{S}$ and the evolution of ϕ will easily get stuck at these local minimum when it touches any of these data points. The situation will be worse if the dataset $\mathcal{S}$ contains noise because the interface reconstructed

Algorithm 1: A fast sweeping method for $|\nabla_\Sigma g| = 1$ [121].

Data: The source location $\mathbf{x}_s$, the mesh size Δx, the level set representation of the surface ψ_i, the inner tube radius ϵ and the outer tube radius ϵ'

Result: g_i in the computational tube

Initialization: Set $g_i = 0$ if $\mathbf{x}_i$ is at the point source, otherwise $g_i = \infty$

while *not converges* **do**

  for *each of the* 2^m *sweeping directions* **do**

    if $|\psi_i| \leq \epsilon$ *and* $g_i \neq 0$ **then**

      update g_i using (2.8) for the Godunov Hamiltonian or (2.9) for the LxF Hamiltonian

    else if $\epsilon < |\psi_i| \leq \epsilon'$ **then**

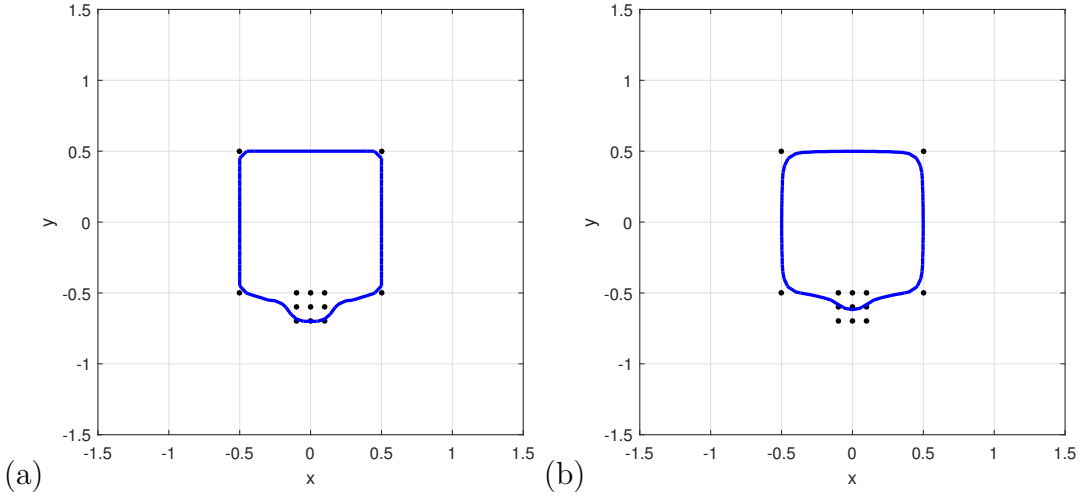
      update g_i using (2.10) with $\mathbf{v} = \text{sgn}(\psi)\nabla\psi/\|\nabla\psi\|$

    end

  end

end

will be significantly polluted by the perturbation. A simple example is shown in Figure 2.1 (a). We consider $\mathcal{M} = \mathbb{R}^2$ and therefore the geodesic function $d(\mathbf{x})$ is now reduced back to the usual Euclidean distance function. We assume data are given at the four corners of a square with extra points located on the side. The reconstruction with the choice of $d(\mathbf{x}) = g(\mathbf{x})$ is plotted using the blue curve. The level set function ϕ is stuck because of those outmost data points.



(a) (b)

Figure 2.1: (a) Typical scenario when $d(\mathbf{x}) = g(\mathbf{x})$ with $\lambda = 0$. (b) $d(\mathbf{x}) = G_\sigma \star g(\mathbf{x})$ with $\lambda = 0$. The data points given in $\mathcal{M} = \mathbb{R}^2$ are plotted in black dots while the reconstructed curve is plotted in blue.

Following the idea of edge-based segmentation algorithms [59, 18], we regularize

$g(\mathbf{x})$ by a simple low pass filter. This can be done easily by using the box filter or simply convolving the geodesic distance function with a Gaussian, denoted by

$$d(\mathbf{x}) = G_\sigma \star g(\mathbf{x}) \tag{2.11}$$

for some standard deviation σ. Since the geodesic distance function is defined in a small neighborhood of the manifold, i.e. $\mathcal{M}_\epsilon$, the box filter implies

$$d(\mathbf{x}) = \frac{\int_{\mathcal{B}_r(\mathbf{x})\cap\mathcal{M}_\epsilon} g(\mathbf{y})d\mathbf{y}}{\int_{\mathcal{B}_r(\mathbf{x})\cap\mathcal{M}_\epsilon} d\mathbf{y}},$$

where $\mathcal{B}_r(\mathbf{x})$ is a ball of radius r centered at the point $\mathbf{x} \in \mathcal{M}_\epsilon$ with $r = O(\epsilon)$. We plot in Figure 2.1 (b) the reconstruction based on this smoothed geodesic distance function with the same dataset as in Figure 2.1 (a). Even though the solution is now regularized with the corners significantly smoothened, such regularity would be important in developing a robust algorithm when dealing with noised data.

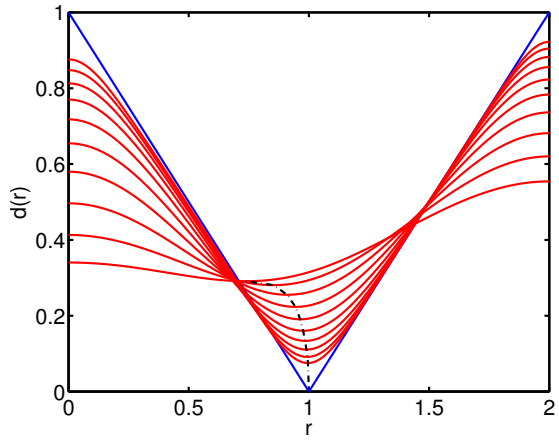


Figure 2.2: Regularization using $d(\mathbf{x}) = G_\sigma \star g(\mathbf{x})$ with $g(\mathbf{x}) = |\|\mathbf{x}\| - r_0|$ and $r_0 = 1$. The minimizer of $d(r)$ for different σ are plotted in black dash-dotted line.

Having said that the smoothing is important for noisy data, the smoothing parameter σ cannot be arbitrarily large since introducing an excessive amount of regularization to the functional might lead to an over-smoothed and undesired solution. To see this, we consider the case where $g(\mathbf{x})$ is given by $|\|\mathbf{x}\| - r_0|$

representing infinitely many data points that are given on the circle of radius r_0 centered at the origin. We consider the function $G_\sigma \star g(\mathbf{x})$ for various magnitude of σ. The larger the value of σ, the larger the averaging window and therefore the solution $d(\mathbf{x})$ will be smoother. Figure 2.2 shows the function $d(r) = d(\|\mathbf{x}\|)$ as a function of r, the distance away from the origin. The solutions for various $\sigma > 0$ are plotted in red. As we can see, the minimum of $d(r)$ shifts towards left as we increase σ. Although the overall energy (2.3) for a circular Γ of radius r^* and $\lambda = 0$ is given by $E(r^*) = [2\pi r^* d^p(r^*)]^{\frac{1}{p}}$, Figure 2.2 has clearly demonstrated that the minimizer to (2.3) might be significantly perturbed if the data mismatch function is over-smoothed.

Once we determine $d(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{M}_\epsilon$, we further extend it to those grid points adjacent to the computational tube by solving the extension equation $\mathbf{n} \cdot \nabla d = 0$ using the fast sweeping method as discussed in [68, 20, 71, 72, 121].

2.4.3 The estimation $\mathbf{p}(\mathbf{x})$ from the PCA

The standard way to determine the principal direction $\mathbf{p}$ consists of two steps. The first step is to project all data points onto a tangent plane at $\mathbf{x}$ on $\mathcal{M}$ using the log map. Then we can apply the typical PCA on the $\mathbb{R}^{m-1}$ plane. In our formulation, however, such approach is not directly applicable. The first concern is about the computational efficiency. Since we need $\mathbf{p}$ at every grid point in our computing tube $\mathcal{M}_\epsilon$, the computational complexity is therefore $O(N)$ when N is the number of data points in $\mathcal{S}$. Secondly, $\mathbf{x}$ is a grid point and it is not necessarily on $\mathcal{M}$. In particular, the point $\mathbf{x}_i$ lives only on the $(m-1)$-dimensional manifold $\{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \psi(\mathbf{x}_i)\}$ but not necessarily in $\psi^{-1}(0)$.

Motivated by [97], we propose to compute the principal direction $\mathbf{p}(\mathbf{x})$ in the following way. At each grid point $\mathbf{x}$, we first collect all data points within a small neighbourhood, denoted by $\mathcal{S}_\mathbf{x} = \{\mathbf{x}_i \in \mathcal{S} : \|\mathbf{x} - \mathbf{x}_i\|_2 < \delta\}$. Then, we

simply define $\mathbf{p}(\mathbf{x})$ to be the normalized eigenvector corresponding to the largest magnitude eigenvalue of

$$\sum_{\mathbf{x}_i \in S_{\mathbf{x}}} (\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T . \tag{2.12}$$

Note that this approach performs the standard PCA in the Euclidean space by relaxing the nonlinear Riemannian constraint. It is *different* from the tangent PCA which involves a projection of all data points onto the tangent plane at $\mathbf{x}$ using the log map. For the ease of the discussion, we will denote the corresponding principal direction on the tangent plane by $\mathbf{p}_{\log}$. In practice, however, it might be numerically challenging to compute the log map on Riemannian manifolds. Instead, we simply replace the log map by the simple Euclidean orthogonal projection and denote the resulting principal direction by $\mathbf{p}_{\text{orth}}$. Indeed, we do not expect to have the same $\mathbf{p}$, $\mathbf{p}_{\log}$ and $\mathbf{p}_{\text{orth}}$. But we are going to show that under some mild conditions on the data, we have $\mathbf{p} = \mathbf{p}_{\text{orth}}$ which turns out to be a good approximation to $\mathbf{p}_{\log}$.

Theorem 2.4.1. *Given $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \in \mathcal{M} \subset \mathbb{R}^3$ for $n > 0$ and denote $\mathbf{y}_i = (\mathbf{x}_i - \mathbf{x})$. Let $\mathbf{n}$ be the unit normal of the tangent plane at $\mathbf{x}$, and $\mathbf{t}_i$ be the unit tangent component of $\mathbf{y}_i$ on the tangent plane. If $\mathbf{y}_i \cdot \mathbf{t}_i > \sqrt{2}\ (\mathbf{y}_i \cdot \mathbf{n})$ for all data points, we have $\mathbf{p} = \mathbf{p}_{orth}$.*

To show this theorem, we need the following two lemmas.

Lemma 2.4.1. *The principal direction $\mathbf{p}_{orth}$ is the first or the second principal direction of the standard PCA in the Euclidean space.*

Proof. Let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ be the set of data points in a small neighbourhood of $\mathbf{x}$ with $\mathbf{x}$ and $\mathbf{x}_i \in \mathcal{M} \subset \mathbb{R}^m$ for all $i = 1, \cdots, n$. Introducing $\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}$ and

$$\mathbf{y}_i = (\mathbf{y}_i \cdot \mathbf{t}_i)\ \ \mathbf{t}_i + (\mathbf{y}_i \cdot \mathbf{n})\ \ \mathbf{n}$$

where $\mathbf{t}_i$ is the unit tangent component of $\mathbf{y}_i$ on the tangent plane at $\mathbf{x}$ and $\mathbf{n}$ is the corresponding unit normal vector of the tangent plane, we compute the standard PCA of the Euclidean space by determining the principal eigenvector of

$$M_n = \sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^T = \sum_{i=1}^{n} \left[(\mathbf{y}_i \cdot \mathbf{t}_i)^2 \, \mathbf{t}_i \mathbf{t}_i^T + (\mathbf{y}_i \cdot \mathbf{n})^2 \, \mathbf{n}\mathbf{n}^T \right] = E_n + F_n$$

with $E_n = \sum_{i=1}^{n} (\mathbf{y}_i \cdot \mathbf{t}_i)^2 \, \mathbf{t}_i \mathbf{t}_i^T$ and

$$F_n = \sum_{i=1}^{n} (\mathbf{y}_i \cdot \mathbf{n})^2 \, \mathbf{n}\mathbf{n}^T = \left[\sum_{i=1}^{n} (\mathbf{y}_i \cdot \mathbf{n})^2 \right] \mathbf{n}\mathbf{n}^T = \xi_n \, \mathbf{n}\mathbf{n}^T$$

where $\xi_n = \sum_{i=1}^{n} (\mathbf{y}_i \cdot \mathbf{n})^2 \geq 0$. Note that $\mathbf{p}_{\text{orth}}$ is simply the principal eigenvector of E_n. The component F_n has only one non-zero eigenvalue given by ξ_n with the corresponding eigenvector $\mathbf{n}$.

Let $\mathbf{s}_j$ be the eigenvectors of E_n for $j = 1, \cdots, m$ with corresponding eigenvalues λ_j with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$, i.e. $E_n \mathbf{s}_j = \lambda_j \mathbf{s}_j$. Since $\mathbf{n}$ is orthogonal to all $\mathbf{t}_i$, it is clear that $\mathbf{s}_m = \mathbf{n}$ is an eigenvector of E_n with the corresponding eigenvalue $\lambda_m = 0$. This implies that

$$M_n \mathbf{n} = (E_n + F_n)\mathbf{n} = 0\mathbf{n} + F_n \mathbf{n} = \xi_n \, \mathbf{n} \, ,$$

and so $\mathbf{n}$ is also an eigenvector of M_n with the corresponding eigenvalue given by ξ_n. For $\mathbf{s}_j \neq \mathbf{n}$, we have

$$M_n \mathbf{s}_j = E_n \mathbf{s}_i + F_n \mathbf{s}_j = E_n \mathbf{s}_i + 0\mathbf{s}_j = \lambda_j \mathbf{s}_j \, .$$

Therefore, the matrices M_n and E_n have the same set of eigen-pairs, except for the eigenvector $\mathbf{n}$ where the corresponding eigenvalue of E_n is 0 while that of M_n

is ξ_n. Finally, if $\lambda_1 > \xi_n$, we have $\mathbf{p} = \mathbf{p}_{\text{orth}}$. Otherwise, $\mathbf{p}_{\text{orth}}$ is the second principal direction of the standard PCA in the Euclidean space. □

Lemma 2.4.2. *Follow Lemma 2.4.1 and consider $m = 3$. If $\mathbf{y}_i \cdot \mathbf{t}_i > \sqrt{2}\,(\mathbf{y}_i \cdot \mathbf{n})$ for all data points $\mathbf{x}_i$, we have $\lambda_1 > \xi_n$.*

Proof. Recall that $\xi_n = \sum_{i=1}^{n} (\mathbf{y}_i \cdot \mathbf{n})^2 = \mathbf{n}^T M_n \mathbf{n}$. First, we will show that for any $n > 0$, there always exists two vectors $\mathbf{g}$ and $\mathbf{h}$ such that $(\mathbf{g} \cdot \mathbf{h}) = 0$ and

$$\mathbf{g}^T M_n \mathbf{g} + \mathbf{h}^T M_n \mathbf{h} > 2\xi_n \tag{2.13}$$

if $\mathbf{y}_i \cdot \mathbf{t}_i > \sqrt{2}\,(\mathbf{y}_i \cdot \mathbf{n})$ for all i.

We prove this by induction. When $n = 1$, we obtain

$$M_1 = (\mathbf{y}_1 \cdot \mathbf{t}_1)^2\, \mathbf{t}_1 \mathbf{t}_1^T + (\mathbf{y}_1 \cdot \mathbf{n})^2\, \mathbf{n}\mathbf{n}^T\,.$$

One can simply choose $\mathbf{g} = \mathbf{t}_1$ and $\mathbf{h}$ to be a unit vector perpendicular to both $\mathbf{g}$ and $\mathbf{n}$. In the three dimensional case, i.e. $m = 3$, we can use $\mathbf{h} = \frac{\mathbf{t}_1 \times \mathbf{n}}{\|\mathbf{t}_1 \times \mathbf{n}\|}$. Then

$$\mathbf{g}^T M_1 \mathbf{g} + \mathbf{h}^T M_1 \mathbf{h} = (\mathbf{y}_1 \cdot \mathbf{t}_1)^2 + 0 > \left[\sqrt{2}\,(\mathbf{y}_1 \cdot \mathbf{n})\right]^2 = 2\xi_1\,.$$

Now, assuming that (2.13) holds when $n = k$ and adding one more data point into our data set so that the number of data is $k + 1$, we have

$$\begin{aligned} \xi_{k+1} &= \sum_{i=1}^{k+1} (\mathbf{y}_i \cdot \mathbf{n})^2\, \xi_k + (\mathbf{y}_{k+1} \cdot \mathbf{n})^2\,, \\ M_{k+1} &= \sum_{i=1}^{k+1} \mathbf{y}_i \mathbf{y}_i^T = M_k + (\mathbf{y}_{k+1} \cdot \mathbf{t}_{k+1})^2\, \mathbf{t}_{k+1} \mathbf{t}_{k+1}^T + (\mathbf{y}_{k+1} \cdot \mathbf{n})^2\, \mathbf{n}\mathbf{n}^T\,. \end{aligned}$$

Since $\{\mathbf{g}, \mathbf{h}\}$ forms an orthonormal basis of the two dimensional tangent plane, we have $\|\mathbf{g}^T \mathbf{t}_{k+1}\|^2 + \|\mathbf{h}^T \mathbf{t}_{k+1}\|^2 = \|\mathbf{t}_{k+1}\|^2 = 1$ for any vector $\mathbf{t}_{k+1}$ on the tangent

plane and so,

$$\begin{aligned}
& \mathbf{g}^T M_{k+1}\mathbf{g} + \mathbf{h}^T M_{k+1}\mathbf{h} \\
= \;& \mathbf{g}^T M_k\mathbf{g} + \mathbf{h}^T M_k\mathbf{h} + (\mathbf{y}_{k+1}\cdot\mathbf{t}_{k+1})^2 \left(\mathbf{g}^T\mathbf{t}_{k+1}\mathbf{t}_{k+1}^T\mathbf{g} + \mathbf{h}^T\mathbf{t}_{k+1}\mathbf{t}_{k+1}^T\mathbf{h}\right) \\
= \;& \mathbf{g}^T M_k\mathbf{g} + \mathbf{h}^T M_k\mathbf{h} + (\mathbf{y}_{k+1}\cdot\mathbf{t}_{k+1})^2 \left(\|\mathbf{g}^T\mathbf{t}_{k+1}\|^2 + \|\mathbf{h}^T\mathbf{t}_{k+1}\|^2\right) \\
= \;& \mathbf{g}^T M_k\mathbf{g} + \mathbf{h}^T M_k\mathbf{h} + (\mathbf{y}_{k+1}\cdot\mathbf{t}_{k+1})^2 \\
> \;& 2\xi_k + (\mathbf{y}_{k+1}\cdot\mathbf{t}_{k+1})^2 > 2\xi_k + \left[\sqrt{2}\,(\mathbf{y}_{k+1}\cdot\mathbf{n})\right]^2 = 2\xi_{k+1}\,.
\end{aligned}$$

To conclude, if we have $\mathbf{y}_i\cdot\mathbf{t}_i > \sqrt{2}\,(\mathbf{y}_i\cdot\mathbf{n})$ for all $1\le i\le n$, then

$$\mathbf{g}^T M_n\mathbf{g} + \mathbf{h}^T M_n\mathbf{h} > 2\xi_n = 2\mathbf{n}^T M_n\mathbf{n}\,.$$

So it gives either $\mathbf{g}_n^T M_n\mathbf{g}_n > \mathbf{n}^T M_n\mathbf{n}$ or $\mathbf{h}_n^T M_n\mathbf{h}_n > \mathbf{n}^T M_n\mathbf{n}$, and which implies that $\mathbf{n}$ cannot be the principal direction of M_n. □

Theorem 2.4.1 follows immediately by these two lemmas. In practice, however, there could be grid points $\mathbf{x}$ far away from the data set $\mathcal{S}$, i.e.

$$\min_{\mathbf{x}_i\in\mathcal{S}} \|\mathbf{x}-\mathbf{x}_i\|_2 > \epsilon\,,$$

and so the set $\mathcal{S}_\mathbf{x}$ is in fact empty. This implies that $n = 0$ in Theorem 2.4.1 and so we cannot use it to find an approximation of $\mathbf{p}$. The fix to this problem is not unique. Since we do not actually expect that the zero level set of ϕ would stay clear to such a location, we can maximize the contribution from the fidelity term by setting $\mathbf{p}(\mathbf{x})$ using the unit normal vector of the level set surface given by $\mathbf{n}(\mathbf{x}) = \nabla\phi/\|\nabla\phi\|$. Such choice, however, would lead to a nonlinear optimization problem. Even though one might linearize the energy by requiring $\mathbf{p}(\mathbf{x})$ to be $\nabla\phi^n/\|\nabla\phi^n\|$ where ϕ^n is the level set function at the previous time step at the gradient flow, we do not recommend this approach since the computational

complexity of this step will be expensive. In this work instead, we propose to simply randomly assign a unit vector to $\mathbf{p}(\mathbf{x})$ if $\mathbf{x}$ is far from $\mathcal{S}$. Because adjacent vectors are independent from each other, perturbation of the zero level set will not alter the contribution from the fidelity term to the overall energy in general. This implies that the change in the energy will be automatically contributed mostly from the part with the data mismatch function.

2.4.4 Level set regularizations

In the previous derivation, we have assumed that the level set function $\phi(\mathbf{x})$ is a signed distance function such that $\|\nabla\phi\| = 1$ for all time $t > 0$ to simplify some computations. This property, however, cannot be satisfied automatically in the gradient flow. This implies that ϕ may develop steep and flat gradients at or near the zero level set as it evolves according to equation (2.7), making the computed location of Γ and further computations inaccurate. Therefore, we use the following regularization procedure which consists of reinitialization and orthogonalization. To restore the distance property for the level set functions, the usual way is to make ϕ a signed distance function without moving its zero level set appreciably. This can be done by the so-called reinitialization. There are also many well-developed numerical methods to efficiently obtain their numerical solutions. We refer all interested readers to [95] and thereafter for a more detailed discussion.

Following the typical level set approach [95], we first reinitialize the level set functions. We introduce an artificial time τ and solve the following reinitialization equation

$$\phi_\tau + \text{sign}(\phi^n)(\|\nabla\phi\| - 1) = 0 \tag{2.14}$$

with the initial condition $\phi(\tau = 0)$ being the intermediate state given by ϕ^n at the n-th iteration step for (2.7). Another regularization is the orthogonal

extension from $\psi = 0$ so that two zero level set surfaces are perpendicular to each other. This can reduce the error in extracting the curve explicitly when we need to visualize the intersection of these surfaces. Mathematically, this can be achieved by solving the following orthogonal extension equation

$$\phi_\tau + \text{sign}(\phi^n)\nabla\psi \cdot \nabla\phi = 0\,, \tag{2.15}$$

which is simply an advection equation with a discontinuous velocity $\mathbf{u} = \text{sign}(\phi^n)\nabla\psi$. There are standard numerical implementations and packages available for solving these two nonlinear equations of the Hamilton-Jacobi type nowadays.

2.4.5 Improving the computational efficiency

To improve the computational complexity of the algorithm, we also incorporate the following two approaches. The first one is to localize the computations near the zero level set in $\mathcal{M}_\epsilon$. In practice, we are interested only in the solution near the zero level set. There is no reason why we need to solve the level set equation in the whole computational domain $\mathcal{M}_\epsilon$ up to the steady state. In the following examples, we only solve these two regularization equations for one single $\Delta\tau$ step each and use these intermediate numerical solutions to replace the original level set function ϕ^n. To further reduce the computational cost, we also apply the local level set strategy as developed in [98] which can concentrate all computational power in the region near the zero level set within the small neighborhood $\mathcal{M}_\epsilon$.

The second way to improve the computational efficiency of the proposed algorithm is a two-stage strategy for the level set initialization. At the initial time when the zero level set is far away from the dataset, we have proposed in Section 2.4.3 to assign a random vector to $\mathbf{p}(\mathbf{x})$. Because of the independence of any neighboring vectors, we do not expect the fidelity term containing $\mathbf{p}(\mathbf{x})$ would

have much influence on the level set evolution. To further speedup the computations, we propose the following *two-stage strategy*. Since the algorithm simply does not want any contribution from $\mathbf{p}(\mathbf{x})$ when the zero level set is far from the dataset, we start the whole algorithm by first turning the fidelity term off with $\lambda = 0$ and obtain the steady state solution. The zero level set of this first-stage solution is expected to be already close to the dataset. Then, the second stage is to use this steady state solution as the initial condition for the non-zero λ. Since the zero level set of this solution is now near the dateset $\mathcal{S}$, the contribution from $\mathbf{p}(\mathbf{x})$ will provide extra information to better fit the data on the manifold $\mathcal{M}$.

2.4.6 Numerical discretization

In this section, we discuss the numerical discretization of various equations in the proposed algorithm.

Level set equations

We first discuss the discretization of the equation (2.7). The equation can be expressed in the form

$$\frac{\partial \phi}{\partial t} = \nabla \cdot [a(\mathbf{x}) \nabla \phi] + \nabla \cdot [(\mathbf{p}(\mathbf{x}) \cdot \nabla \phi)\, \mathbf{b}(\mathbf{x})]\,,$$

where

$$\begin{aligned} a(\mathbf{x}) &= \gamma_1\, d^p(\mathbf{x})\,, \\ \mathbf{b}(\mathbf{x}) &= \lambda \gamma_2 [\mathbf{p}(\mathbf{x}) + \mathcal{P}\mathbf{p}(\mathbf{x})] \\ &= (b^x(\mathbf{x}), b^y(\mathbf{x}), b^z(\mathbf{x}))^T\,, \\ \mathbf{p}(\mathbf{x}) &= (p^x(\mathbf{x}), p^y(\mathbf{x}), p^z(\mathbf{x}))^T\,. \end{aligned}$$

The diffusion term on the right hand side of the equation can be easily discretized using the symmetric central difference given by

$$\begin{aligned} & \frac{1}{\Delta x^2}\left[a_{i-\frac{1}{2},j,k}\phi_{i-1,j,k} - \left(a_{i-\frac{1}{2},j,k} + a_{i+\frac{1}{2},j,k}\right)\phi_{i,j,k} + a_{i+\frac{1}{2},j,k}\phi_{i+1,j,k}\right] \\ + & \frac{1}{\Delta y^2}\left[a_{i,j-\frac{1}{2},k}\phi_{i,j-1,k} - \left(a_{i,j-\frac{1}{2},k} + a_{i,j+\frac{1}{2},k}\right)\phi_{i,j,k} + a_{i,j+\frac{1}{2},k}\phi_{i,j+1,k}\right] \\ + & \frac{1}{\Delta z^2}\left[a_{i,j,k-\frac{1}{2}}\phi_{i,j,k-1} - \left(a_{i,j,k-\frac{1}{2}} + a_{i,j,k+\frac{1}{2}}\right)\phi_{i,j,k} + a_{i,j,k+\frac{1}{2}}\phi_{i,j,k+1}\right] \end{aligned}$$

where

$$\begin{aligned} a_{i\pm\frac{1}{2},j,k} &= \frac{1}{2}\left[a(\mathbf{x}_{i,j,k}) + a(\mathbf{x}_{i\pm1,j,k})\right] \\ a_{i,j\pm\frac{1}{2},k} &= \frac{1}{2}\left[a(\mathbf{x}_{i,j,k}) + a(\mathbf{x}_{i,j\pm1,k})\right] \\ a_{i,j,k\pm\frac{1}{2}} &= \frac{1}{2}\left[a(\mathbf{x}_{i,j,k}) + a(\mathbf{x}_{i,j,k\pm1})\right] . \end{aligned}$$

The second term on the right hand side is a little tedious since it consists of nine mixed derivatives of ϕ including $[b^x p^x \phi_x]_x$, $[b^x p^y \phi_y]_x$, $[b^x p^z \phi_z]_x$, $[b^y p^x \phi_x]_y$, $[b^y p^y \phi_y]_y$, $[b^y p^z \phi_z]_y$, $[b^z p^x \phi_x]_z$, $[b^z p^y \phi_y]_z$ and $[b^z p^z \phi_z]_z$. Numerically, we construct the projection $\mathcal{P}\mathbf{p}_{i,j,j}$ in $\mathbf{b}(\mathbf{x}_{i,j,k})$

$$\mathcal{P}\mathbf{p}_{i,j,k} = \mathcal{P}\mathbf{p}(\mathbf{x}_{i,j,k}) = \mathbf{p}_{i,j,k} - (\mathbf{n}_{i,j,k} \cdot \mathbf{p}_{i,j,k})\,\mathbf{n}_{i,j,k}$$

with $\mathbf{n}_{i,j,k} = \nabla\phi_{i,j,k}$ computed using central difference. Then, we approximate those mixed derivatives of ϕ using central difference. To simplify the following notation, we introduce $c^{x,x} = b^x p^x$, $c^{x,y} = b^x p^y$, and etc. Now, we have

$$\begin{aligned} [c^{x,x}\phi_x]_x &= \frac{1}{\Delta x^2}\left[c^{x,x}_{i-\frac{1}{2},j,k}\phi_{i-1,j,k} - \left(c^{x,x}_{i-\frac{1}{2},j,k} + c^{x,x}_{i+\frac{1}{2},j,k}\right)\phi_{i,j,k} + c^{x,x}_{i+\frac{1}{2},j,k}\phi_{i+1,j,k}\right] \\ [c^{x,y}\phi_y]_x &= \frac{1}{4\Delta x\Delta y}\left[c^{x,y}_{i+1,j,k}(\phi_{i+1,j+1,k} - \phi_{i+1,j-1,k}) - c^{x,y}_{i-1,j,k}(\phi_{i-1,j+1,k} - \phi_{i-1,j-1,k})\right] \end{aligned}$$

where $c^{x,y}_{i\pm\frac{1}{2},j,k} = (c^{x,y}_{i,j,k} + c^{x,y}_{i\pm1,j,k})/2$.

Finally, the temporal derivative is approximated by the forward Euler method with the stability condition

$$\Delta t < \frac{1}{2} \frac{\Delta x^2}{\max(\|a(\mathbf{x})\|, \|\mathbf{b}(\mathbf{x})\|)} \,.$$

Level set regularizations

The reinitialization equation (2.14) and the orthogonal extension equation (2.15) are both Hamilton-Jacobi equations. There are well-developed algorithms for obtaining highly accurate numerical solutions. In this work, we apply the TVDRK [106] and the WENO [78, 56] for the temporal and the spatial derivatives, respectively. The Hamiltonian is approximated by the Lax-Friedrichs Hamiltonian. We will just briefly state the discretization procedure here but refer readers to the above references and thereafter for some detail discussions. For simplicity, we consider only the one-dimension Hamilton-Jacobi equation given by

$$\frac{\partial \phi}{\partial \tau} + H\left(\frac{\partial \phi}{\partial x}\right) = 0 \,.$$

High dimensional generalization is straight forward. First, we discuss the spatial discretization of the equation. We approximate the Hamiltonian by the Lax-Friedrichs Hamiltonian given by

$$\hat{H}^{\mathrm{LxF}}\left(\left.\frac{\partial \phi}{\partial x}\right|_{x=x_i}\right) = H\left(\frac{p_i^+ + p_i^-}{2}\right) - \sigma_x \left(\frac{p_i^+ - p_i^-}{2}\right) \,,$$

where the derivatives $p^{\pm}$ are approximated by the WENO strategy. The viscosity parameter σ_x is chosen so that for any x_i we have the monotonicity requirements given by

$$\sigma_x \geq \max_{\phi,\phi_x} \left| \frac{\partial \hat{H}^{\mathrm{LxF}}(\phi_x)}{\partial \phi_x} \right| .$$

Taking the third order WENO (i.e. WENO3) as an example, we have

$$\begin{aligned} p_i^- &= (1-\omega_-)\left(\frac{\phi_{i+1}-\phi_{i-1}}{2\Delta x}\right) + \omega_- \left(\frac{3\phi_i - 4\phi_{i-1} + \phi_{i-2}}{2\Delta x}\right) , \\ p_i^+ &= (1-\omega_+)\left(\frac{\phi_{i+1}-\phi_{i-1}}{2\Delta x}\right) + \omega_+ \left(\frac{-3\phi_i + 4\phi_{i+1} - \phi_{i+2}}{2\Delta x}\right) , \end{aligned}$$

with $\omega_\pm = \left(1 + 2\gamma_\pm^2\right)^{-1}$ and

$$\gamma_- = \frac{\delta^2 + (\phi_i - 2\phi_{i-1} + \phi_{i-2})^2}{\delta^2 + (\phi_{i+1} - 2\phi_i + \phi_{i-1})^2} \text{ and } \gamma_+ = \frac{\delta^2 + (\phi_i - 2\phi_{i+1} + \phi_{i+2})^2}{\delta^2 + (\phi_{i+1} - 2\phi_i + \phi_{i-1})^2}$$

for some small positive constant δ to avoid division by zero. Concerning the temporal derivative, we apply the method of line strategy. In particular, at each grid point, we update the solution using

$$\begin{aligned} \hat{\phi}_i^{m+1} &= \phi^m - \Delta\tau\, \hat{H}^{\mathrm{LxF}}\left(\left.\frac{\partial \phi^m}{\partial x}\right|_{x=x_i} \right) \\ \hat{\phi}_i^{m+2} &= \hat{\phi}^{m+1} - \Delta\tau\, \hat{H}^{\mathrm{LxF}}\left(\left.\frac{\partial \hat{\phi}^{m+1}}{\partial x}\right|_{x=x_i} \right) \\ \phi_i^{m+1} &= \frac{1}{2}\left(\phi_i^m + \hat{\phi}_i^{m+2}\right) . \end{aligned}$$

Because of the CFL stability condition, the time marching step is chosen to be $\Delta\tau = O(\Delta x)$.

2.4.7 The overall algorithm

The proposed method is summerized in Algorithm 2. Now, we discuss the computational complexity of the overall algorithm. Let N_x be the number of the grid points in each physical dimension and N be the total number of grid points

in the small neighborhood of the manifold $\mathcal{M}$ denoted by $\mathcal{M}_\epsilon$. We first discuss the computational cost of the implicit representation of $\mathcal{M}$ used in the proposed method and that by a typical explicit representation using triangulations. Since $\mathcal{M}$ is a co-dimensional one surface in $\mathbb{R}^3$, we have $N = O(N_x^2)$. Therefore the order of these two costs are in fact the same which is in fact optimal. Now, it takes $O(N)$ operations to determine $g(\mathbf{x})$, $d(\mathbf{x})$ and also $\mathbf{p}(\mathbf{x})$ because these functions are defined in a point-wise fashion. For each iteration step, it also take $O(N)$ operations to update and regularize the level set function ϕ if we do not implement the local level set approach but to update ϕ for all $\mathbf{x} \in \mathcal{M}_\epsilon$. If we further concentrate the computational power near the zero level set in $\mathcal{M}_\epsilon$, the computational complexity of each iteration can be further reduced to $O(N_x)$.

Algorithm 2: The proposed algorithm with the two-stage strategy.

Data: The data set $\mathbf{x}_s$, the mesh size $(\Delta x, \Delta y, \Delta z)$, the level set representation of the manifold ψ_i, the weight λ for the data fidelity

Result: Γ represented by the intersection of zero level set of ϕ and ψ

```
Initialization: Set φ such that Γ encloses all data points
Compute g(x) by Algorithm 1;
Determine d(x) by smoothing g(x) using (2.11);
Find p(x) by (2.12);
for λ' = 0 and then set λ' = λ do
    while E(φ; λ') in (2.5) does not converge do
        Update φ using (2.7) according to Section 2.4.6;
        Regularize φ using (2.14) and (2.15) according to Section 2.4.6;
    end
end
```

2.5 Reconstruction using an open curve

In the previous discussions, we have assumed that $\Gamma = \{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \phi(\mathbf{x}) = 0\}$ is a closed co-dimensional one manifold of $\mathcal{M}$. When we try to determine an open curve on a two dimensional $\mathcal{M}$, on the other hand, we cannot simply implicitly represent Γ by $\{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \phi(\mathbf{x}) = 0\}$. Instead, we follow [107]

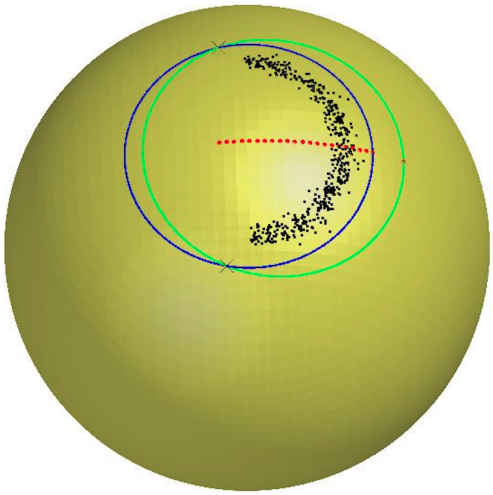


Figure 2.3: Reconstructing the data set using an open curve. The given data set is given in the black dots. The black crosses is the given two end points. The red points are trace of d we tested.The green curve represents the intersection of the zero level set of $\rho(\mathbf{x})$ and $\mathcal{M}$ where we identify regions where the intersection of the zero level set function of ϕ and $\mathcal{M}$, the blue curve, is updated.

and introduce a third level set function ρ to implicitly represent the end points of Γ by $\partial\Gamma = \{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \phi(\mathbf{x}) = \rho(\mathbf{x}) = 0\}$. In particular, we define the low dimension representation of the data set by $\Gamma = \{\mathbf{x} \in \mathbb{R}^m : \psi(\mathbf{x}) = \phi(\mathbf{x}) = 0 \text{ and } \rho(\mathbf{x}) < 0\}$.

The choice of ρ for the same curve is clearly not unique. Assuming that we are given the locations of $\partial\Gamma$, we propose the following algorithm to initialize the level set functions ϕ and ρ if the manifold $\mathcal{M}$ is the unit sphere $\mathbb{S}^2$ with the origin denoted by O. We denote the given two end points of Γ by A and B. If A and B are not on the opposite poles of the sphere, we find the midpoint along the geodesic between A and B on $\mathbb{S}^2$ and we denote that point by C_0. We let P be the plane containing these points A, B and C_0. Clearly the intersection of P and $\mathbb{S}^2$ gives the unique great circle passing through both points A and B. Now, we construct the plane Q which is perpendicular to P and passes through C_0 and the origin O. The intersection of Q and $\mathbb{S}^2$ will give another great circle on $\mathbb{S}^2$. We denote this great circle by $\mathcal{C}$ which can be parametrized by $C(\theta)$ with $C(0) = C_0$. A typical situation is shown in Figure 2.3. We plot the given data

set on $\mathbb{S}^2$ by black dots. The end points $\partial\Gamma$ are plot by black cross. The locations of various $C(\theta)$ are plotted in red dots. For each particular θ, we can determine unique signed distance function ϕ_θ containing A, B and $C(\theta)$, in the form of

$$\phi_\theta(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_\theta\| - r_\theta \,.$$

Finally, we gradually increase (or decrease) the value of θ from 0 until we find a value θ^* such that $\phi_{\theta^*}(\mathbf{x}_i) < 0$ for all given data points $\mathbf{x}_i$. This implies that the zero level set of ϕ_{θ^*} now enclosed all data points in $\mathbb{S}^2$ and, therefore, the intersection of $\{\phi_{\theta^*} = 0\}$ and $\mathbb{S}^2$ gives a good initial condition of ϕ. We are plotting this intersection by the blue curve in Figure 2.3. Once we have ϕ constructed from θ^*, we further increase (or decrease) the value of θ by ϵ and we can use it to define $\rho(\mathbf{x}) = \phi_{\theta^*+\epsilon\,\mathrm{sign}(\theta^*)}(\mathbf{x})$. The zero level set of ρ on $\mathbb{S}^2$ is plotted by green solid line in Figure 2.3. The black dots is the given data set. The black crosses is the given two end points. The red points are trace of d we tested. The green circle is the intersection of the zero level set of $\rho(\mathbf{x})$ and $\mathcal{M}$ such that all data points on the black curve is in the interior. The intersection of the zero level set function of ϕ and $\mathcal{M}$ is plotted in blue.

The overall algorithm for this case is almost the same like the one we developed above, except only the way how we define Γ and having an extra constraint on where we update the level set function. The overall algorithm is summarized in Algorithm 3.

2.6 Numerical examples and discussions

In this section, we consider various examples to demonstrate the effectiveness of the proposed algorithm. Unless specific otherwise, we have chosen $\lambda = 0.01$ in all of our numerical experiments. In the following examples, we first analytically

Algorithm 3: The proposed algorithm for an open curve with the two-stage strategy.

Data: The data set $\mathbf{x}_s$, the mesh size $(\Delta x, \Delta y, \Delta z)$, the level set representation of the manifold ψ_i, the weight λ for the data fidelity
Result: Γ represented by the intersection of zero level set of ϕ and ψ
Initialization: Set ϕ such that Γ encloses all data points, and ρ according to Section 2.5
Compute $g(\mathbf{x})$ by Algorithm 1;
Determine $d(\mathbf{x})$ by smoothing $g(\mathbf{x})$ using (2.11);
Find $\mathbf{p}(\mathbf{x})$ by (2.12);
for $\lambda' = 0$ *and then set* $\lambda' = \lambda$ **do**
  while $E(\phi; \lambda')$ *in (2.5) does not converge* **do**
    if $\rho(\mathbf{x}_i) < 0$ **then**
      Update ϕ using (2.7) according to Section 2.4.6;
    end
    Regularize ϕ using (2.14) and (2.15) according to Section 2.4.6;
  end
end

determine a target curve on the underlying manifold. These target solutions are plotted in red solid lines. Then the data points are randomly drawn from an exact target curves. In the case when we would like to test the robustness of the proposed algorithm, we add random Gaussian noise with the standard deviation equals to $\frac{h}{2}$ to these data with a constraint that the perturbed locations are still staying on the surface. These data sets are plotted using black dots while the level set $\{\phi = 0\}$ is shown in blue solid line.

In Section 2.6.1, we test our algorithm with closed curves. In Section 2.6.2, we solve problems with the solution being open curves. For some examples, we compare the performance of our algorithm and the Principal Flow method. In Section 2.6.3, we solve problems with outliers to show the robustness of our algorithm. In Section 2.6.4 we consider problems with incomplete data to demonstrate the effect of λ. To end this section, in Section 2.6.5, we show the application our algorithm on the real earthquake data.

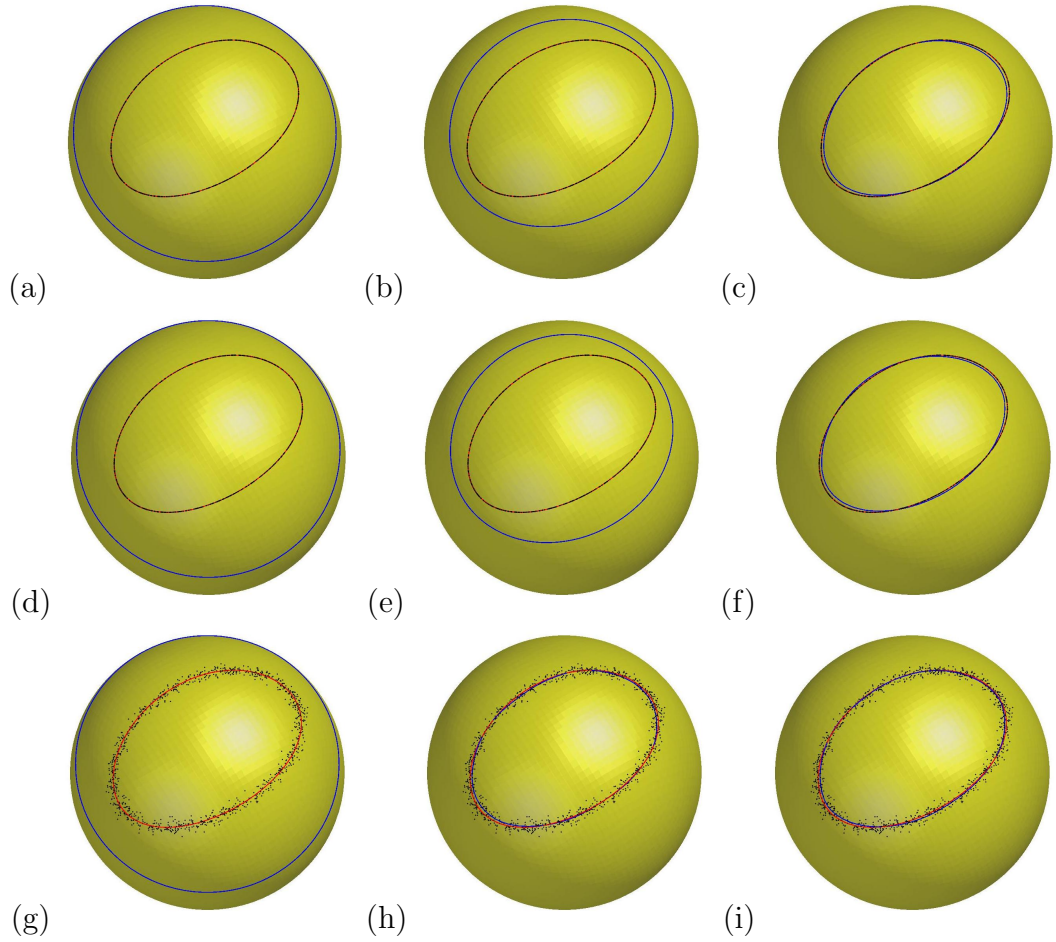


Figure 2.4: (Example 2.6.1) Ellipsoidal curves on a sphere. (a-c) Reconstruction using $\lambda = 0$. (a) The initial condition, (b) an intermediate solution and (c) the final solution. (d-f) Reconstruction using $\lambda = 0.01$. (d) The initial condition, (e) an intermediate solution and (f) the final solution. (g) The dataset with noise. Our reconstruction results using (h) $\lambda = 0$ and (i) $\lambda = 0.01$.

2.6.1 Closed curves on a sphere and a torus

Figure 2.4 (a-f) show the reconstruction using the proposed algorithm without and with the fidelity term using the estimation by the PCA. The variational formulation can robustly determine a curve Γ which fits the data reasonably well even when there is noise in the data, as shown in Figure 2.4 (c). In this example, the fidelity term from the PCA does not contribute much new information for the reconstruction. The iteration stops almost immediately when we switch from $\lambda = 0$ to a nonzero λ in the two stage strategy. Therefore, for simply cases

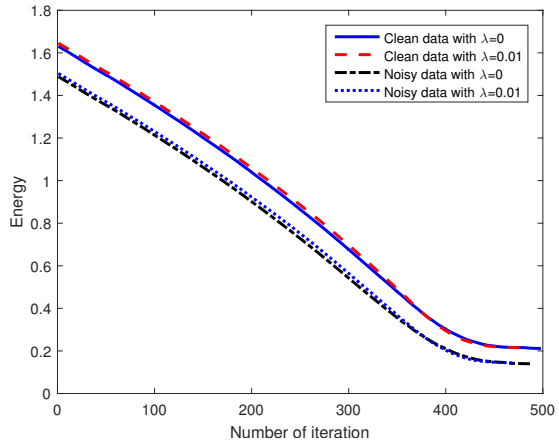
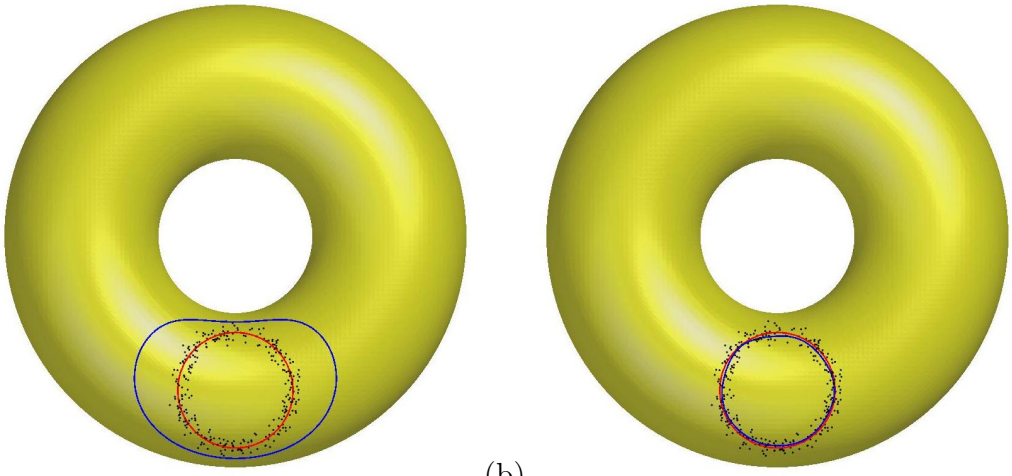


Figure 2.5: (Example 2.6.1) Reconstructing an ellipsoidal curve on a sphere. The change in the energy for various tests for the clean data and noisy data with $\lambda = 0$ and $\lambda = 0.01$.



(a) (b)

Figure 2.6: (Example 2.6.1) A closed target curve on a torus. (a) Our initial condition. (b) Result by our variational formula.

like these we are testing on, we simply start with a nonzero λ immediately. In Figure 2.5, we show the change in the energy in the iterations for all these cases. It is clearly that the mismatch energy is gradually reduced to its steady state as we evolve Γ according to the proposed method. For this example, it takes approximately 500 iterations to get to these final solutions.

The algorithm can be easily extended to manifolds other than a sphere. In Figure 2.6, we consider the reconstruction of a circular curve on a torus. The result by our algorithm can naturally estimate the target curve even when there

are significant amount of noise in the given data.

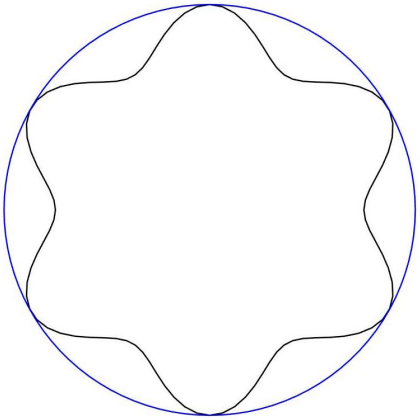
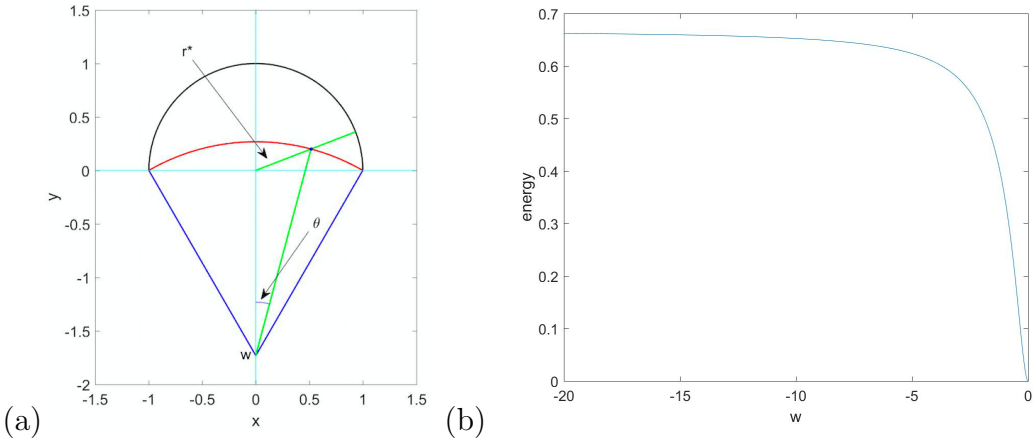


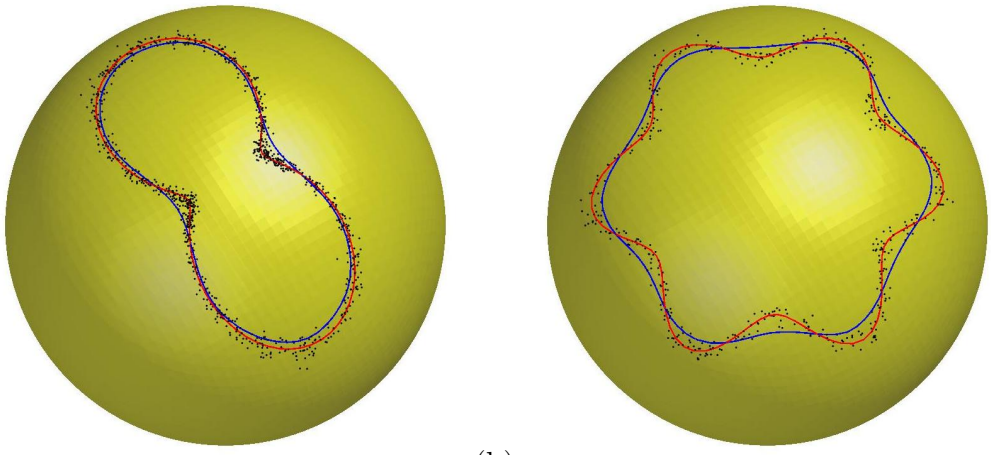
Figure 2.7: (Example 2.6.1) Reconstructing a nonconvex curve. The zero level set of ϕ is given in blue and is touching the target at the vertices.

One interesting example is to reconstruct a nonconvex object using the current approach. As the zero level set shrinks, we would like to know if the total length of the level curve would be able to increase itself to better fit the data at some point of the evolution. Figure 2.7 shows a scenario where the zero level set of $\phi(\mathbf{x})$ touches a six-folded star shape at its vertices. After this particular time, the curve will further shrink a little until the total length has to increase if we want to reach the *global* minimum of the functional.



(a) (b)

Figure 2.8: (Example 2.6.1) (a) The black semicircle with radius 1 is the data set. The red arc is part of a circle centered at $(0, w)$ with $w \leq 0$. And assume it is the evolution of Γ with end points being the same as the semi-circle and are fixed. (b) Energy of the red arc for different w.



(a) (b)

Figure 2.9: (Example 2.6.1) (a) A two-folded and (b) a six-folded curves on a sphere. Our reconstructions are drawn in blue solid lines.

To simplify the discussion, we approximate one-sixth of the above curve by a semi-circle of radius 1, as shown in Figure 2.8 (a). In this analysis, we determine the resulting energy of various segments and try to conclude that the weight based on the distance function $d(\mathbf{x})$ indeed allows an elongation of the curve to better fit the data. We consider the part of the smaller arc of a circle centered at $(0, w)$ touches the semi-circle at $(\pm 1, 0)$ for some $w < 0$, denoted by the red segment in Figure 2.8 (a). As w increases from $-\infty$ to 0, Γ evolves from the diameter of the semi-circle to the semi-circle itself. For each point $\mathbf{z}$ on this arc, we define θ to be the angle it made with the positive y-axis and r^* to the its distance from the origin. The shortest distance from $\mathbf{z}$ to the black semi-circle is therefore given by $d(\mathbf{z}) = 1 - r^*$ with

$$(r^*)^2 = 1 + 2w^2 - 2|w|\sqrt{1+w^2}\cos\theta\,.$$

This implies that the corresponding energy associated to this arc is

$$E(w) = 2\sqrt{1+w^2}\int_0^{\arctan\frac{1}{|w|}} (1-r^*)^2 d\theta\,.$$

Figure 2.8 (b) shows the energy $E(w)$ as we vary w. While the length of the arc increases from 2 to π as w changes from $-\infty$ to 0, the energy decreases monotonically to 0 reaching the *global* minimizer of the energy.

Two simple tests of the algorithm on non-convex shapes are shown in Figure 2.9. We first randomly draw some data points from a two-folded and a six-folded curves, as shown in (a) and (b), respectively. Gaussian noise is added to these points while constraining that the final location is still on $\mathbb{S}^2$. Our algorithm can still reach the target curve reasonably well, as shown by the blue solid curve.

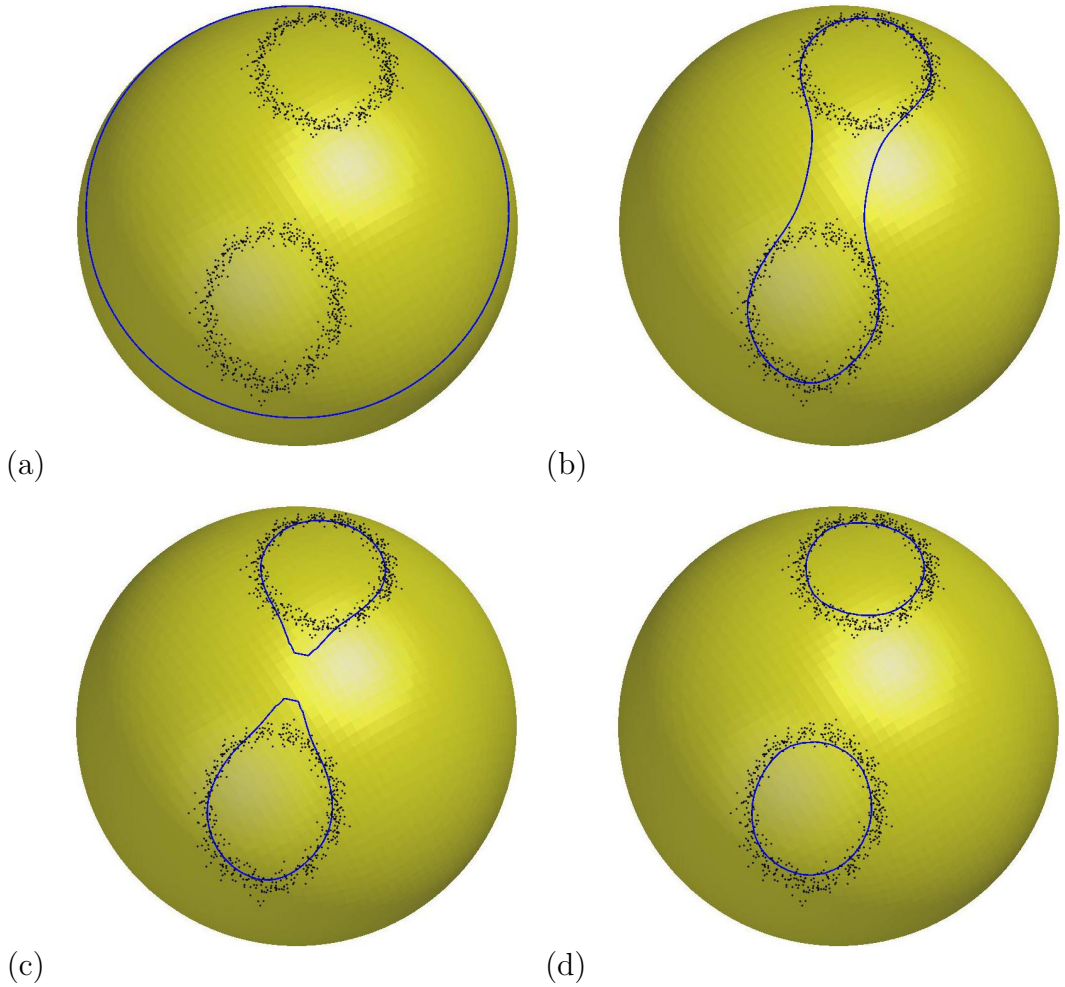
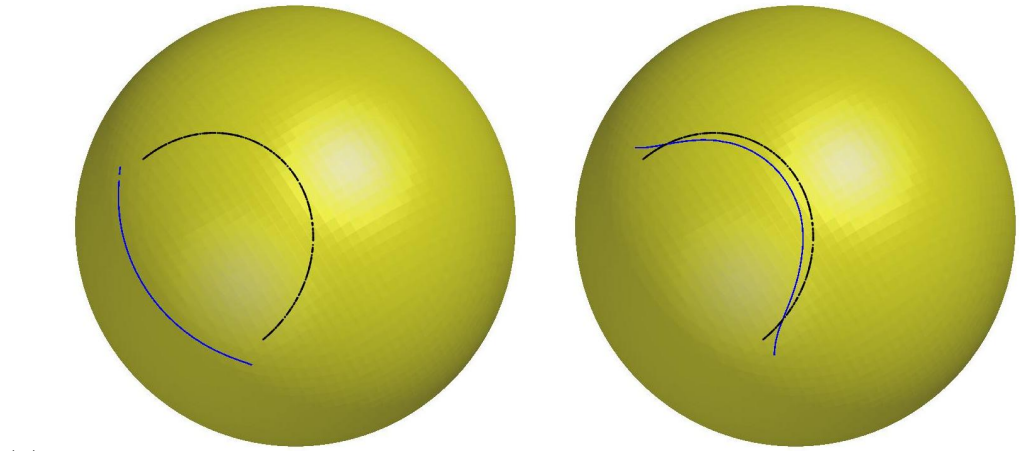


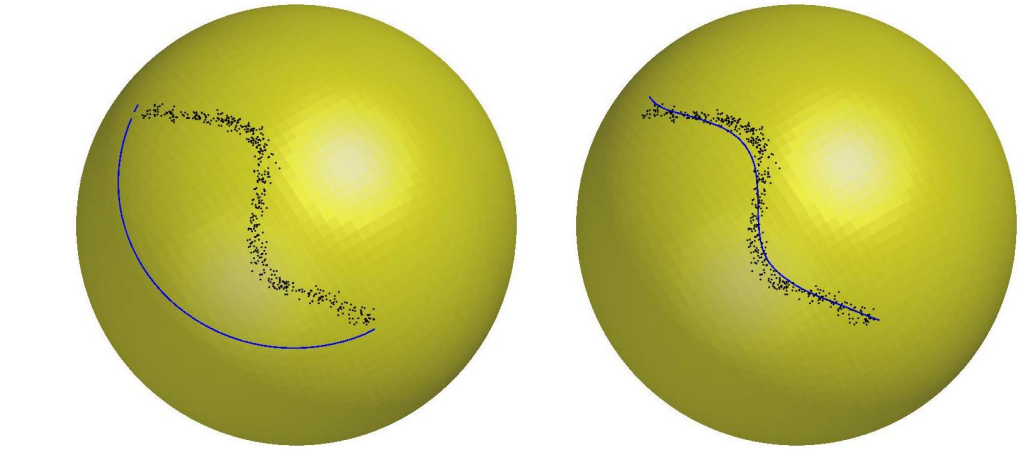
Figure 2.10: (Example 2.6.1) Reconstructing two circular curves on a sphere with $\lambda = 0.01$. (a) The initial condition, (b-c) two intermediate solutions showing the topological change in the level set evolution and (d) the final solution.

Because of the implicit representation of the reconstruction, the proposed method

does not require any *a priori* information on the topology of the underlying dataset. For example, in Figure 2.10, we are given some noisy data on a sphere generated from two circular curves. Once again, we start with an initial zero level set surrounding all data points, as shown in Figure 2.10 (a). As the level set evolves according to the proposed algorithm, it automatically splits into two disjoint curves without any user intervention as demonstrated in (b) and (c). The test example is a challenging one for explicit methods like the original principle flow method since user has to be able to segment the dataset into two disjoint groups and then identify an initial point for each of these groups.



(a)



(b)

Figure 2.11: (Example 2.6.2) Open curves on a sphere. We plot the given data points on the sphere using black dots. Our solutions Γ are plotted in blue. (Left) The initial condition and (b) the reconstructions.

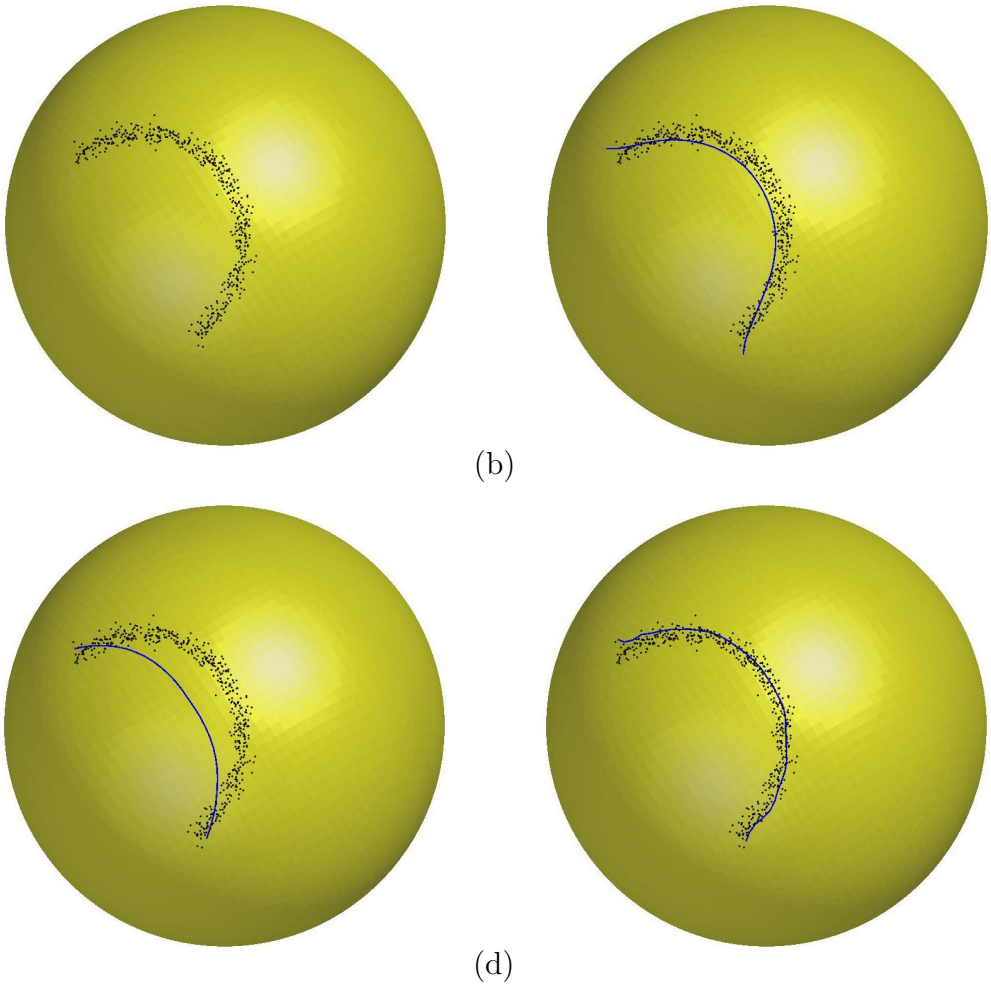


Figure 2.12: (Example 2.6.2) (a) Data sampled from a C-shape curve on a sphere with noise. (b) The result by our proposed approach. (c) The result by the principal flow method with the starting point being the Fréchet mean. (d) The result by the principal flow method with the starting point being the mid-point of the underlying clean C-shape curve.

2.6.2 Open curves on a sphere

In this example, we consider three cases where the target structures are represented by open curves given by a C-shape curve, a sine function and a noised C-shape curve. The result for the first two cases are shown in Figure 2.11. We can see that our variational formula can naturally handle these open curves. In the case of the C-shape curve, the initial condition does not actually prefer the final solution since the curve Γ has to increase his total length. Nevertheless our computed solution represents very well the given data. For the case with the

noised C-shape, we have also compared our result with that by the principal flow method developed in [97]. The results are shown in Figure 2.12. We have used two different ways to initialize the principal flow method. The first one is based on the Fréchet mean of the dataset which gives a very rough initial starting point of the tracing algorithm. Because of this, the solution from the principal flow does not give a well-reconstruction of the data, as shown in Figure 2.12 (c). To incorporate more information to the reconstruction, we also initialize the principal flow method using the mid-point of the underlying clean C-shape curve. The reconstruction result is shown in Figure 2.12 (d). The method can now reconstruct the underlying manifold pretty well. In real applications, however, the mid-point to the *unknown* underlying curve is unavailable. Comparing to the original principle flow method, the proposed method can automatically drive the evolution starting from an initial level set function enclosing the data to a reasonably well reconstruction.

2.6.3 Examples with outlier

To further test the performance of our proposed algorithm under noise, we manually add some outliers to the noisy data sets. We consider a circular target curve and also an open curve on $\mathbb{S}^2$. We plot the evolutions of our reconstruction under the datasets with and without outliers in Figure 2.13.

In these examples, we found that the steady state solutions do not heavily depend on the choice of λ. For a zero λ, the proposed method can already capture the expected trend in the data on the manifold. When followed by a nonzero λ in the two stage strategy, the method stops almost immediately in a few extra iterations. The corresponding energy evolutions are shown in Figure 2.14. We observe that both red dashed lines take more iterations to reach the steady state. It implies that more iterations are required to get over the outliners in the data set. But

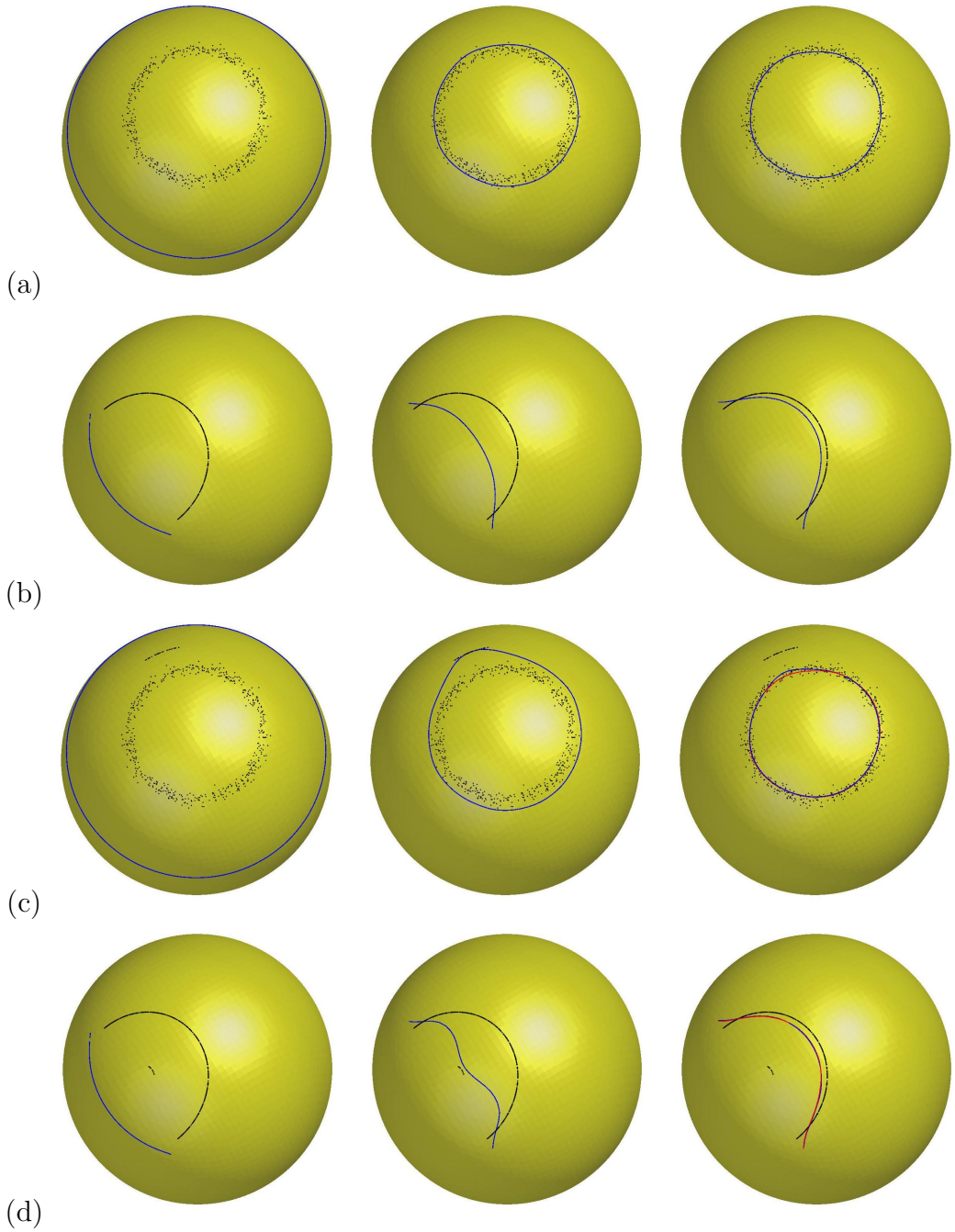
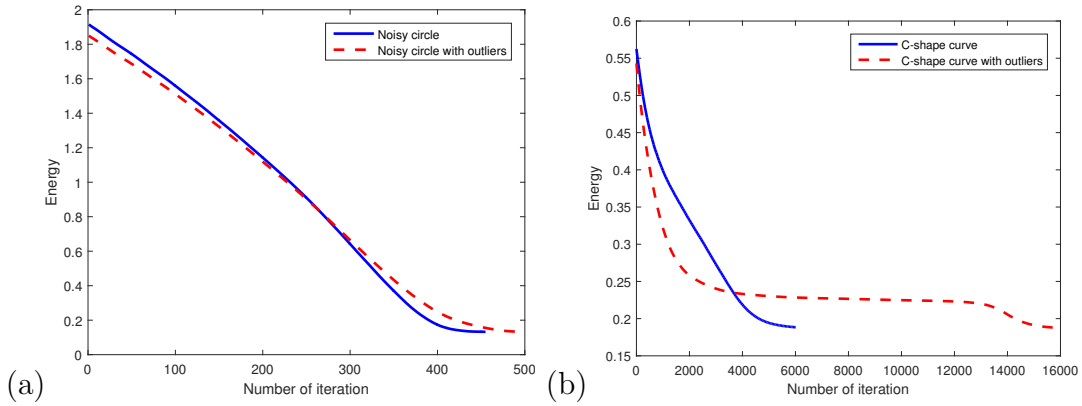


Figure 2.13: (Example 2.6.3) Data points with outlier are plotted in black. (a-b) Evolutions of the zero level set for data sets without outliers. (c-d) Evolutions of the zero level set for the corresponding data sets with outliers. The red curves are final results from (a) and (b) respectively.

more importantly, the existence of outliers in the given data does not affect our final reconstruction. The steady state solution seems to be the same like the one obtained by the correpsonding dataset without the outliers, as demonstrated in



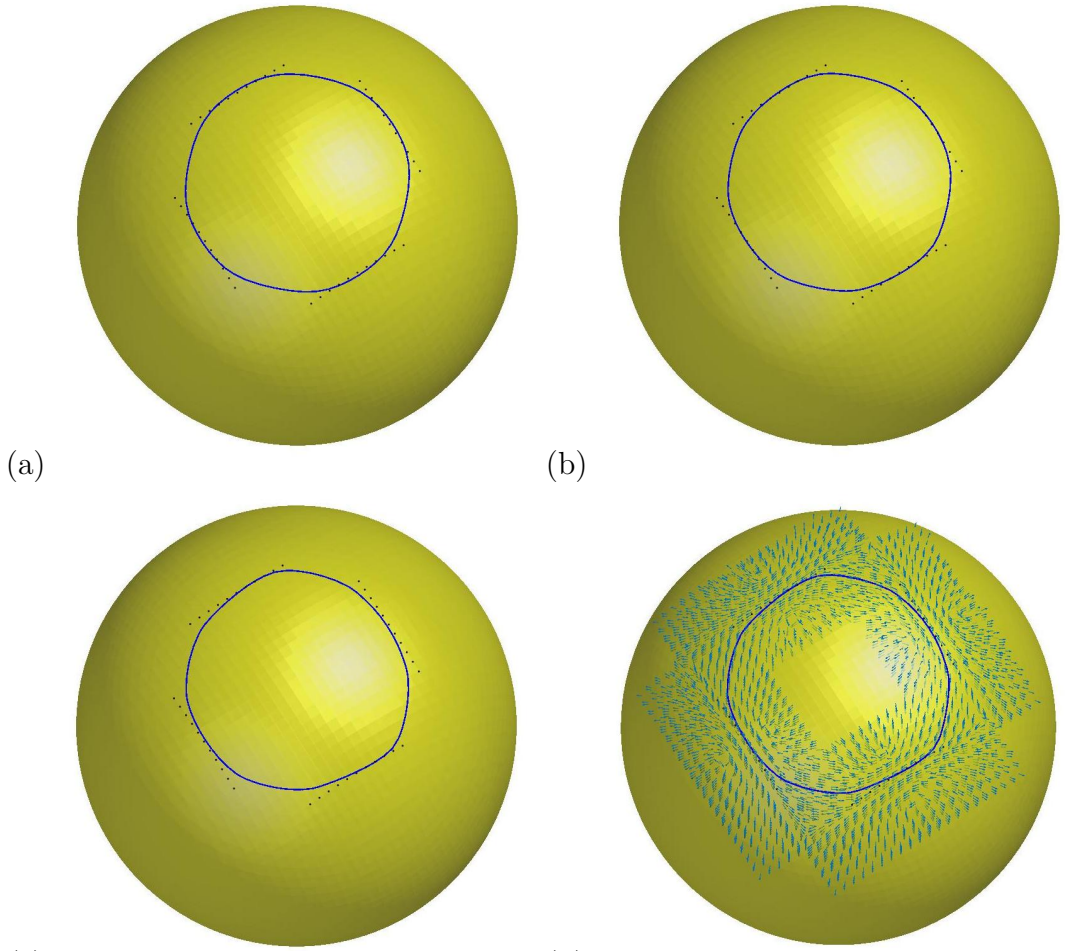
(a) (b)

Figure 2.14: (Example 2.6.3) Changes in the energy versus the iteration number. (a) Energies corresponding to the test examples in Figure 2.13 (a) and (c). (b) Energies corresponding to the test examples in Figure 2.13 (b) and (d).

Figure 2.13 and that the final energies are comparable in Figure 2.14.

2.6.4 Examples with incomplete information

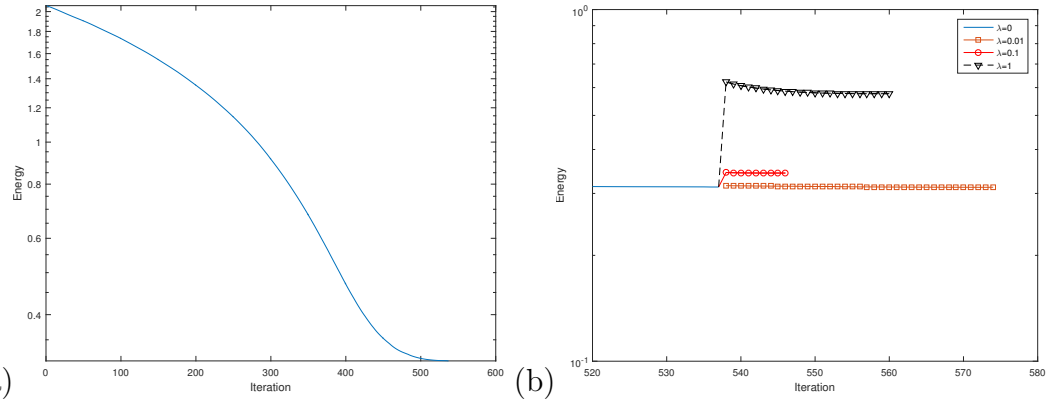
It is in general not reasonable to assume that the given data can perfectly resolve the underlying geometry on the manifold. In this problem, we assume that the given data can only sample part of the target shape. In particular, we use only partial data from a square on a sphere with missing data at all four corners. The reconstruction from our algorithm for various λ's are shown in Figure 2.15. Without the matching from the PCA, i.e. $\lambda = 0$, our solution are not able to recover the corner since the algorithm takes into account only the location of the data points on the manifold. However, as we increase the value of λ, from 0 in (a) to $\lambda = 1$ in (d), the contribution from the PCA gives a more realistic reconstruction and helps the curve Γ to predict a better trend in the data. Figure 2.16 shows the change in the energy with different λ's. In Figure 2.16 (a), we show the energy in the first stage of the iterations when we have $\lambda = 0$. It takes around 540 iterations to reach the steady state solution. In the second stage of the algorithm, we then use this steady state solution as the initial condition for different nonzero λ. This explains why there are jumps in the energies as shown



(a) (b)

(c) (d)

Figure 2.15: (Example 2.6.4) An example when we miss data points at some place on the original curve. Given data points are plotted in black dots. Our reconstruction based on (a) $\lambda = 0$, (b) $\lambda = 0.01$, (c) $\lambda = 0.1$ and (d) $\lambda = 1$. In (d), we have also plotted the PCA vectors near the data points.



(a) (b)

Figure 2.16: (Example 2.6.4) The change in the energy with (a) $\lambda = 0$ followed by (b) $\lambda = 0.01$, $\lambda = 0.1$ and $\lambda = 1$, respectively.

in Figure 2.16 (b).

2.6.5 Examples based on earthquake data

In this example, we test our algorithm using the earthquake data from the U.S. geological survey. We consider two datasets containing the epicenter of the earthquakes with magnitude larger than or equal to 4 occurred between December 29, 2014 to January 1, 2016 in the Russia-Alaska region and in the Australia region, as shown in Figure 2.17. Our initial conditions and results for these two data sets are shown in Figure 2.18 (a) and (b), respectively.

2.7 Conclusion

In this chapter we proposed a variational formulation for dimension reduction on Riemannian manifolds. The algorithm is developed based on the level set method to obtain a fully implicit formulation so that the method does not require any *a prior* knowledge on the topology of the structure we are reconstructing. We have tested the algorithm on various numerical examples with different manifolds. Even with given measurements with noise or outliers, the proposed algorithm gives good and robust reconstructions.

There are several possible extensions of our proposed algorithm. Even though it might need a careful study of the data structure such as [82, 22], the algorithm can be extended to handle even higher dimensional problems. Furthermore, since the method is a variational approach and does not require any labeled response from the user, such unsupervised algorithm has the potential to be extended to a manifold learning where the embedding is not necessarily known.

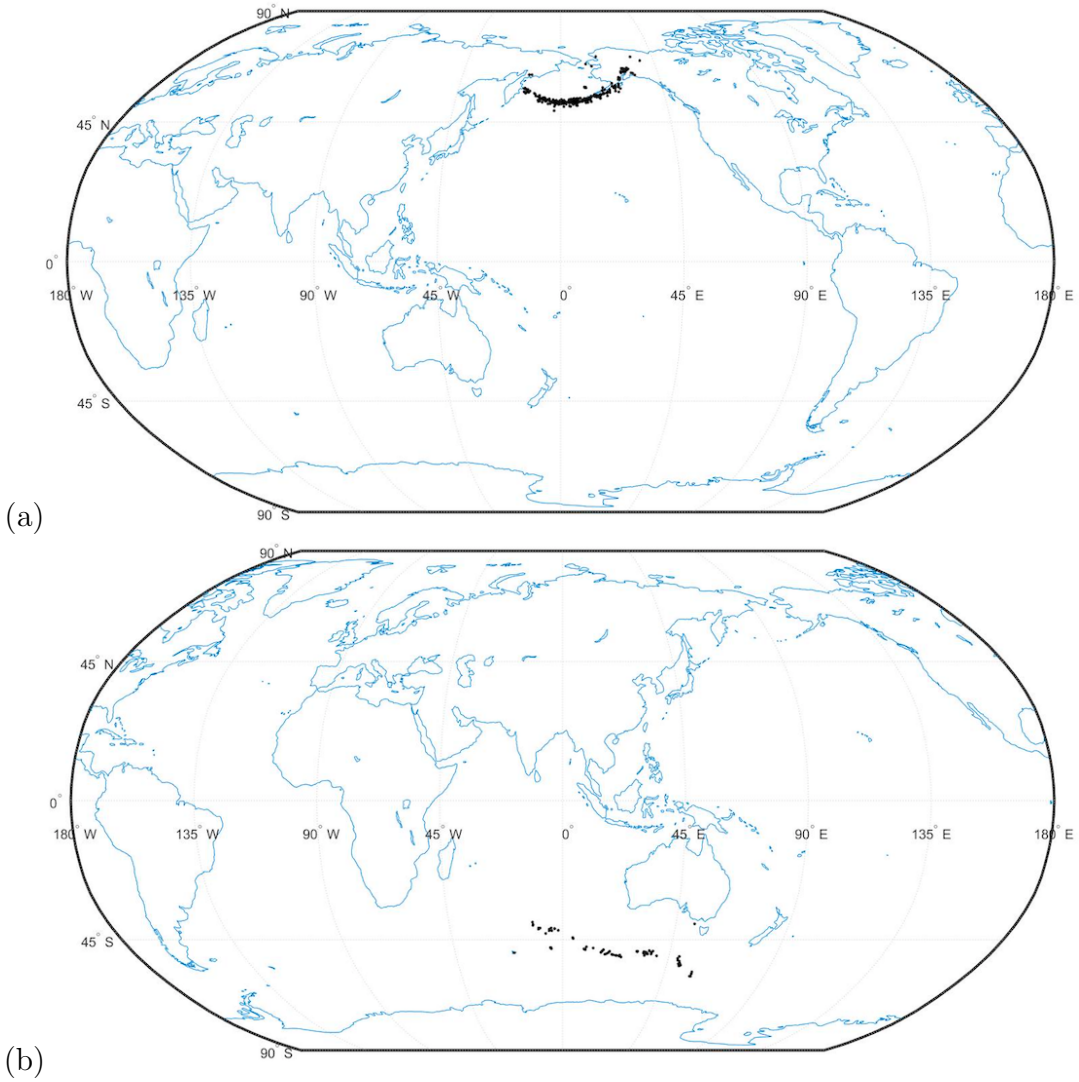
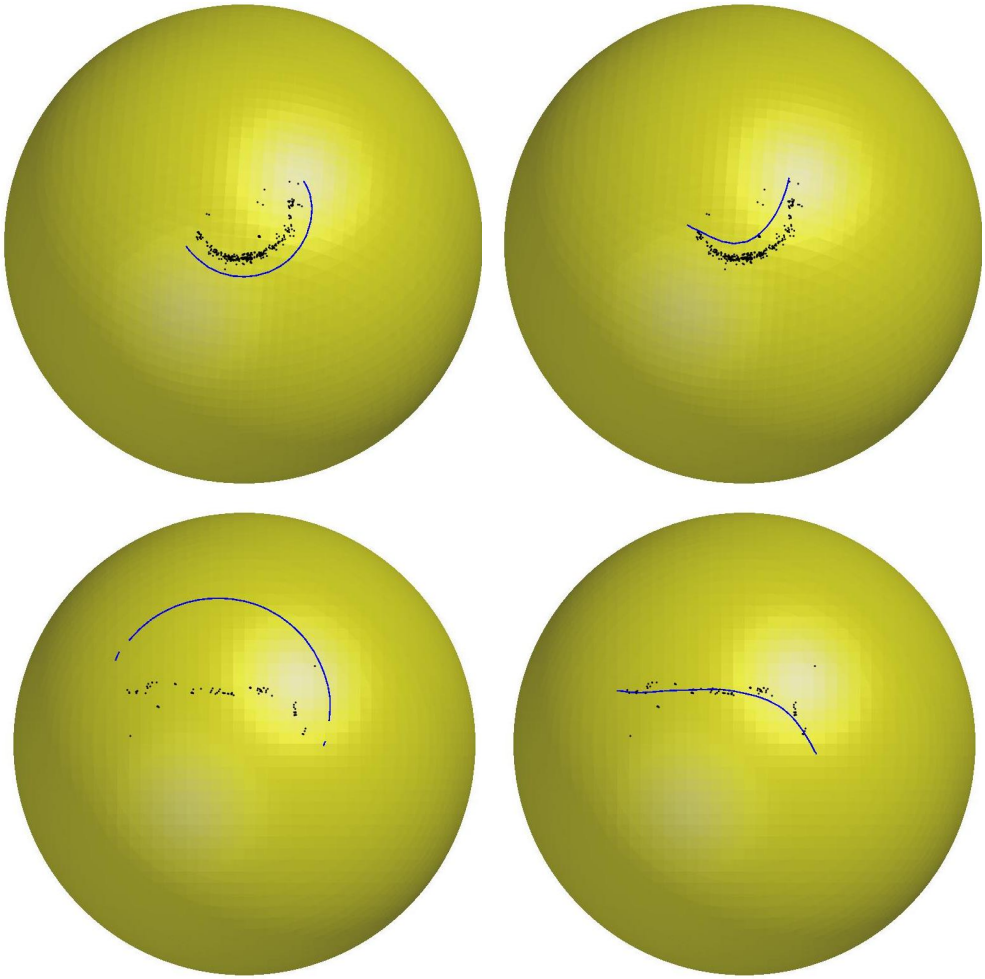


Figure 2.17: (Example 2.6.5) The location of the epicenters of earthquake with magnitude larger than between December 29, 2014 to January 1, 2016. (a) Epicenters of the earthquake in the Russia-Alaska region. (b) Epicenters of the earthquake in the Australia region.



(a)

(b)

Figure 2.18: (Example 2.6.5) (a) Our initial condition and result for earthquake in the Russia-Alaska region. (b) Initial condition and result in the Australia region.

Chapter 3

Operator-Splitting Method for the Two Dimensional Monge-Ampère Equation with the Dirichlet Boundary Condition

3.1 Introduction

Let Ω be a bounded domain of $\mathbb{R}^2$. The Dirichlet problem for the canonical Monge-Ampère equation reads as

$$\begin{cases} \det \mathbf{D}^2 u = f \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega\,, \end{cases} \tag{3.1}$$

$\mathbf{D}^2 u$ being the Hessian matrix of u, that is

$$\mathbf{D}^2 u = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 \partial x_2} \\ \frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_2^2} \end{pmatrix},$$

the functions f and g being given. If $f > 0$, problem (3.1) is a prototypical fully nonlinear elliptic boundary value problem. The existence and regularity properties of the solutions to fully nonlinear elliptic problems have been discussed in [41, 5, 17], a particular attention being given to the canonical Monge-Ampère equation in [52]. As shown in, e.g., [16, 6, 87, 40] and [33], the Monge-Ampère equation has a wide range of applications, differential geometry, optimal transportation, physics and mechanics among them.

Starting with [93] various numerical methods have been developed for the numerical solution of fully nonlinear elliptic boundary problems, problem (3.1) being the most investigated by far. The fast multiplication of these methods during the last decade has made keeping track of all of them an almost impossible task. Several of them have been reported in [33], but a visit to Google Scholar has become a must to have a more complete view. Focusing on those approaches with which we have some familiarity, we will classify them roughly into two families. The methods of the first family treat a finite difference or finite element approximation of the equation under consideration (possibly coupled to a regularization procedure as done in [34, 35]; see also [33]); such methods, and the iterative solution of the resulting discrete problems, are discussed in, e.g.,[38, 39, 7, 79, 104, 11, 12, 3]. Another approach is to reformulate the nonlinear elliptic problem as an optimization one; this can be done via least-squares or via the introduction of a well-chosen augmented Lagrangian algorithm. Such optimization based methods are discussed in e.g.[26, 29, 28, 15, 27, 30, 31, 87, 43, 47].

The method discussed in this chapter concerns problem (3.1), specifically. It relies on the following ingredients: (i) An equivalent divergence formulation of problem (3.1). (ii) An equivalent mixed formulation of the above divergence

formulation, with which we associate an initial value problem. (iii) The time-discretization by operator-splitting of the above initial value problem. (iv) A mixed finite element implementation of the above methodology. Another ingredient worth mentioning is the following one: In order to enforce the local positivity of the approximate Hessian, we employ at each time step a simple eigenvalue projection algorithm, similar to those widely used for low rank matrix approximation or, more recently in matrix completion via the so-called singular value shrinkage operator. Also, for those problems where the solution of problem (3.1) has enough regularity, we propose a two-stage strategy to further improve the speed of convergence: during the first stage, the dynamical system (flow) we consider is associated with $\partial u/\partial t$, while during the second stage we use $\partial Su/\partial t$, S being a well-chosen elliptic operator, giving to this second stage a Newton-like flavor.

As reported in, e.g., various chapters of [47] (see also the references therein), operator-splitting methods have a long history for providing efficient solution methods for a very large variety of problems modelled by partial differential equations and inequalities. Among the many applications of operator-splitting methods (not necessarily differential equations related) let us mention low rank approximation [80], stochastic differential equations, in finance and elsewhere [47, 90], image processing [50, 47], high-frequency wave propagation [46], travel-time tomography in seismic applications [45]. Implicit or explicit alternating direction methods [66, 99, 47] are particular operator-splitting methods. For a more complete discussion of operator-splitting methods and their many applications, we refer the interested readers to [47] and the references therein. To the best of our knowledge, the first publication making use of an operator-splitting method for the solution of a fully nonlinear elliptic problem is the celebrated article [6] by Y. Brenier and J.D. Benamou on the solution of the Monge-Kantorovich optimal mass transfer problem by the alternating direction methods of multipli-

ers (ADMM), a particular operator-splitting method. The solution of problem (3.1) by another ADMM algorithm is discussed in [26, 28, 29, 31, 44, 47].

To the best of our knowledge, the methodology discussed in the following sections is one of the very few able to handle unstructured meshes and domains Ω with curved boundary, using continuous piecewise affine approximations, while preserving optimal, or nearly optimal, convergence orders for the approximation errors. Preliminary promising results suggest that this methodology can be generalized to three dimensional problems, obstacle problems for the Monge-Ampère operator (like those discussed in [103]) and to the Gaussian curvature equation $\det \mathbf{D}^2 u = K \left(1 + |\nabla u|^2\right)^{1+d/2}$, with $d = 2$ or 3, K (the given curvature) being a positive function. These work will be discussed in following chapters.

3.2 A divergence formulation of problem (3.1) and an associated initial value problem

Let us denote by $\mathrm{cof}(\mathbf{D}^2 u)$ the matrix valued function $\begin{pmatrix} \frac{\partial^2 u}{\partial x_2^2} & -\frac{\partial^2 u}{\partial x_1 \partial x_2} \\ -\frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_1^2} \end{pmatrix}$. One can easily show that problem (3.1) is equivalent to

$$\begin{cases} -\nabla \cdot (\mathrm{cof}(\mathbf{D}^2 u) \nabla u) + 2f = 0 \text{ in } \Omega\,, \\ u = g \text{ on } \partial\Omega\,. \end{cases} \tag{3.2}$$

Similarly, one can also easily show that (3.2) characterizes formally u as being either a minimizer or a maximizer of the functional I over the space V_g where

$$I(v) = \int_\Omega \left(\mathrm{cof}\left(\mathbf{D}^2 v\right)\right) \nabla v \cdot \nabla v \, dx + 6 \int_\Omega f v \, dx\,,$$

and

$$V_g = \{v | v \text{ smooth}, v = g \text{ on } \partial\Omega\}.$$

Assume that u is solution of problem (3.1), (3.2). Since the symmetric matrix-valued functions $u \to \mathbf{D}^2 u$ and $u \to \text{cof}(\mathbf{D}^2 u)$ are either point-wise positive definite or negative definite in the neighborhood of u, one can easily show that functional I is either convex or concave in the neighborhood of u, justifying using a well-initialized descent type method to compute the solutions of problem (3.1), (3.2); this can be achieved via the time integration of an initial value problem associated with problem (3.2). To partly overcome the nonlinear coupling between u and its second order derivatives we introduce a matrix-valued function $\mathbf{p}$ verifying the linear relation $\mathbf{p} = \mathbf{D}^2 u$. Problem (3.2) is clearly equivalent to the following system of partial differential equations:

$$\begin{cases} \begin{cases} -\nabla \cdot (\text{cof}(\mathbf{p}) \nabla u) + 2f = 0 \text{ in } \Omega\,, \\ u = g \text{ on } \partial\Omega\,, \end{cases} \\ \mathbf{p} - \mathbf{D}^2 u = \mathbf{0}\,. \end{cases} \tag{3.3}$$

To handle those situations where $\inf_{x \in \Omega} f(x) = 0$, or for the solution of obstacle problems for the Monge-Ampère operator (as those considered in [103]), we found that one is on the safe side if one considers the following variant of problem (3.3), obtained by regularization:

$$\begin{cases} \begin{cases} -\nabla \cdot [(\varepsilon \mathbf{I} + \text{cof}(\mathbf{p}))\, \nabla u] + 2f = 0 \text{ in } \Omega\,, \\ u = g \text{ on } \partial\Omega\,, \end{cases} \\ \mathbf{p} - \mathbf{D}^2 u = \mathbf{0}\,, \end{cases} \tag{3.4}$$

where ε is a small positive number. In order to solve problem (3.4), we associate with it the following initial value problem (*flow* in the dynamical system terminology):

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}) \right) \nabla u \right] + 2f = 0 \text{ in } \Omega \times (0, +\infty), \\ u = g \text{ on } \partial\Omega \times (0, +\infty), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \left(\mathbf{p} - \mathbf{D}^2 u \right) = \mathbf{0} \text{ in } \Omega \times (0, +\infty), \\ u(0) = u_0, \ \mathbf{p}(0) = \mathbf{p}_0, \end{cases} \tag{3.5}$$

with γ a positive constant (above and below, $\phi(t)$ denotes the function $x \to \phi(x,t)$).

Before discussing (in Section 3.3) the time discretization of problem (3.5), we will address two important issues, namely: (i) The choice of γ, and (ii) the choice of u_0 and $\mathbf{p}_0$. Concerning γ, the idea is to pick a value so that $\mathbf{p}(t)$ evolves in time roughly at the same speed than $u(t)$. Taking advantage of the fact that $\mathbf{p}$ and $\mathrm{cof}(\mathbf{p})$ have the same eigenvalues, we suggest taking

$$\gamma = \beta \lambda_0 \left(\varepsilon + \sqrt{\alpha} \right),$$

where λ_0 is the smallest eigenvalue of operator $-\nabla^2$ in $H_0^1(\Omega)$, α is the lower bound of function f, and β is a constant of the order of 1. Assuming that we are looking for the convex solutions of problem (3.1), several possibilities (which do not exclude each other, in general) do exist in order to force this convexity property. The simplest one is a proper choice of u_0 and $\mathbf{p}_0$ in (3.5). Following the discussion in [43, 44], we suggest taking for u_0 the solution of the following Poisson problem

$$\begin{cases} \nabla^2 u_0 = 2\lambda\sqrt{f} \text{ in } \Omega\,, \\ u_0 = g \text{ on } \partial\Omega\,, \end{cases} \tag{3.6}$$

with $\lambda(>0)$ of the order of 1. Concerning $\mathbf{p}_0$, an obvious choice is

$$\mathbf{p}_0 = \mathbf{D}^2 u_0, \tag{3.7}$$

a simpler alternative being $\mathbf{p}_0 = \lambda\sqrt{f}\,\mathbf{I}$.

3.3 Time discretization by operator-splitting method

The structure of system (3.5) suggests using operator-splitting for its time-discretization. Among the many possible operator-splitting schemes (see, e.g., [47] for further information on operator-splitting methods) we advocate the particular Lie scheme described below, where $\Delta t(>0)$ is a time-discretization step and $t^n = n\Delta t$:

$$u^0 = u_0\,,\ \mathbf{p}^0 = \mathbf{p}_0\,. \tag{3.8}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \left\{u^{n+1/2}, \mathbf{p}^{n+1/2}\right\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

Fractional Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot [(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n))\, \nabla u] + 2f = 0 \text{ in } \Omega \times (t^n, t^{n+1}) \,, \\ u = g \text{ on } \partial\Omega \times (t^n, t^{n+1}) \,, \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}) \,, \\ u(t^n) = u^n \,,\ \mathbf{p}(t^n) = \mathbf{p}^n \,, \end{cases} \tag{3.9}$$

and set

$$u^{n+1/2} = u(t^{n+1}) \,,\ \mathbf{p}^{n+1/2} = \mathbf{p}^n \,.$$

Fractional Step 2:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}) \,, \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/2} \text{ in } \Omega \times (t^n, t^{n+1}) \,, \\ u(t^n) = u^{n+1/2} \,,\ \mathbf{p}(t^n) = \mathbf{p}^n \,, \end{cases} \tag{3.10}$$

and set

$$u^{n+1} = u(t^{n+1}) \,,\ \mathbf{p}^{n+1} = \mathbf{p}(t^{n+1}) \,.$$

Scheme (3.8)-(4.14) is first-order accurate at best, and semi-constructive since we still have to solve the sub-initial value problems (4.13) and (4.14). There is no difficulty with (4.14) since it has a closed form solution. To time-discretize (4.13), we advocate just one step of the backward Euler scheme. The resulting scheme (of the *Markchuk-Yanenko* type) reads as follows (using a more compact notation):

$$u^0 = u_0\,,\ \mathbf{p}^0 = \mathbf{p}_0\,. \tag{3.11}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1}-u^n}{\Delta t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n) \right) \nabla u^{n+1} \right] + 2f = 0 \text{ in } \Omega\,, \\ u^{n+1} = g \text{ on } \partial\Omega\,, \end{cases} \tag{3.12}$$

$$\mathbf{p}^{n+1} = e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1}\,. \tag{3.13}$$

If the matrix-valued function $\mathbf{p}^n$ is pointwise positive semi-definite, then (3.12) is a formally well-posed elliptic boundary value problem. In practice, in order to force the positive semi-definiteness of matrix $\mathbf{p}^n$, we complete (3.13) by a (kind of) projection on the cone of the positive semi-definite symmetric matrices (see Section 3.4.5 for details).

Remark 3.3.1. *As already mentioned, (3.11)-(3.13) is not the only operator-splitting scheme which can be used for the discretization of system (3.5). We can use for example*

$$u^0 = u_0\,,\ \mathbf{p}^0 = \mathbf{p}_0\,. \tag{3.14}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n) \right) \nabla u^{n+1/2} \right] = 0 \ \textit{in}\ \Omega\,, \\ u^{n+1/2} = g \ \textit{on}\ \partial\Omega\,, \end{cases} \tag{3.15}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right)\,, \tag{3.16}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\Delta t} + 2f = 0\,, \tag{3.17}$$

where in (3.16), P_+ is a projection operator on the convex cone of the positive

semi-definite symmetric matrices(see Section 3.4.5 for details).

3.4 On the finite element implementation of the operator-splitting schemes

3.4.1 Synopsis

The equivalent divergence formulations (3.2) and (3.3) of problem (3.1) strongly suggest employing space approximations based on variational principles. To achieve that goal, we are going to use finite element spaces consisting of functions which are globally continuous and piecewise affine on triangulations of Ω. As in, e.g., [15, 44] (see also the references therein), we are going to use a mixed finite element method, relying basically on the same finite dimensional spaces to approximate u, its three second order derivatives, and the entries of the matrix-valued function $\mathbf{p}$.

3.4.2 The basic finite element spaces

We follow the presentations in [42, 29, 15, 44]: Assuming that Ω is a polygonal domain of $\mathbb{R}^2$ (or has been approximated by such a domain), we introduce a family $(\mathcal{T}_h)_h$ of triangulations of Ω, like the ones in Figure 3.1; usually, one denotes by h the length of the largest edge(s) of $\mathcal{T}_h$.

The first finite element space we introduce is the finite dimensional space V_h defined by

$$V_h = \left\{ v | v \in C^0\left(\bar{\Omega}\right), v|_T \in P_1, \forall T \in \mathcal{T}_h \right\}, \tag{3.18}$$

where P_1 is the space of the polynomials of two variables of degree ≤ 1. Let

us denote by Σ_h the set of the vertices of the triangles of $\mathcal{T}_h$; we have then $\Sigma_h = \{Q_j\}_{j=1}^{N_h}$. Next, we associate with each vertex Q_j the (shape) function w_j, uniquely defined by:

$$\begin{cases} w_j \in V_h \,, \\ w_j(Q_j) = 1 \,, \\ w_j(Q_k) = 0, \forall k = 1, \ldots, N_h, k \neq j \,. \end{cases}$$

The set $\mathcal{B}_h = \{w_j\}_{j=1}^{N_h}$ is a vector basis of the space V_h; it verifies

$$v = \sum_{j=1}^{N_h} v(Q_j) w_j, \forall v \in V_h \,,$$

implying that dim $V_h = N_h$. We observe that the support of the basis function w_j is the union of those triangles of $\mathcal{T}_h$ which have Q_j as a common vertex.

Assuming that $g \in C^0(\partial\Omega)$, we define V_{gh}, an affine subspace of V_h, by

$$V_{gh} = \{v | v \in V_h, v(Q_j) = g(Q_j), \forall Q_j \in \Sigma_h \cap \partial\Omega\} \,.$$

Note that if $g = 0$, then $V_{gh} = V_{0h}$, where

$$V_{0h} = \{v | v \in V_h, v = 0 \text{ on } \partial\Omega\} \left(= V_h \cap H_0^1(\Omega)\right) \,.$$

Following [15, 44] we advocate the following vector space

$$\mathbf{Q}_{0h} = \left\{ \mathbf{q} | \mathbf{q} = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix} \in (V_{0h})^{2\times 2}, q_{12} = q_{21} \right\} \tag{3.19}$$

to approximate the matrix-valued function $\mathbf{p}$. Indeed, as shown in [15, 44], good numerical results were obtained approximating $\mathbf{p}$ in $\mathbf{Q}_{0h}$. However, unlike the

collocation approach used in [15, 44], the divergence form approach associated with formulations (3.2) and (3.3) requires an approximation of $\mathbf{p}$ which does not vanish on the whole $\partial\Omega$; this issue associated with the boundary conditions to impose on $\mathbf{p}$ will be discussed in Section 3.4.3.

In Section 3.4.3, just below, we are going to address the approximation of $\partial^2 u/\partial x_1^2, \partial^2 u/\partial x_2^2$ and $\partial^2 u/\partial x_1 \partial x_2$, a most important issue, indeed.

3.4.3 Finite element approximation of the three second order derivatives

In the context of the numerical solution of the Monge-Ampère problem (3.1), one has discussed in, e.g., [15, 44] the finite element approximation of $\partial^2 u/\partial x_1^2, \partial^2 u/\partial x_2^2$ and $\partial^2 u/\partial x_1 \partial x_2$. In the following sub-sections we will start our discussion by assuming that the discrete analogues of the second order derivatives are taken in V_{0h}, which is consistent with approximating $\mathbf{p}$ by $\mathbf{p}_h \in \mathbf{Q}_{0h}$, with $\mathbf{Q}_{0h}$ defined by (3.19).

Approximating the second order derivatives in V_{0h}

Let us consider a function $\psi \in H^2(\Omega)$; it follows then from the divergence theorem that $\forall \phi \in H_0^1(\Omega)$, one has

$$\begin{cases} \int_\Omega \frac{\partial^2 \psi}{\partial x_1^2} \phi dx = -\int_\Omega \frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_1} dx, \\ \int_\Omega \frac{\partial^2 \psi}{\partial x_2^2} \phi dx = -\int_\Omega \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_2} dx, \\ \int_\Omega \frac{\partial^2 \psi}{\partial x_1 \partial x_2} \phi dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_2} + \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right] dx. \end{cases} \tag{3.20}$$

Let us consider now a function $v \in V_h$; inspired by the relations (3.20), we define D_{11h}^2, D_{22h}^2 and $D_{12h}^2 (= D_{21h}^2)$, the respective approximations of $\frac{\partial^2}{\partial x_1^2}, \frac{\partial^2}{\partial x_2^2}$

and $\frac{\partial^2}{\partial x_1 \partial x_2}$, by

$$\begin{cases} \forall i = 1, 2, D^2_{iih}(v) \in V_{0h}, \\ \int_\Omega D^2_{iih}(v) w dx = -\int_\Omega \frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_i} dx, \forall w \in V_{0h}, \end{cases} \tag{3.21}$$

$$\begin{cases} D^2_{12h}(v) \in V_{0h}, \\ \int_\Omega D^2_{12h}(v) w dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_1} \frac{\partial w}{\partial x_2} + \frac{\partial v}{\partial x_2} \frac{\partial w}{\partial x_1} \right] dx, \forall w \in V_{0h}. \end{cases} \tag{3.22}$$

The above discrete variational problems are well-posed.

Remark 3.4.1. *The numerical experiments reported in [15] show that, in general, the approximate second order derivatives computed via (3.21), (3.22) have poor convergence properties for finite element spaces like V_h defined by (3.18). Two possible cures are: (i) Use continuous approximations of order higher than one (piecewise quadratic for example), (ii) Combine with a regularization procedure (as done in [15]). The above nuisance disappears if Ω is a rectangle and that one uses uniform triangulations, like the one in Figure 3.1(a); in such a case the approximations defined by (3.21) and (3.22) are second order accurate for interior points, as are the related approximate solutions of problem (3.1), obtained by the collocation method discussed in [15, 44].*

Remark 3.4.2. *As mentioned above, the functions $D^2_{ijh}(v)$ are uniquely defined by relations (3.21) and (3.22). However, in order to simplify the computation of the above discrete second order partial derivatives, we will use the trapezoidal rule to evaluate the integrals in the left-hand sides of (3.21) and (3.22). Owing to their practical importance, let us detail these calculations:*

1. *Let us denote by Σ_{0h} the set of the vertices of the triangles of the triangulation $\mathcal{T}_h$ which are not located on $\partial\Omega$. We assume that*

$$\Sigma_{0h} = \{Q_k\}_{k=1}^{N_{0h}},$$

implying that $\dim V_{0h} = N_{0h}$.

2. *If* $Q_k \in \Sigma_h$, *let us denote by* ω_k *the polygonal union of those triangles of* $\mathcal{T}_h$ *which have* Q_k *as a common vertex, and by* $|\omega_k|$ *the area of* ω_k. *We clearly have:*

$$\textit{support of } w_k = \bar{\omega}_k.$$

3. *Applying the trapezoidal rule to the integrals in the left hand side of (3.21) and (3.22) we obtain:*

$$\begin{cases} \forall i = 1, 2, D^2_{iih}(v) \in V_{0h}, \\ D^2_{iih}(v)(Q_k) = -\frac{3}{|\omega_k|} \int_{\omega_k} \frac{\partial v}{\partial x_i} \frac{\partial w_k}{\partial x_i}, \forall k = 1, \ldots, N_{0h}, \end{cases} \tag{3.23}$$

$$\begin{cases} D^2_{12h}(v) \,(= D^2_{21h}(v)) \in V_{0h}, \\ D^2_{12h}(v)(Q_k) = -\frac{3}{2|\omega_k|} \int_{\omega_k} \left[\frac{\partial v}{\partial x_1} \frac{\partial w_k}{\partial x_2} + \frac{\partial v}{\partial x_2} \frac{\partial w_k}{\partial x_1} \right] dx, \forall k = 1, \ldots, N_{0h}. \end{cases} \tag{3.24}$$

Computing the integrals in the right hand sides of (3.23) and (3.24) is quite simple since the first order derivatives of v *and* w_k *are piecewise constant, and* ω_k *is the union of a small number of triangles in general (6 triangles for the triangulation in Figure 3.1(a)).*

Remark 3.4.3. *Suppose that* $\Omega = (0,1)^2$ *and that the triangulation* $\mathcal{T}_h$ *is of the same type as the one in Figure 3.1(a). Suppose that* $h = \frac{1}{I+1}, I$ *being an integer greater than 1. In this particular case, the sets* Σ_h *and* Σ_{0h} *are given by*

$$\begin{cases} \Sigma_h = \{Q_{ij} | Q_{ij} = \{ih, jh\}, 0 \le i, j \le I+1\}, \\ \Sigma_{0h} = \{Q_{ij} | Q_{ij} = \{ih, jh\}, 1 \le i, j \le I\}, \end{cases}$$

implying that $N_h = (I+1)^2$ *and* $N_{0h} = I^2$. *It follows then from relation (3.23) and (3.24) that (with obvious notation):*

$$D^2_{11h}(v)(Q_{ij}) = \frac{v_{i+1j} + v_{i-1j} - 2v_{ij}}{h^2}, \tag{3.25}$$

$$D^2_{22h}(Q_{ij}) = \frac{v_{ij+1} + v_{ij-1} - 2v_{ij}}{h^2}, \tag{3.26}$$

$$D^2_{12h}(v)(Q_{ij}) = \tag{3.27}$$

$$\frac{(v_{i+1j+1} + v_{i-1j-1} + 2v_{ij}) - (v_{i+1j} + v_{i-1j} + v_{ij+1} + v_{ij-1})}{2h^2}. \tag{3.28}$$

Relations (3.25)-(3.28) are exact for the two variable polynomials of degree ≤ 2. *Also, as expected,*

$$D^2_{11h}(v)(Q_{ij}) + D^2_{22h}(v)(Q_{ij}) = \frac{v_{i+1j} + v_{i-1j} + v_{ij+1} + v_{ij-1} - 4v_{ij}}{h^2};$$

we have recovered thus the 5-point discretization formula for the finite difference approximation of the Laplace operator.

A smoothing procedure for the approximation of the second order derivatives

In [15, 44], one solves problem (3.1) numerically using a least-squares/collocation method relying on the approximations (3.23), (3.24) of the second order derivatives. As reported in the two references above the convergence properties of the approximate solutions are highly dependent on the triangulations, triangulations like the one shown in Figure 3.1(a) (resp., 3.1(b)) having very good (resp., very poor) convergence properties. Albeit quite different, the solution methods discussed in this article suffer from the same drawback. To overcome the above difficulty and allow a large variety of triangulations of Ω, one advocated in [15, 44]

a regularization procedure where one replaces (3.22) by

$$\begin{cases} \forall i = 1,2,\ j = 1,2,\ D^2_{ijh}(v) \in V_{0h}, \\ \displaystyle\int_\Omega D^2_{ijh}(v) w\, dx + C \sum_{T \in \mathcal{T}_h} |T| \int_T \nabla D^2_{ijh}(v) \cdot \nabla w\, dx \\ \qquad = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j} \frac{\partial w}{\partial x_i}\, dx \right], \forall w \in V_{0h}, \end{cases} \tag{3.29}$$

where in (3.29), C is a positive constant of the order of 1 and $|T|$ = measure of T.

Remark 3.4.4. *Denote by $h_{\min}$ the length of the smallest edge(s) of $\mathcal{T}_h$; if the ratio $h/h_{\min} \approx 1$, it is reasonable to replace (3.29) by*

$$\begin{cases} \forall i = 1,2,\ j = 1,2,\ D^2_{ijh}(v) \in V_{0h}, \\ \displaystyle\int_\Omega D^2_{ijh}(v) w\, dx + \varepsilon_1 \int_\Omega \nabla D^2_{ijh}(v) \cdot \nabla w\, dx \\ \qquad = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j} \frac{\partial w}{\partial x_i}\, dx \right], \forall w \in V_{0h}, \end{cases} \tag{3.30}$$

with $\varepsilon_1 (> 0)$ of the order of h^2. The numerical results reported in [15], obtained with $C = 1$, show that the regularization procedure associated with (3.30) produces accurate results, for a variety of domains Ω (including some with curved boundary) and triangulations, when combined with the least-squares/collocation method discussed there. However, numerical experiments show that the finite element realization of the operator-splitting scheme (3.11)-(3.13), relying on (3.21), (3.22) or (3.23), (3.24) and (3.29), (3.30) to approximate the second order derivatives, leads to approximate solutions which do not converge to the exact ones as $h \to 0$. Possible cures will be discussed in Section 3.4.3 just below.

On the boundary conditions to impose on the discrete second order derivatives

Unlike the collocation method discussed in [15], the values taken on $\partial\Omega$ by the discrete second order derivatives affect the approximate solution. From that point of view, enforcing these discrete derivatives to vanish on $\partial\Omega$ leads to large approximation errors on the solutions of problem (3.12), a consequence of the poor approximation of $\mathrm{cof}(\mathbf{p}^n)$ in the neighborhood of $\partial\Omega$. To overcome this difficulty several approaches can be considered. The *first* one is to observe that the divergence theorem implies:

$$\begin{cases} \forall i,j=1,2, \forall v \in H^2(\Omega), \\ \int_\Omega \frac{\partial^2 v}{\partial x_i \partial x_j} w dx = -\frac{1}{2}\int_\Omega \left[\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w}{\partial x_i}\right] dx \\ \qquad +\frac{1}{2}\int_{\partial\Omega}\left[\frac{\partial v}{\partial x_i}n_j + \frac{\partial v}{\partial x_j}n_i\right] w d(\partial\Omega), \forall w \in H^1(\Omega), \end{cases} \tag{3.31}$$

where n_1 and n_2 are the two components of $\mathbf{n}$, the unit outward normal vector at $\partial\Omega$. Following Section 3.4.3 and 3.4.3 we approximate (3.31) by

$$\begin{cases} \forall i,j=1,2, \forall v \in V_h, D^2_{ijh}(v) \in V_h \text{ and} \\ \int_\Omega D^2_{ijh}(v) w dx = -\frac{1}{2}\int_\Omega \left[\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w}{\partial x_i}\right] dx \\ \qquad +\frac{1}{2}\int_{\partial\Omega}\left[\left(\frac{\partial v}{\partial x_i}\right)_h n_j + \left(\frac{\partial v}{\partial x_j}\right)_h n_i\right] w d(\partial\Omega), \forall w \in V_h, \end{cases} \tag{3.32}$$

or by its regularized variant

$$\begin{cases} \forall i,j=1,2, \forall v \in V_h, D^2_{ijh}(v) \in V_h \text{ and} \\ C\sum_{T\in\mathcal{T}_h}|T|\int_T \nabla D^2_{ijh}(v)\cdot\nabla w dx + \int_\Omega D^2_{ijh}(v) w dx = -\frac{1}{2}\int_\Omega \left[\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w}{\partial x_i}\right] dx \\ \qquad +\frac{1}{2}\int_{\partial\Omega}\left[\left(\frac{\partial v}{\partial x_i}\right)_h n_j + \left(\frac{\partial v}{\partial x_j}\right)_h n_i\right] w d(\partial\Omega), \forall w \in V_h, \end{cases} \tag{3.33}$$

The finite dimensional problems (3.32) and (3.33) are both well posed and Remark 3.4.4 still applies here. The simplest choice for $\left(\frac{\partial v}{\partial x_i}\right)_h$ is obviously the first order accurate one given by $\left(\frac{\partial v}{\partial x_i}\right)_h = \frac{\partial v}{\partial x_i}|_{\partial\Omega}$, making the computation of the boundary integrals in (3.32), (3.33) very easy (since the above derivatives are piecewise constant). More sophisticated (and accurate) approximations of these derivative traces do exist.

Suppose that as in Section 3.4.3 (whose notation we keep) we use the trapezoidal rule to compute $\int_\Omega D^2_{ijh}(v)w dx$; the associated variants of (3.32) and (3.33) are

$$\begin{cases} \forall i,j=1,2, D^2_{ijh}(v) = \sum_{k=1}^{N_h} D^2_{ijh}(v)(Q_k)w_k \text{ with} \\ D^2_{ijh}(v)(Q_k) = -\frac{3}{2|\omega_k|}\int_{\omega_k}\left[\frac{\partial v}{\partial x_i}\frac{\partial w_k}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w_k}{\partial x_i}\right]dx \\ \qquad\qquad +\frac{3}{2|\omega_k|}\int_{\partial\Omega\cap\partial\omega_k}\left[\left(\frac{\partial v}{\partial x_i}\right)_h n_j + \left(\frac{\partial v}{\partial x_j}\right)_h n_i\right] w_k d(\partial\Omega), \\ \forall k=1,...,N_h, \end{cases} \tag{3.34}$$

and

$$\begin{cases} \forall i,j=1,2, D^2_{ijh}(v) = \sum_{k=1}^{N_h} D^2_{ijh}(v)(Q_k)w_k \text{ with} \\ C\sum_{T\in\mathcal{T}_h^k}|T|\int_T \nabla D^2_{ijh}(v)\cdot\nabla w_k dx + \frac{|\omega_k|}{3}D^2_{ijh}(v)(Q_k) = \\ \qquad -\frac{1}{2}\int_{\omega_k}\left[\frac{\partial v}{\partial x_i}\frac{\partial w_k}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w_k}{\partial x_i}\right]dx + \frac{1}{2}\int_{\partial\Omega\cap\partial\omega_k}\left[\left(\frac{\partial v}{\partial x_i}\right)_h n_j + \left(\frac{\partial v}{\partial x_j}\right)_h n_i\right] w_k d(\partial\Omega), \\ \forall k=1,...,N_h, \end{cases} \tag{3.35}$$

respectively. In (3.35), $\mathcal{T}_h^k$ denotes the subset of $\mathcal{T}_h$ consisting of the triangles which have Q_k as a vertex.

In the *second* approach we considered, we imposed (approximately) the condition $\frac{\partial}{\partial\mathbf{n}}D^2_{ijh}(v) = 0$ on $\partial\Omega$ by requiring

$$\int_{\partial\Omega\cap\partial\omega_k}\frac{\partial}{\partial\mathbf{n}}D^2_{ijh}(v)w_k d(\partial\Omega) = 0, \forall k = N_{0h}+1,...,N_h. \tag{3.36}$$

This leads us to compute $D^2_{ijh}(v)$ via the solution of the following linear systems:

$$\begin{cases} \forall i,j=1,2, \forall v \in V_h, D^2_{ijh}(v) = \sum_{k=1}^{N_h} D^2_{ijh}(v)(Q_k)w_k, \text{ with} \\ \int_{\omega_k} D^2_{ijh}(v) w_k dx = -\frac{1}{2}\int_{\omega_k} \left[\frac{\partial v}{\partial x_i}\frac{\partial w_k}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w_k}{\partial x_i} \right] dx, \forall k = 1, ..., N_{0h}, \\ \int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0, \forall k = N_{0h}+1, ..., N_h, \end{cases} \tag{3.37}$$

(*non regularized* case) and

$$\begin{cases} \forall i,j=1,2, \forall v \in V_h, D^2_{ijh}(v) = \sum_{k=1}^{N_h} D^2_{ijh}(v)(Q_k)w_k, \text{ with} \\ C\sum_{T\in\mathcal{T}_h^k} |T| \int_T \nabla D^2_{ijh}(v) \cdot \nabla w_k dx + \int_{\omega_k} D^2_{ijh}(v) w_k dx = \\ \qquad -\frac{1}{2}\int_{\omega_k} \left[\frac{\partial v}{\partial x_i}\frac{\partial w_k}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w_k}{\partial x_i} \right] dx, \forall k = 1, ..., N_{0h}, \\ \int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0, \forall k = N_{0h}+1, ..., N_h, \end{cases} \tag{3.38}$$

(*regularized* case). Deriving the variants of (3.37), (3.38) obtained by applying the trapezoidal rule to the computation of $\int_\Omega D^2_{ijh}(v) w_k dx$ is quite easy and will not be further detailed here.

Remark 3.4.5. *The relations* $\int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0, \forall k = N_{0h}+1, ..., N_h$, *involve clearly two triangles of* $\mathcal{T}_h$ *at most (the four corners of the triangulation of Figure 3.1(c) involve one triangle only, making things simpler). Let us consider the two triangle situation and denote by* T_1^k *and* T_2^k *these two triangles. We have then*

$$2\int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = \sum_{l=1}^{2} \nabla D^2_{ijh}(v)|_{T_l^k} \cdot \mathbf{n}_l^k |\partial\Omega \cap \partial T_l^k| = 0, \tag{3.39}$$

where in (3.39), $\mathbf{n}_l^k$ *denotes the unit outward normal vector at* ∂T_l^k *and* $|\partial\Omega\cap\partial T_l^k|$ *the length of the edge* $\partial\Omega \cap \partial T_l^k$. *It is very simple to modify (3.39) if relation* $\int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0$ *involves only one triangle: just take* $T_1^k = T_2^k$.

Remark 3.4.6. *In practice, instead of using*

$$\int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0, \forall k = N_{0h}+1, ..., N_h,$$

to force (approximately) the condition $\frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) = 0$ *on* $\partial\Omega$*, we employed in this article*

$$\int_{\omega_k} \nabla D^2_{ijh}(u^{n+1/2}) \cdot \nabla w_k \, dx = 0, \tag{3.40}$$

a condition easier to implement. Let us (try to) justify this alternate approximate homogeneous Neumann condition: We have

$$\int_{\partial\omega_k} \frac{\partial \psi}{\partial \mathbf{n}} w_k d(\partial\Omega) = \int_{\omega_k} \nabla^2\psi w_k dx + \int_{\omega_k} \nabla\psi \cdot \nabla w_k dx, \forall \psi \in H^2(\Omega). \tag{3.41}$$

Suppose now that ψ *is* harmonic *(that is* $\nabla^2\psi = 0$*); then, relation (3.41) reduces to*

$$\int_{\partial\omega_k} \frac{\partial \psi}{\partial \mathbf{n}} w_k d(\partial\Omega) = \int_{\omega_k} \nabla\psi \cdot \nabla w_k dx. \tag{3.42}$$

Albeit the function D^2_{ijh} *is* piecewise harmonic *only (since it is a piecewise affine function), we use (3.42) to impose, approximately, the Neumann condition* $\frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) = 0$ *on* $\partial\Omega$*, explaining where (3.40) comes from. It is clear that replacing relation* $\int_{\partial\Omega\cap\partial\omega_k} \frac{\partial}{\partial \mathbf{n}} D^2_{ijh}(v) w_k d(\partial\Omega) = 0$ *by (3.40) is a typical example of* variational crime *(in the sense of [109]). However, the numerical experiments reported in Section 3.6 show that the results obtained with (3.40) are as accurate as those obtained using (3.35) (if no more).*

3.4.4 Finite element implementation of the operator-splitting schemes (3.11)-(3.13) and (3.14)-(3.17)

Let us recall that the set Σ_{0h} of the vertices of $\mathcal{T}_h$ interior to Ω has been ordered so that $\Sigma_{0h} = \{Q_k\}_{k=1}^{N_{0h}}$ and $\Sigma_h = \Sigma_{0h} \cup \{Q_k\}_{k=N_{0h}+1}^{N_h}$. In the sequel, the space

$\{\mathbf{q}|\mathbf{q} \in (V_h)^{2\times 2}, \mathbf{q} = \mathbf{q}^T\}$ will be denoted by $\mathbf{Q}_h$.

Implementation of scheme (3.11)-(3.13)

A fully discrete analogue of scheme (3.11)-(3.13) reads as

$$u^0 = u_{0h} \left(\in V_{gh}\right), \mathbf{p}^0 = \mathbf{p}_{0h} \left(\in \mathbf{Q}_h\right). \tag{3.43}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows:

Solve

$$\begin{cases} u^{n+1} \in V_{gh}, \\ \int_\Omega u^{n+1} v dx + \Delta t \int_\Omega \left(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)\right) \nabla u^{n+1} \cdot \nabla v dx \\ \qquad = \int_\Omega u^n v dx - 2\Delta t \int_\Omega f_h v dx, \forall v \in V_{0h}, \end{cases} \tag{3.44}$$

and compute $\mathbf{p}^{n+1}$ via

$$\begin{cases} \mathbf{p}^{n+1}(Q_k) = e^{-\gamma \Delta t} \mathbf{p}^n(Q_k) + \left(1 - e^{-\gamma \Delta t}\right) \begin{pmatrix} D^2_{11h}\left(u^{n+1}\right)(Q_k) & D^2_{12h}\left(u^{n+1}\right)(Q_k) \\ D^2_{12h}\left(u^{n+1}\right)(Q_k) & D^2_{22h}\left(u^{n+1}\right)(Q_k) \end{pmatrix}, \\ \forall k = 1, \ldots, N_h, \end{cases} \tag{3.45}$$

where in (3.44), $f_h (> 0)$ is a continuous approximation of f, and where in (3.45), the discrete second order derivatives of u^{n+1} are computed using the methods discussed in Section 3.4.3.

The initialization of scheme (3.43)-(3.45) and the solution of the discrete variational problem (3.44) will be discussed in Section 4.6 and Section 3.4.4, respectively.

Implementation of scheme (3.14)-(3.17)

A fully discrete analogue of scheme (3.14)-(3.17) reads as

$$u^0 = u_{0h} \left(\in V_{gh}\right), \mathbf{p}^0 = \mathbf{p}_{0h} \left(\in \mathbf{Q}_h\right). \tag{3.46}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows:

1. Solve

$$\begin{cases} u^{n+1/2} \in V_{gh}, \\ \int_\Omega u^{n+1/2} v dx + \Delta t \int_\Omega \left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n)\right) \nabla u^{n+1/2} \cdot \nabla v dx \\ \qquad = \int_\Omega u^n v dx, \forall v \in V_{0h}. \end{cases} \tag{3.47}$$

2. Compute $\mathbf{p}^{n+1}$ and u^{n+1} via

$$\begin{cases} \mathbf{p}^{n+1}(Q_k) = e^{-\gamma \Delta t} \mathbf{p}^n(Q_k) \\ \quad + \left(1 - e^{-\gamma \Delta t}\right) \begin{pmatrix} D_{11h}^2 \left(u^{n+1/2}\right)(Q_k) & D_{12h}^2 \left(u^{n+1/2}\right)(Q_k) \\ D_{12h}^2 \left(u^{n+1/2}\right)(Q_k) & D_{22h}^2 \left(u^{n+1/2}\right)(Q_k) \end{pmatrix}, \\ \forall k = 1, \ldots, N_h, \end{cases} \tag{3.48}$$

$$\begin{cases} u^{n+1}(Q_k) = u^{n+1/2}(Q_k) - 2\Delta t f_h(Q_k), \\ \forall k = 1, \ldots, N_{0h}. \end{cases} \tag{3.49}$$

Initialization of schemes (3.43)-(3.45) and (3.46)-(3.49)

Our starting point will be problem (3.6) defining u_0. The most natural discrete analogue of problem (3.6) is given by

$$\begin{cases} u_{0h} \in V_{gh}, \\ \int_\Omega \nabla u_{0h} \cdot \nabla v dx = -2\lambda \int_\Omega \sqrt{f_h} v dx, \forall v \in V_{0h}. \end{cases} \tag{3.50}$$

From a practical point of view, we advocate using the trapezoidal rule to compute (approximately) the integral $\int_\Omega \sqrt{f_h} v dx$. We obtain then (using notation from previous sections):

$$\int_\Omega \sqrt{f_h} v dx \approx \frac{1}{3} \Sigma_{j=1}^{N_{0h}} |\omega_j| \sqrt{f_h(Q_j)} v(Q_j). \tag{3.51}$$

Suppose now that $\Omega = (0,1)^2$ and that the triangulation we employ is of the type of the one depicted in Figure 3.1(a). Denoting $u_{0h}(Q_{ij})$ by U_{ij}, the combination (3.50), (3.51) reduces to (with $Q_{ij} = \{ih, jh\}$):

$$\begin{cases} 4U_{ij} - U_{i+1j} - U_{i-1j} - U_{ij+1} - U_{ij-1} = -2\lambda h^2 \sqrt{f_h(Q_{ij})}, \\ 1 \leq i, j \leq I, \\ U_{kl} = g(Q_{kl}) \text{ if } Q_{kl} \in \partial\Omega. \end{cases} \tag{3.52}$$

Problem (3.52) can be solved by a fast Poisson solver.

The solution of (3.50) for more general domains Ω and triangulations $\mathcal{T}_h$ will be discussed in Section 3.4.4.

Once u_{0h} has been computed we use the methods discussed in Section 3.4.3 to

define $\mathbf{p}_{0h}$ by

$$\mathbf{p}_{0h} = \sum_{k=1}^{N_h} \begin{pmatrix} D_{11h}(u_{0h})(Q_k) & D_{12h}(u_{0h})(Q_k) \\ D_{12h}(u_{0h})(Q_k) & D_{22h}(u_{0h})(Q_k) \end{pmatrix} w_k. \tag{3.53}$$

An alternative to (3.53) is to define $\mathbf{p}_{0h}$ from $\mathbf{p}_0 = \lambda\sqrt{f}\mathbf{I}$, that is

$$\mathbf{p}_{0h} = \lambda \sum_{k=1}^{N_h} \sqrt{f_h(Q_k)}\mathbf{I} w_k, \tag{3.54}$$

a discrete analogue of $\mathbf{p}_0 = \lambda\sqrt{f}\mathbf{I}$.

On the solution of problems (3.44), (3.47) and of related linear variational problems

What follows is closely related to [42]. Problems (3.44), (3.47) and (3.50) are all particular cases of the following finite dimensional linear variational problem (of the Dirichlet type):

$$\begin{cases} \psi \in V_{gh}, \\ a\int_\Omega \psi\phi dx + \int_\Omega \mathbf{M}\nabla\psi \cdot \nabla\phi dx = \int_\Omega f^*\phi dx, \forall \phi \in V_{0h}, \end{cases} \tag{3.55}$$

where in (3.55), a is a non-negative constant, $\mathbf{M}$ is a piecewise affine uniformly positive definite symmetric matrix-valued function, and f^* is a continuous given function.

From the positivity of matrix $\mathbf{M}$, problem (3.55) has a unique solution. We will return on the matrix $\mathbf{M}$ positivity issue in Section 3.4.5.

Using the trapezoidal rule to approximate the first and third integrals in (3.55),

the above problem takes the following formulation:

$$\begin{cases} \psi \in V_{gh}, \\ \dfrac{a}{3}\displaystyle\sum_{l=1}^{N_{0h}} |\omega_l|\, \psi(Q_l)\phi(Q_l) + \int_\Omega \mathbf{M}\nabla\psi \cdot \nabla\phi dx \\ \qquad = \dfrac{1}{3}\displaystyle\sum_{l=1}^{N_{0h}} |\omega_l|\, f^*(Q_l)\phi(Q_l), \forall \phi \in V_{0h}. \end{cases} \tag{3.56}$$

The set $\{w_k\}_{k=1}^{N_{0h}}$ being a vector basis of V_{0h}, there is equivalence between (3.56) and

$$\begin{cases} \psi \in V_{gh}, \\ \frac{a}{3}\,|\omega_k|\,\psi(Q_k) + \int_\Omega \mathbf{M}\nabla\psi \cdot \nabla w_k dx = \frac{1}{3}\,|\omega_k|\, f^*(Q_k), \\ k = 1, ..., N_{0h}. \end{cases} \tag{3.57}$$

Problem (3.57) can be written as a linear system associated with a symmetric positive definite matrix. To derive this system we take advantage of the fact that

$$\psi = \sum_{l=1}^{N_{0h}} \psi(Q_l)w_l + \sum_{l=N_{0h}+1}^{N_h} g(Q_l)w_l. \tag{3.58}$$

Let us denote $\psi(Q_l)$ by ψ_l; vector $\{\psi_k\}_{k=1}^{N_{0h}}$ is clearly the solution to the following linear system:

$$\begin{cases} \frac{a}{3}\,|\omega_k|\,\psi_k + \sum_{l=1}^{N_{0h}} \left(\int_{\omega_k\cap\omega_l} \mathbf{M}\nabla w_k \cdot \nabla w_l dx\right)\psi_l \\ \qquad = \frac{1}{3}\,|\omega_k|\, f^*(Q_k) - \sum_{l=N_{0h}+1}^{N_h} \left(\int_{\omega_k\cap\omega_l} \mathbf{M}\nabla w_k \cdot \nabla w_l dx\right) g(Q_l), \\ k = 1, \ldots, N_{0h}. \end{cases} \tag{3.59}$$

Since, in practice, h is small compared to the diameter of Ω, the matrix associated with the linear system (3.59) is sparse. Moreover, since the symmetric matrix-valued function $\mathbf{M}$ is uniformly positive definite, the matrix associated with the linear system (3.59) is symmetric and positive definite, implying that one can solve (3.59) using a sparse Cholesky solver, or a diagonally preconditioned

conjugate gradient algorithm.

It follows from the above discussion that the main issue needing to be addressed is the computation of $\int_{\omega_k \cap \omega_l} \mathbf{M}\nabla w_k \cdot \nabla w_l dx$. Since $\mathbf{M}$ is piecewise affine, $\nabla w_k \cdot \nabla w_l$ is piecewise constant, and $\omega_k \cap \omega_l$ consisting of a small number of triangles (two at most if $k \neq l$), the above computation is conceptually simple. Indeed, let us consider a triangle $T = A_1A_2A_3$, the vertices A_1, A_2 and A_3 being oriented counter-clockwise. Suppose now that one wants to compute

$$\int_T \mathbf{M}\nabla\phi \cdot \nabla\theta dx, \tag{3.60}$$

where ϕ and θ are both real-valued functions affine over T, and $\mathbf{M}$ is a matrix-valued function affine over T. Since the two vectors $\nabla\phi$ and $\nabla\theta$ are constant over T, we have

$$\int_T \mathbf{M}\nabla\phi \cdot \nabla\theta dx = \frac{|T|}{3}\left(\sum_{i=1}^{3} \mathbf{M}(A_i)\right)\nabla\phi \cdot \nabla\theta. \tag{3.61}$$

For $i = 1, 2,$ and 3, let us uniquely define the affine function w_i by

$$\begin{cases} w_i \in P_1, \\ w_i(A_i) = 1, w_i(A_j) = 0 \text{ if } j \neq i. \end{cases} \tag{3.62}$$

Since $\phi = \sum_{i=1}^{3} \phi(A_i)w_i$ and $\theta = \sum_{i=1}^{3} \theta(A_i)w_i$, one has

$$\nabla\phi = \sum_{i=1}^{3} \phi(A_i)\nabla w_i \text{ and } \nabla\theta = \sum_{i=1}^{3} \theta(A_i)\nabla w_i. \tag{3.63}$$

Let a_i and b_i denote the two coordinates of vertex A_i. We can easily show that

$$\nabla w_1 = \frac{1}{2|T|}\begin{pmatrix} b_2 - b_3 \\ a_3 - a_2 \end{pmatrix}, \nabla w_2 = \frac{1}{2|T|}\begin{pmatrix} b_3 - b_1 \\ a_1 - a_3 \end{pmatrix}, \text{ and } \nabla w_3 = \frac{1}{2|T|}\begin{pmatrix} b_1 - b_2 \\ a_2 - a_1 \end{pmatrix}. \tag{3.64}$$

Combining (3.62), (3.63) and (3.64), we can easily compute the integrals in (3.59), and therefore those in (3.57) since, in practice, the integration sets are union of (a small number of) triangles of $\mathcal{T}_h$. These computations further simplify if $\mathcal{T}_h$ is a triangulation of the same type as those considered in Figure 3.1(a).

3.4.5 Enforcing the local positive semi-definiteness of p by eigenvalue projection

In this section, we describe a classical method to force the positive semi-definiteness of $\mathbf{p}$. Assume that $\mathbf{p}_i(=\mathbf{p}(Q_i))$ has the following diagonalization $\mathbf{p}_i = \mathbf{V}_i\mathbf{\Lambda}_i\mathbf{V}_i^{-1}$, where $\mathbf{\Lambda}_i = \begin{pmatrix} \lambda_i^1 & 0 \\ 0 & \lambda_i^2 \end{pmatrix}$, λ_i^1 and λ_i^2 being the eigenvalues of $\mathbf{p}_i$. Let us denote by $\mathbf{\Lambda}_i^+$ the matrix $\mathbf{\Lambda}_i^+ = \begin{pmatrix} \lambda_i^{1+} & 0 \\ 0 & \lambda_i^{2+} \end{pmatrix}$, where $\lambda_i^{k+} = \max(\lambda_i^k, 0)$, $\forall k = 1, 2$. In practice we use (with obvious notation) the following variant of (3.45) (resp., (3.48)) to update $\mathbf{p}^n$ in scheme (3.11)-(3.13) (resp., (3.14)-(3.17)):

$$\begin{cases} \forall i = 1, \ldots, N_h, \\ \mathbf{p}^{n+1/2}(Q_i) = e^{-\gamma\Delta t}\mathbf{p}^n(Q_i) + \left(1 - e^{-\gamma\Delta t}\right) \begin{pmatrix} D_{11h}^2\left(u^{n+1}\right)(Q_i) & D_{12h}^2\left(u^{n+1}\right)(Q_i) \\ D_{12h}^2\left(u^{n+1}\right)(Q_i) & D_{22h}^2\left(u^{n+1}\right)(Q_i) \end{pmatrix}, \\ \mathbf{p}^{n+1}(Q_i) = \mathbf{V}_i\mathbf{\Lambda}_i^+(\mathbf{p}^{n+1/2})\mathbf{V}_i^{-1}. \end{cases}$$

(resp.,

$$\begin{cases} \forall i = 1, \ldots, N_h, \\ \mathbf{p}^{n+1/2}(Q_i) = e^{-\gamma\Delta t}\mathbf{p}^n(Q_i) + \left(1 - e^{-\gamma\Delta t}\right) \begin{pmatrix} D_{11h}^2\left(u^{n+1/2}\right)(Q_i) & D_{12h}^2\left(u^{n+1/2}\right)(Q_i) \\ D_{12h}^2\left(u^{n+1/2}\right)(Q_i) & D_{22h}^2\left(u^{n+1/2}\right)(Q_i) \end{pmatrix}, \\ \mathbf{p}^{n+1}(Q_i) = \mathbf{V}_i\mathbf{\Lambda}_i^+(\mathbf{p}^{n+1/2})\mathbf{V}_i^{-1}). \end{cases}$$

3.5 A two-stage convergence acceleration strategy

In this section, we propose a simple strategy to speed up the convergence of schemes (3.11)-(3.13) and (3.14)-(3.17). Instead of (3.5), we consider the following initial value problem

$$\begin{cases} \begin{cases} -\frac{\partial(\nabla\cdot[(\varepsilon\mathbf{I}+\mathbf{M})\nabla u])}{\partial\tau} - \nabla\cdot\left[\left(\varepsilon\mathbf{I}+\operatorname{cof}(\mathbf{p})\right)\nabla u\right] + 2f = 0, \text{ in } \Omega\times(0,+\infty), \\ u = g \text{ on } \partial\Omega\times(0,+\infty), \end{cases} \\ \frac{\partial\mathbf{p}}{\partial\tau} + \gamma\left(\mathbf{p}-\mathbf{D}^2u\right) = \mathbf{0} \text{ in } \Omega\times(0,+\infty), \\ u(0) = u_1,\ \mathbf{p}(0) = \mathbf{p}_1, \end{cases} \tag{3.65}$$

where τ is an artificial time. In (3.65), $\mathbf{M}$ is a time independent matrix-valued function, which is supposed to be reasonably close to $\operatorname{cof}(\mathbf{D}^2u^*)$, u^* being the convex solution of problem (3.1) (assuming it exists).

We have the following theorem about the convergence order of a general form of (3.65):

Theorem 3.5.1. *Let u^* be the solution to*

$$Au + f = 0. \tag{3.66}$$

where A is a linear operator applied on u. Assume A can be discretized as a positive definite matrix with largest eigenvalue larger than 2. Starting from an initial condition u^0, solve

$$\frac{\partial u}{\partial t} + Au + f = 0 \tag{3.67}$$

or

$$\frac{\partial Mu}{\partial t} + Au + f = 0 \tag{3.68}$$

to steady state to get an approximation of u^*. Denote the discretization of M and A by $\mathbf{M}$ and $\mathbf{A}$ respectively. In (3.68), M is an approximation of A in the sense that the range of eigenvalues of $\mathbf{M}, \mathbf{A}$ are similar. If (3.67) and (3.68) are solved by forward Euler method, then the stability time restriction of (3.68) is more relaxable than (3.67), especially when $\mathbf{A}$ has large eigenvalues.

Proof. Using the forward Euler method, (3.67) can be discretized as

$$\frac{u^{n+1} - u^n}{\Delta t} + \mathbf{A}u^n + f = 0. \tag{3.69}$$

Using the condition u^* is the exact solution, (3.69) is equivalent to

$$\frac{(u^{n+1} - u^*) - (u^n - u^*)}{\Delta t} + \mathbf{A}(u^n - u^*) = 0 \tag{3.70}$$

which can be written as

$$e_1^{n+1} = (\mathbf{I} - \Delta t\mathbf{A})e_1^n. \tag{3.71}$$

$e_1^n = u^n - u^*$ is the error of u^n when solving (3.69) at time level t^n. Denote the largest and smallest eigenvalue of $\mathbf{A}$ by λ_{max} and λ_{min} respectively. Then to get a convergent scheme, we need $|1 - \Delta t\lambda_{max}| < 1$. So we need $\Delta t < \frac{2}{\lambda_{max}}$. The time restriction of (3.69) is of order $O(\frac{2}{\lambda_{max}})$. It can be very small when λ_{max} is large.

Similarly, (3.67) can be discretized as

$$\frac{\mathbf{M}u^{n+1} - \mathbf{M}u^n}{\Delta t} + \mathbf{A}u^n + f = 0. \tag{3.72}$$

and we have

$$e_2^{n+1} = (\mathbf{I} - \Delta t\mathbf{M}^{-1}\mathbf{A})e_2^n. \tag{3.73}$$

$e_1^n = u^n - u^*$ is the error of u^n when solving (3.70) at time level t^n. Denote the largest and smallest eigenvalue of $\mathbf{A}$ by α_{max} and α_{min} respectively. The stability

condition on the time step is $|1-\Delta t\alpha_{max}| < 1$. Since we assume $\mathbf{M}$ approximates $\mathbf{A}$ well, $\alpha_{min}, \alpha_{max}$ should be close to 1. So the time step restriction of (3.72) is of order $O(1)$. □

Remark 3.5.1. *Although Theorem 3.5.1 is about explicit scheme, in our experiment, this technique also provides extra stability for implicit scheme. We can use a much larger time step with faster convergence rate.*

Scheme (3.11)-(3.13) can be easily modified in order to accommodate (3.65): we just have to replace $\frac{u^{n+1}-u^n}{\Delta t}$ in (3.12) by

$$-\nabla \cdot \left[(\varepsilon \mathbf{I} + \mathbf{M}) \nabla \left(\frac{u^{n+1} - u^n}{\Delta \tau} \right) \right].$$

Compared to (3.12), the above preconditioning allows the use of a larger (pseudo) time step $\Delta\tau$, typically of the order of 1 if $\gamma \approx 1$, and u_1 and $\mathbf{p}_1$ are well-chosen. If convergence takes place, we expect that $\lim_{n\to+\infty} u^n = u^*$. To guarantee convergence, a proper choice of $\{u_1, \mathbf{p}_1\}$ is in order: a simple way to achieve that goal is to proceed as follows:

(i) Start iterating with scheme (3.11)-(3.13) for a small value of Δt, then stop time-stepping after a sufficiently (but not too) large number of time steps, denoting by $\{u_1, \mathbf{p}_1\}$ the pair produced by scheme (3.11)-(3.13).

(ii) Take for $\mathbf{M}$ the matrix-valued function $\mathrm{cof}(\mathbf{D}^2 u_1)$, and, using $\{u_1, \mathbf{p}_1\}$ as initializer, switch until convergence to the variant of scheme (3.11)-(3.13) associated with the initial value-problem (3.65).

A similar strategy can be used to speed up the convergence of scheme (3.14)-(3.17).

It is shown in particular that if $\mathbf{M}$ is close to $\mathrm{cof}(\mathbf{D}^2 u^*)$, where u^* is solution to problem (3.1), large values of τ can be used leading to fast convergence properties.

Remark 3.5.2. *As shown in Section 3.6, the above two-stage strategy can improve significantly the computational efficiency, particularly for those situations where the Monge-Ampère problem (3.1) has smooth, sufficiently isotropic, classical solutions. On the other hand, this two-stage strategy does not speed up significantly convergence if the solution is either strongly anisotropic, or non-smooth, or of the viscosity type. The readers will notice the Newton-like flavor of the above two-stage strategy.*

Remark 3.5.3. *In Theorem 3.5.1, we assume* $\mathbf{M}$ *is close to* $\mathbf{A}$*. In our experiments, this is not so restrictive.* $\mathbf{M}$*, no far from* $\mathbf{A}$*, provides very good results. Only a few steps in Stage 1 is enough.*

3.6 Numerical experiments

3.6.1 Generalities

In this section we will apply the algorithms discussed in Section 3.3 to 3.5, to the solution of a variety of test problems, with or without classical or smooth solutions. The associated domains Ω will be in particular the unit square $(0,1)^2$ (a popular one as one can guess) and the disk of radius $1/2$, centered at $(1/2, 1/2)$. Triangulations of these two domains have been visualised in Figures 3.1(a), (b), (c), and (d): The mesh in Figure 3.1(a) will be called the *regular* mesh, while the mesh on Figure 3.1(b) will be called the *symmetric* mesh (it has five symmetries, while the mesh in Figure 3.1(a) has three symmetries 'only'). The meshes on Figures 3.1(c) and 3.1(d) will be the *non-regular* (or *non-uniform*) ones, although they are fairly isotropic. It is worth mentioning that the meshes shown in Figures 3.1(c) and (d) have been generated using *distmesh* [99]. As expected (from the experiments reported in [15]), of all the triangulations we tested, those as the one shown in Figure 3.1(a), were the only ones not requiring a smoothing of the

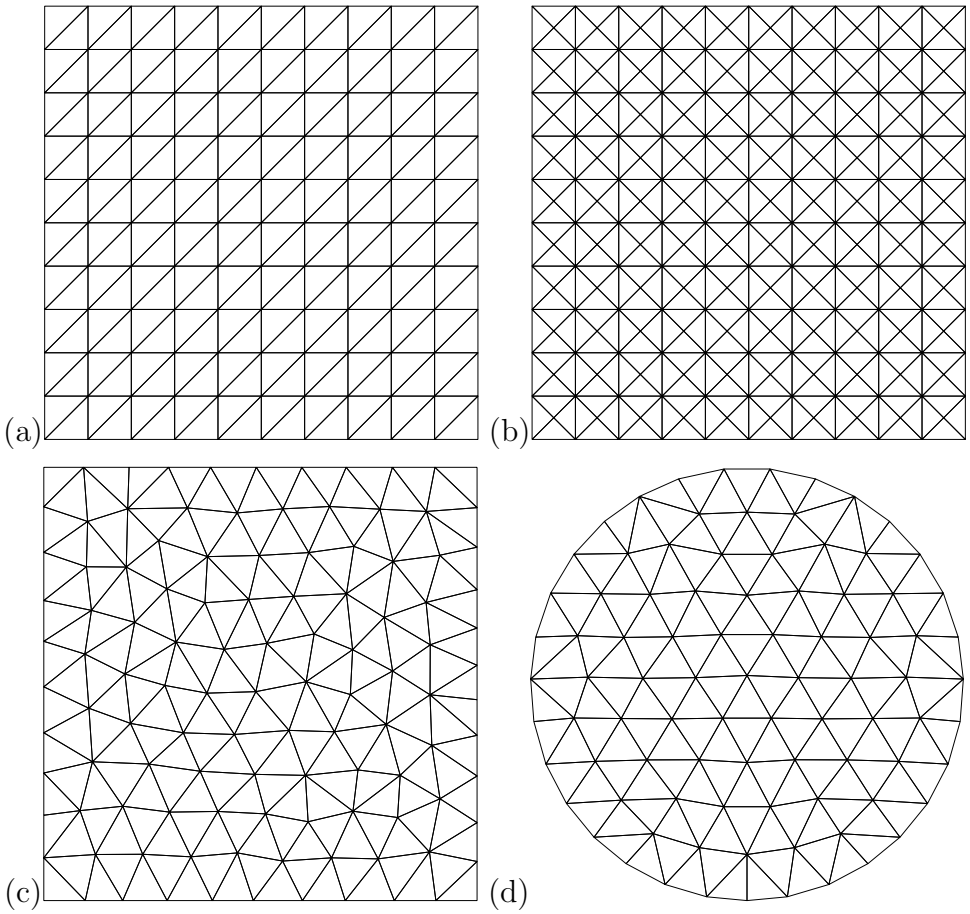


Figure 3.1: Four meshes for the two different domains used in the numerical experiments. (a) A regular mesh on a square. (b) A (highly) symmetric mesh on a square. (c) A non-regular mesh on a square. (d) A non-regular mesh on the half-unit disk.

discrete second order derivatives to produce accurate results.

Our main goal with the first two examples was to test and compare the performances of the original one-stage algorithm (3.11)-(3.13) (actually of a finite element variant of it, including the projection operation on the cone of the positive semi-definite matrices as discussed in Section 3.4.5) with those of the two-stage algorithm discussed in Section 3.5 . We took $\varepsilon = \varepsilon_1 = h^2, \beta = 1/4$ and $\Delta t = 2h^2$ in the discrete analogue of (3.11)-(3.13). In the first stage of the two-stage algorithm we used $\|\mathbf{D}^2 u^n - \mathbf{p}^n\| < 10$ as the criterion to switch to stage 2 (above,

the matrix-function norm $\|\cdot\|$ is defined by

$$\|\mathbf{S}\|^2 = \frac{1}{3}\sum_{k=1}^{N_h} |\omega_k| \|\mathbf{S}(Q_k)\|^2, \ \forall \mathbf{S} \in \mathbf{Q}_h,$$

the matrix norm of $\mathbf{S}(Q_k)$ being the Fröbenius one). In stage 2, we used $\Delta\tau = 1/2$ and took $\|u^{n+1} - u^n\|_2 < 10^{-5}$, typically, as stopping criterion ($\|\cdot\|_2$ being a trapezoidal rule based approximation of the L_2- norm). When using only the discrete analog of scheme (3.11)-(3.13), we use a stopping criterion similar to the one used in stage-2, but with a smaller tolerance to compensate the fact that $\Delta t (= 2h^2)$ is quite small (this appears clearly on Figure 3.3).

Remark 3.6.1. *A smaller stopping criterion can be used for stage 2, but this is not necessary since (as visible on Figure 3.2) the L_2 and L_∞ distances to the exact solution do not decrease anymore for n sufficiently large (likely an effect of the various regularizations we employ); this appears clearly on Figures 3.2(b), (c) and (d).*

3.6.2 Example 1

For the first example, we took as exact solution the function u^* defined by

$$u^*(x_1, x_2) = 8\left[\alpha\left(x_1 - \frac{1}{2}\right)^2 + \frac{1}{\alpha}\left(x_2 - \frac{1}{2}\right)^2\right] - 1$$

with the parameter $\alpha \geq 1, g = u^*|_{\partial\Omega}$ and $f = 256$.

The first case we consider is $\alpha = 1$, corresponding clearly to an isotropic solution u^*. On Table 3.1, we have reported some of the results obtained when applying the two-stage scheme to the solution of the related Monge-Ampère problem (3.1), indeed, for various meshes of the unit square and half-unit disk, we have shown the number of iterations (time steps) necessary to achieve convergence, the L_2

(a)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1414	16	6.78×10^{-6}	3.87×10^{-4}		6.93×10^{-4}	
0.0707	15	5.30×10^{-6}	9.75×10^{-5}	1.99	1.99×10^{-4}	2.00
0.0354	28	6.71×10^{-6}	2.79×10^{-5}	1.81	4.74×10^{-5}	1.88

(b)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	13	8.10×10^{-6}	7.89×10^{-3}		1.40×10^{-2}	
0.05	17	4.18×10^{-6}	1.98×10^{-3}	1.99	3.52×10^{-3}	1.99
0.025	23	4.49×10^{-6}	4.92×10^{-4}	2.01	8.72×10^{-4}	2.01

(c)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	17	5.05×10^{-6}	4.56×10^{-3}		1.41×10^{-2}	
0.05	15	5.19×10^{-6}	1.01×10^{-3}	2.17	3.94×10^{-3}	1.84
0.025	21	8.37×10^{-6}	2.25×10^{-4}	2.17	1.30×10^{-3}	1.60

(d)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	17	4.90×10^{-6}	1.93×10^{-3}		1.22×10^{-2}	
0.05	14	6.86×10^{-6}	6.46×10^{-4}	1.58	2.96×10^{-3}	2.04
0.025	20	7.56×10^{-6}	1.14×10^{-4}	2.5	8.30×10^{-4}	1.83

Table 3.1: (Example 1, $\alpha = 1$) Number of iterations necessary for the convergence of the 2-stage scheme, approximation errors and space approximation convergence rates for: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square non-uniform mesh, and (d) the half-unit disk triangulation.

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	38	1.45×10^{-6}	6.36×10^{-5}	1.30×10^{-4}
(ii)	0.0354	28	6.71×10^{-6}	2.79×10^{-5}	4.74×10^{-5}
(iii)	0.025	48	1.00×10^{-5}	5.20×10^{-4}	1.09×10^{-3}
(iv)	0.025	23	4.49×10^{-6}	4.92×10^{-4}	8.72×10^{-4}

Table 3.2: (Example 1, $\alpha = 1$) Comparison between the original one-stage scheme (i.e., the discrete analogue of scheme (3.11)-(3.13)) and the two-stage scheme for the meshes shown in Figure 3.1(a) and Figure 3.1(b): (i) Mesh (a) with the original one-stage scheme. (ii) Mesh (a) with the two-stage strategy. (iii) Mesh (b) with the original one-stage scheme. (iv) Mesh (b) with the two-stage strategy.

and L_∞ approximation errors and the corresponding convergence rates (roughly of the order of two, which is generically optimal for continuous piecewise affine finite element approximations). On Figure 3.2, we have visualized the graphs of the computed solutions, together with the related convergence and approximation errors histories for the four type of meshes considered here. We clearly see

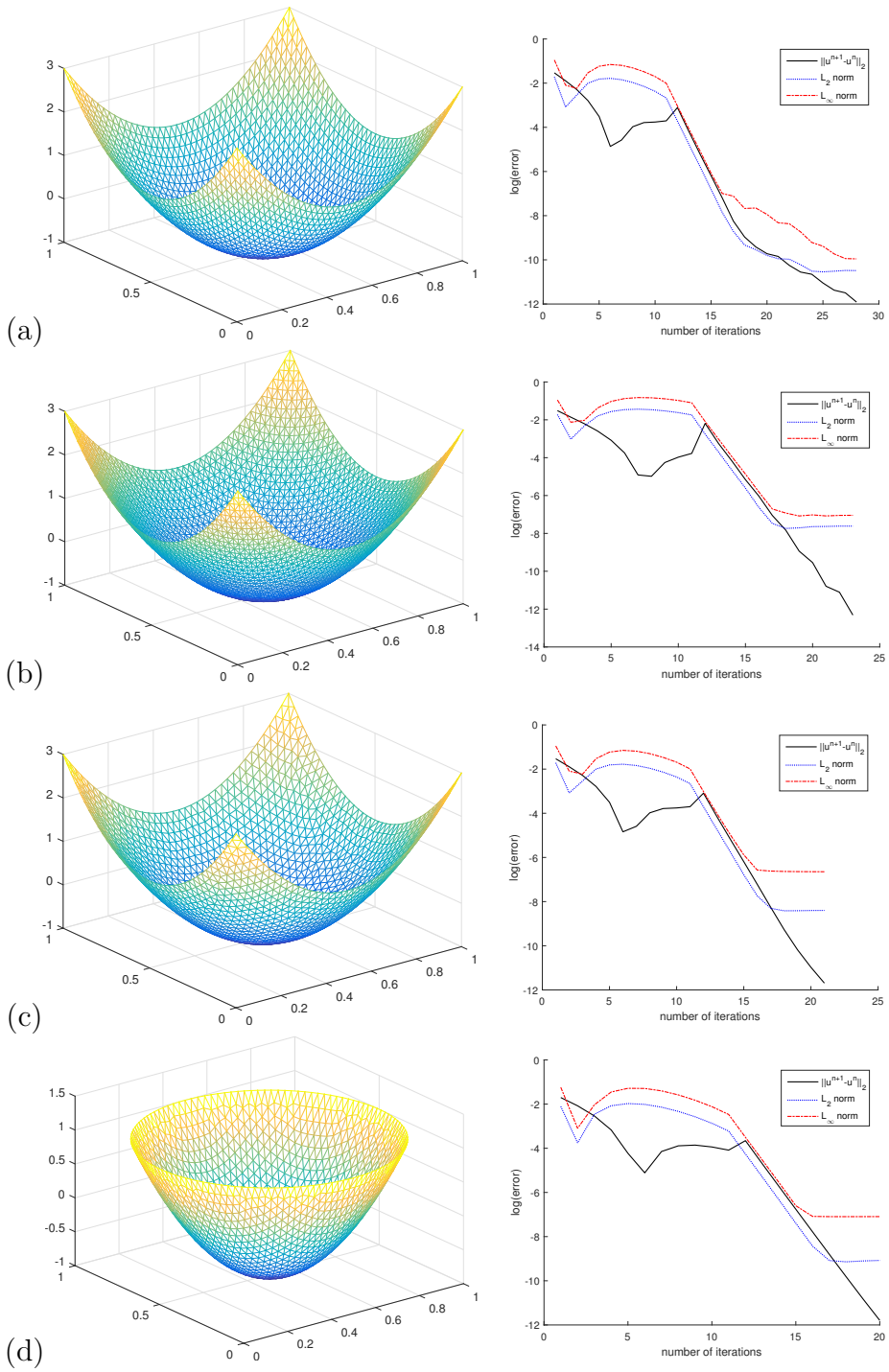


Figure 3.2: (Example 1, $\alpha = 1$) Graphs of the computed solutions obtained via the two-stage strategy and related convergence behaviors for: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square non-uniform mesh, and (d) the half-unit disk triangulation.

that the L_2 and L_∞ approximation errors reach a plateau for n large enough.

On Table 3.2 we have compared for meshes (a) and (b) of the unit square, the

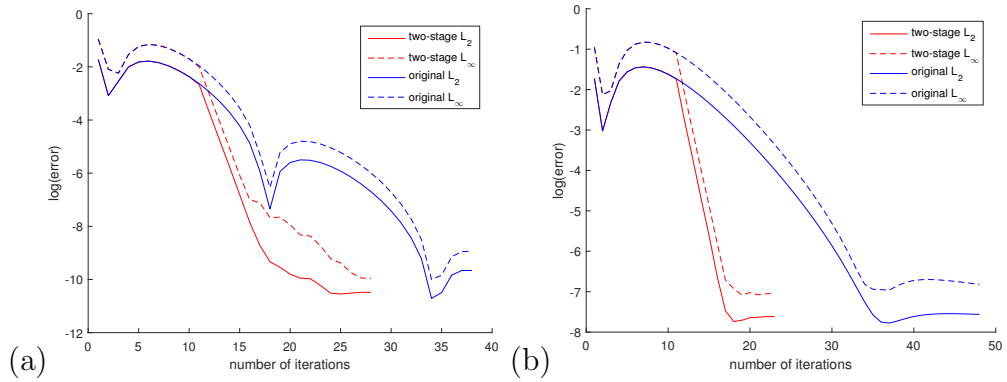


Figure 3.3: (Example 1, $\alpha = 1$) Convergence histories of the original one stage and two-stage schemes on (a) the unit square regular mesh and (b) the unit square symmetric mesh. Note that the red curves on these figures are the same as the corresponding ones in Figure 3.2(a) and (b).

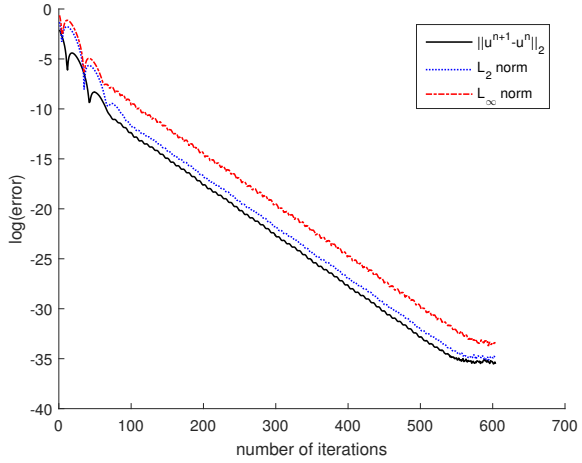


Figure 3.4: (Example 1, $\alpha = 1$) Error history for the original one-stage algorithm with $\epsilon = \epsilon_1 = 0$, $dt = h^2$ (unit square regular mesh).

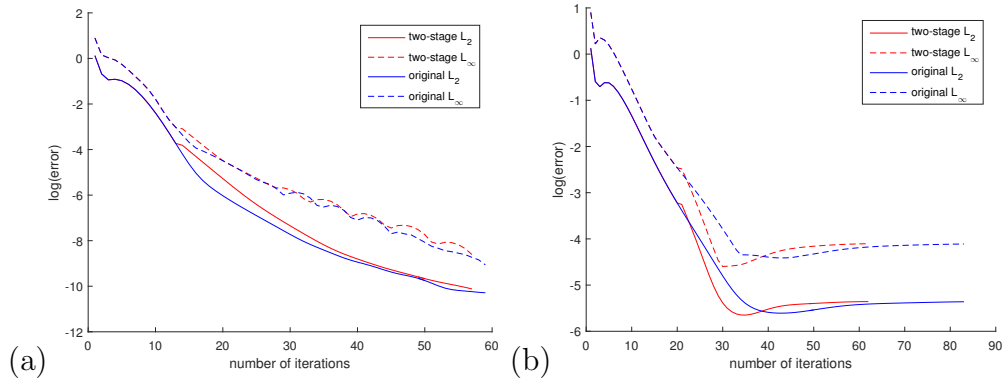


Figure 3.5: (Example 1, $\alpha = 5$) Convergence histories of the original one-stage and two-stage schemes for (a) the unit square regular mesh and (b) the unit square symmetric mesh.

convergence behaviors of the original one-stage and two-stage schemes, with the corresponding convergence indicators reported in Figure 3.3. Actually, Figure 3.3 shows very clearly the dramatic speed of convergence increase provided by the two-stage strategy (a Newton-like approach).

Suppose that one takes $\varepsilon = \varepsilon_1 = 0$ and focus on the regular mesh (a) for the unit square. Since the solution u^* of this test problem is a polynomial function of degree 2, relations (3.25) to (3.28) are exact for u^* if one employs a regular mesh like the one in Figure 3.1(a). Since $\frac{\partial}{\partial \mathbf{n}}\mathbf{D}^2 u^* = \mathbf{0}$, one expects to reach machine precision if one takes a sufficiently large number of time steps. This prediction is confirmed by the results reported in Figure 3.4, obtained using scheme 1 (the original one-stage scheme) with $\Delta t = h^2$ and $h = 0.0354$; these results show indeed that both the L_2 and L_∞ approximation errors converge to 0 with machine precision.

Next, still in the context of Example 1 test problem, we used our Monge-Ampère solver to investigate how varying the parameter α affects the convergence. Our approach works well on the regular mesh, even without regularization. On the other hand, we found that for the other meshes the proposed algorithms do not converge if one does not regularize (as shown in Section 3.4.3 and 3.4.3) the discrete second order derivatives (this was already mentioned in [15], so that the least-squares solution method discussed there produce converging solutions when $h \to 0$). Beyond $\alpha = 5$, the two-stage method does not bring notable improvement to the speed of convergence. In that direction, we have compared on Figure 3.5, the convergence properties of the original one-stage and two-stage schemes, for $\alpha = 5, \Delta\tau = 1/8$ in stage-2 of the second scheme, the meshes we used for this comparison being the unit square regular mesh with $h = 0.0354$, and the unit square symmetric mesh with $h = 0.025$. From Figure 3.5, the convergence histories of both schemes are quite similar.

Remark 3.6.2. *All the results reported in the current Section 3.6.2 have been*

obtained using relation (3.40) to impose (approximately) the condition $\frac{\partial \mathbf{p}_h}{\partial \mathbf{n}} = \mathbf{0}$ *on* $\partial\Omega$*. The results obtained using relations (3.34), (3.35) are very close to those obtained via (3.40), from both the number of iterations and approximation error points of view, this property still holding for the other test problems considered in this article. Since the relation (3.40) based approach is easier to implement it will be the only one to be considered concerning the results reported in the following sections.*

(a)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1414	10	7.37×10^{-7}	1.64×10^{-2}		3.75×10^{-2}	
0.0707	17	6.51×10^{-6}	4.95×10^{-3}	1.73	1.07×10^{-2}	1.81
0.0354	23	7.11×10^{-6}	1.28×10^{-3}	1.95	2.67×10^{-3}	2.00

(b)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	11	2.47×10^{-6}	2.43×10^{-2}		5.04×10^{-2}	
0.05	20	5.51×10^{-6}	8.33×10^{-3}	1.54	1.60×10^{-2}	1.66
0.025	22	8.10×10^{-6}	2.70×10^{-3}	1.63	4.76×10^{-3}	1.75

(c)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	10	2.43×10^{-6}	1.83×10^{-2}		4.19×10^{-2}	
0.05	17	6.53×10^{-6}	5.20×10^{-3}	1.82	1.18×10^{-2}	1.83
0.025	22	5.47×10^{-6}	1.39×10^{-3}	1.90	3.12×10^{-3}	1.92

(d)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	10	4.15×10^{-8}	1.14×10^{-2}		2.93×10^{-2}	
0.05	17	5.10×10^{-6}	3.25×10^{-3}	1.81	7.54×10^{-3}	1.96
0.025	22	4.41×10^{-6}	8.83×10^{-4}	1.88	2.09×10^{-3}	1.85

Table 3.3: (Example 2) Number of iterations necessary for the convergence of the two-stage scheme, approximation errors and space approximation convergence rates for (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square non-uniform mesh, and (d) the half-unit disk triangulation.

3.6.3 Example 2

In this example, we take $f(x_1, x_2) = 64e^{2r^2}(1 + 2r^2)$ and $g = u^*|_{\partial\Omega}$ with

$$u^*(x_1, x_2) = 4e^{r^2} - \frac{9}{2}$$

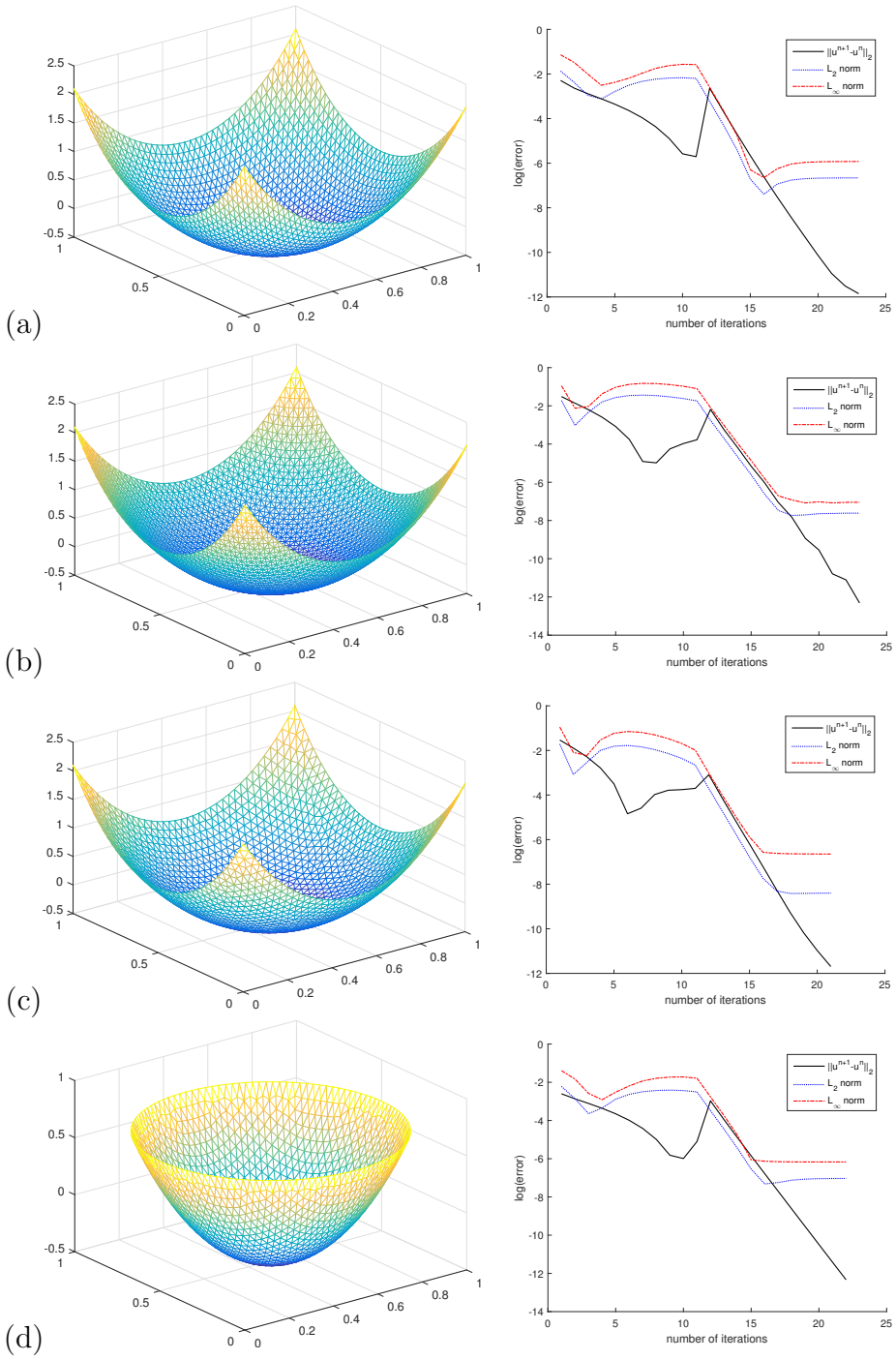


Figure 3.6: (Example 2) Graphs of the computed solutions obtained via the two-stage scheme and visualization of the convergence behaviors obtained on the unit square for the regular mesh (a), the symmetric mesh (b), the non-uniform mesh (c), and on the half-unit disk for a triangulation of type (d).

with $r^2 = (x_1 - 1/2)^2 + (x_2 - 1/2)^2$ so that $\det \mathbf{D}^2 u^* = f$. We have visualised in Figure 5.4 both the numerical solutions obtained by the two-stage strategy

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	65	9.74×10^{-6}	1.29×10^{-3}	2.69×10^{-3}
(ii)	0.0354	23	7.11×10^{-6}	1.28×10^{-3}	2.67×10^{-3}
(iii)	0.025	93	9.64×10^{-6}	2.61×10^{-3}	4.68×10^{-3}
(iv)	0.025	22	8.10×10^{-6}	2.70×10^{-3}	4.76×10^{-3}

Table 3.4: (Example 2) Comparison of the original one-stage and two-stage schemes for the meshes of Figure 3.1(a) and 3.1(b): (i) Mesh (a) with the original one-stage scheme. (ii) Mesh (a) with the two-stage strategy. (iii) Mesh (b) with the original one-stage scheme. (iv) Mesh (b) with the two-stage strategy.

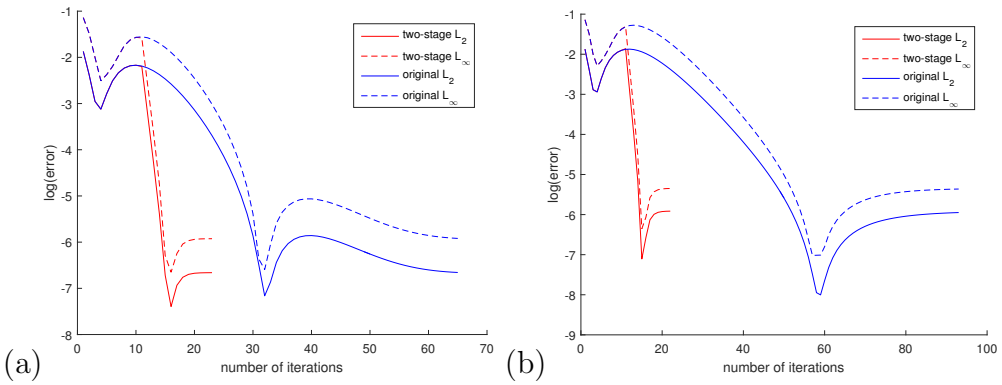


Figure 3.7: (Example 2) Convergence histories of the original one-stage and two-stage schemes for (a) the unit square regular mesh and (b) the unit square symmetric mesh. Note that the red curves on these figures are the same as the corresponding ones in Figure 5.4(a) and (b).

and also the convergence histories. Numerical results obtained for various values of h are shown in Table 3.3. We observe that the convergence rates for the L_2- and the L_∞-errors have slightly dropped from 2 (to approximately 1.8). For this example where problem (3.1) has a smooth classical convex solution, we expect the two-stage strategy to significantly improve the speed of convergence (as it did for the Example 1 test problem for α close to 1). This is indeed the case as shown by the results reported in Figure 3.7 and Tables 3.3 and 4.5.

3.6.4 Example 3

In this example, we consider a particular case of problem (3.1) defined by

(a)

h	Iterations	$\|u^{n+1}-u^n\|_2$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1414	40	7.81×10^{-7}	6.43×10^{-2}	2.90×10^{-2}
0.0707	139	9.14×10^{-7}	1.72×10^{-1}	6.76×10^{-2}
0.0354	531	9.79×10^{-7}	2.33×10^{-1}	8.90×10^{-2}

(b)

h	Iterations	$\|u^{n+1}-u^n\|_2$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1	58	8.99×10^{-7}	3.57×10^{-1}	1.53×10^{-1}
0.05	152	8.55×10^{-7}	3.66×10^{-1}	1.49×10^{-1}
0.025	710	9.87×10^{-7}	3.59×10^{-1}	1.40×10^{-1}

(c)

h	Iterations	$\|u^{n+1}-u^n\|_2$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1	37	9.25×10^{-7}	6.54×10^{-2}	2.98×10^{-2}
0.05	144	9.99×10^{-7}	1.38×10^{-1}	5.50×10^{-2}
0.025	562	9.92×10^{-7}	2.46×10^{-1}	9.30×10^{-2}

Table 3.5: (Example 3 with $\Omega=(0,1)^2$) Matching errors for the solutions computed on (a) the unit square regular mesh, (b) the unit square symmetric mesh, and (c) the unit square non-uniform mesh.

(a)

	Ω_1	Ω_1	Ω_2	Ω_2
h	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1414	1.49×10^{-3}	1.16×10^{-3}	8.12×10^{-4}	1.10×10^{-3}
0.0707	6.50×10^{-4}	5.95×10^{-4}	4.73×10^{-4}	5.57×10^{-4}
0.0354	1.27×10^{-3}	1.23×10^{-3}	8.82×10^{-4}	1.10×10^{-3}

(b)

	Ω_1	Ω_1	Ω_2	Ω_2
h	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1	1.48×10^{-3}	1.33×10^{-4}	5.54×10^{-4}	7.41×10^{-4}
0.05	2.06×10^{-3}	1.93×10^{-3}	1.80×10^{-3}	2.19×10^{-3}
0.025	6.50×10^{-3}	6.36×10^{-3}	3.32×10^{-3}	4.20×10^{-3}

(c)

	Ω_1	Ω_1	Ω_2	Ω_2
h	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$	$\|\mathbf{D}^2u^n-\mathbf{p}^n\|$	$\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$
0.1	5.92×10^{-4}	5.78×10^{-4}	2.98×10^{-4}	4.06×10^{-4}
0.05	3.51×10^{-4}	3.74×10^{-4}	2.51×10^{-4}	3.30×10^{-4}
0.025	1.30×10^{-3}	1.33×10^{-3}	8.30×10^{-4}	1.10×10^{-3}

Table 3.6: (Example 3 with $\Omega=(0,1)^2$) L^2 norms of the computed discrete analogues of the gap $\mathbf{D}^2u-\mathbf{p}$ restricted to the subdomains $\Omega_1=(0.2,0.8)^2$ and $\Omega_2=(0.25,0.75)^2$ for (a) the unit square regular mesh, (b) the unit square symmetric mesh, and (c) the non-uniform mesh on the unit square.

$$\begin{cases} \det \mathbf{D}^2u = 1 \text{ in } \Omega, \\ u = 0 \text{ on } \partial\Omega. \end{cases} \tag{3.74}$$

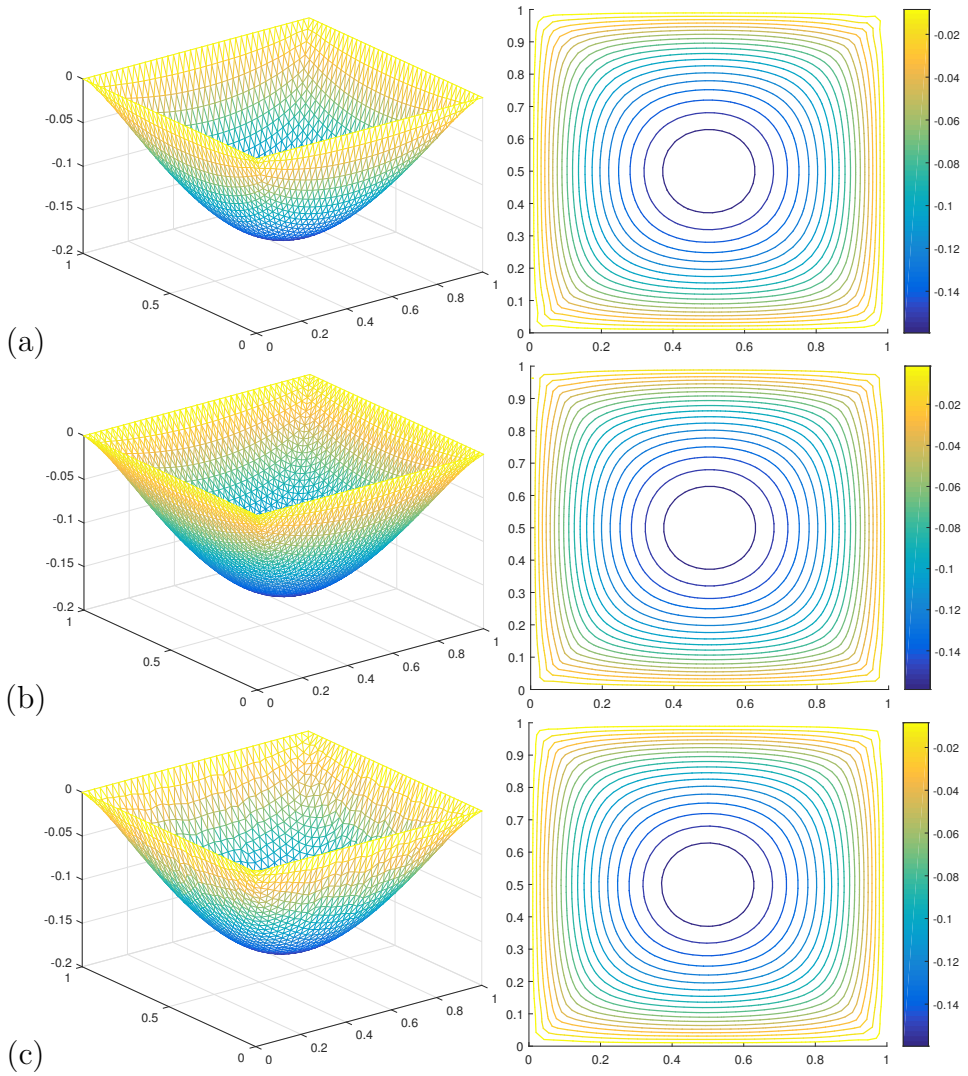


Figure 3.8: (Example 3 with $\Omega = (0,1)^2$) Graphs and contours of the computed solutions obtained by the original one-stage method for (a) the unit square regular mesh with $h = 0.0354$, (b) the unit square symmetric mesh with $h = 0.025$, and (c) the unit square non-uniform mesh with $h = 0.025$.

If $\Omega = (0,1)^2$, this example (introduced in [15]), has become a classical test problem for Monge-Ampère solvers, despite the fact it has no classical solutions, a consequence of the non-strict convexity of domain Ω(see the related discussion in [29]). This section has two parts: In the first part, we test our method using $\Omega = (0,1)^2$. In the second part, we solve (3.74) for a series of strictly convex domains which converge to $(0,1)^2$ to further test the performance of our method.

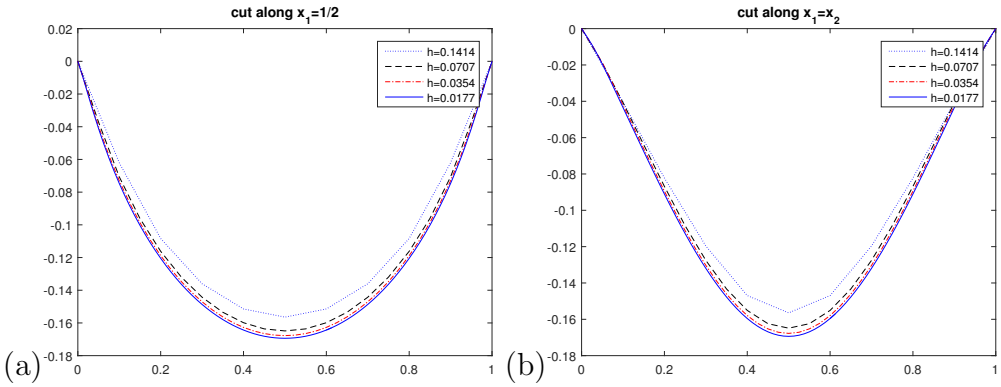
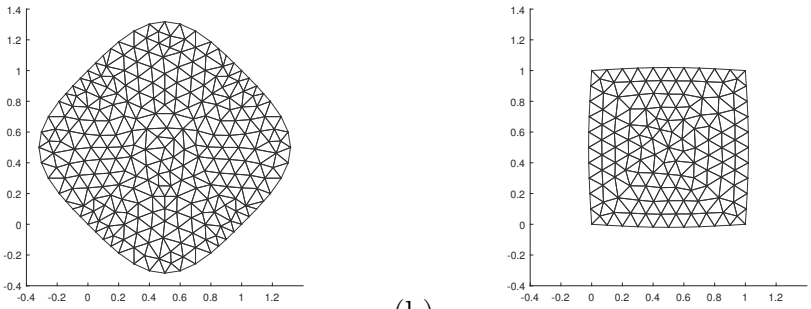


Figure 3.9: (Example 3 with $\Omega = (0,1)^2$) Cross sections for several values of h of the approximate solutions along (a) the line $x_1 = 1/2$ and (b) the first bisector $x_1 = x_2$ (regular meshes).

Example 3 with $\Omega = (0,1)^2$

With $\Omega = (0,1)^2$, although (3.74) has no classical solutions, it has however viscosity solutions, and also least-squares solutions in the sense of [27, 15, 30, 44]. To solve this challenging problem, we gave the value 10^{-3} to the two smoothing parameters. It seems that for this test problem the two-stage strategy does not improve the convergence to the steady state solutions, explaining why we employed only the original one-stage operator-splitting scheme, using $\|u^{n+1} - u^n\|_2 < 10^{-6}$ as stopping criterion. We have reported in Table 3.5, for various values of h and types of triangulations of the unit square, the number of time steps necessary to achieve convergence according to the above stopping criterion, and the L_2-norm of the computed discrete analogue of the gap $\mathbf{D}^2u - \mathbf{p}$. On Figure 3.8, we have visualized the graphs and contours of the computed solutions obtained with various types of triangulations of Ω; these various approximate solutions are very close to each other.

The results reported in Table 3.6 and Figure 3.9 are particularly interesting. Starting with Figure 3.9, we observe that the computed solutions converge to a smooth limit when $h \to 0$ and that the convergence is super-linear with respect to h; actually, this limit matches very accurately the least-squares solution obtained



(a) (b)

Figure 3.10: (Example 3 for a series of strictly convex domains) Finite element triangulation of Ω_a for $h = 1/80$ and (a) $a = 1/\pi$, (b)$a = 1/16\pi$.

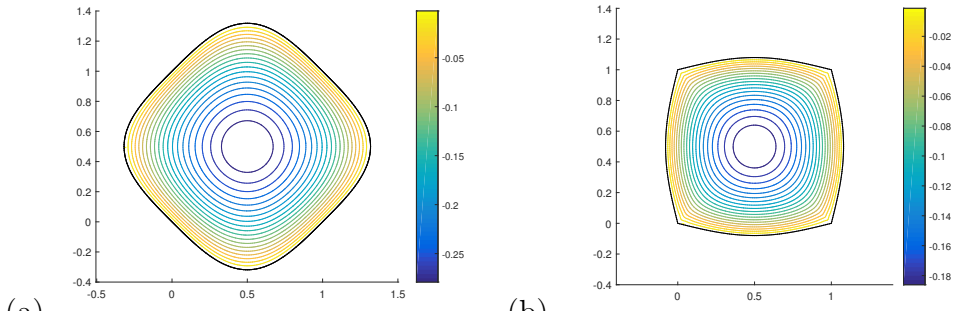
in [27, 15, 30, 44] for the same test problem. Figure 3.9(b) shows also that the above limit is concave close to the corners. On the other hand, Table 3.6 shows that sufficiently far way from the boundary, but not that far, the gap $\mathbf{D}^2u - \mathbf{p}$ is fairly small, implying in turn a similar property for the relation $\det \mathbf{D}^2u = 1$.

Example 3 for a series of strictly convex domains

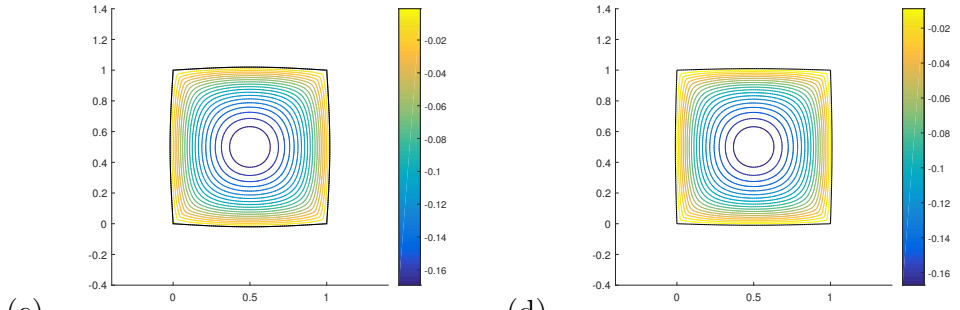
Since the non-existence of classical solutions to problem (3.74) stems form the non-strict convexity of domain $\Omega = (0,1)^2$, we were wondering what happens if Ω is approximated by a converging family of strictly convex domains. To investigate this issue, we introduced a family $\{\Omega_a\}_{a>0}$ of bounded two-dimensional domains whose boundary $\partial\Omega_a$ is defined by $\partial\Omega_a = \cup_{j=1}^4 \Gamma_a^j$ with

$$\begin{cases} \Gamma_a^1 = \{\{x_1, x_2\} | 0 \le x_1 \le 1, x_2 = -a \sin \pi x_1\}, \\ \Gamma_a^2 = \{\{x_1, x_2\} | x_1 = 1 + a \sin \pi x_2, 0 \le x_2 \le 1\}, \\ \Gamma_a^3 = \{\{x_1, x_2\} | 0 \le x_1 \le 1, x_2 = 1 + a \sin \pi x_1\}, \\ \Gamma_a^4 = \{\{x_1, x_2\} | x_1 = -a \sin \pi x_2, 0 \le x_2 \le 1\}. \end{cases} \tag{3.75}$$

Domain Ω_a is strictly convex if $a \in (0, 1/\pi]$ and we clearly have $\lim_{a\to 0} \Omega_a = \Omega$.

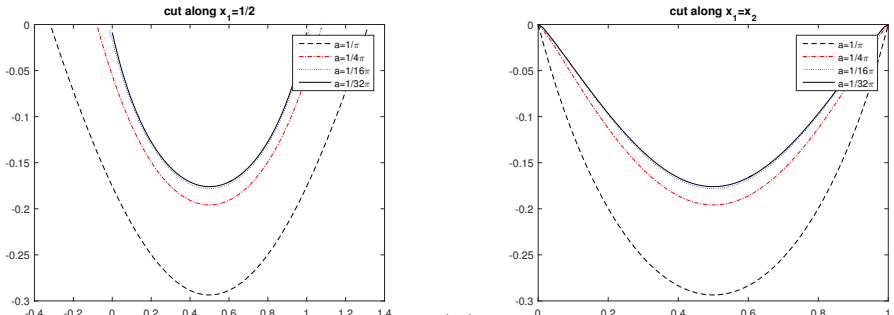


(a) (b)



(c) (d)

Figure 3.11: (Example 3 for a series of strictly convex domains) Contours of the computed solutions of problem (3.76): (a) $a = 1/\pi$, (b) $a = 1/4\pi$, (c) $a = 1/16\pi$, (d) $a = 1/32\pi$ ($h = 1/80$).



(a) (b)

Figure 3.12: (Example 3 for a series of strictly convex domains) Restriction of the computed solution of the problem (3.76) to: (a) the line $x_1 = 1/2$, (b) the line $x_1 = x_2$ ($h = 1/80$).

We applied the one-stage algorithm discussed in the preceding sections to the solution of problem

$$\begin{cases} \det \mathbf{D}^2 u = 1 \text{ in } \Omega_a, \\ u = 0 \text{ on } \partial\Omega_a, \end{cases} \tag{3.76}$$

for $a = 1/\pi, 1/4\pi, 1/16\pi$ and $1/32\pi$, the common mesh size of the triangulations of these four domains being $h = 1/80$ (the triangulation corresponding to $a = 1/16\pi$ has been visualized on Figure 3.10).

On Figure 3.11 (resp., Figure 3.12) we have visualized the contours of the computed solutions of problem (3.76) (resp., the graphs of the restriction of the computed solutions of problem (3.76) to the lines $x_1 = 1/2$ and $x_1 = x_2$). Figure 3.12 shows that for $a = 1/16\pi$ and $1/32\pi$, the computed solutions of problem (3.76) are very close to those visualized in Figure 3.9 for $h = 0.0354$ and 0.0177. Figure 3.12 suggests also that the restriction to Ω of the solution u_a of problem (3.76) converges super-linearly to the generalized solution of problem (3.74) obtained in Section 3.6.4.

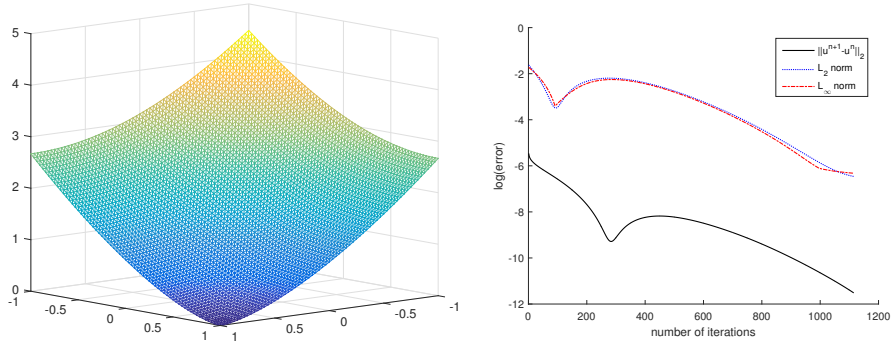


Figure 3.13: (Example 4) Graphs of the computed solution for $h = 0.0442$ and related convergence histories and approximation error behaviors(regular mesh).

3.6.5 Example 4

The fourth test problem that we consider is the following particular case of problem (3.1):

(a)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1768	106	9.69×10^{-6}	9.23×10^{-3}		8.75×10^{-3}	
0.0884	311	9.83×10^{-6}	3.71×10^{-3}	1.31	3.59×10^{-3}	1.29
0.0442	1115	9.96×10^{-6}	1.56×10^{-3}	1.25	1.79×10^{-3}	1.00

(b)

Grid Size	Error	rate	M1	M2
16	2.50×10^{-2}		564	601
32	1.60×10^{-2}	0.64	585	651
64	1.10×10^{-2}	0.54	976	1037

Table 3.7: (Example 4)Number of iterations and errors on regular meshes for different mesh sizes.(a) Results by the method discussed in the current article. (b) Results by the two methods discussed in [104] (M1 and M2 are the numbers of iterations needed by the two methods discussed in [104] to achieve convergence).

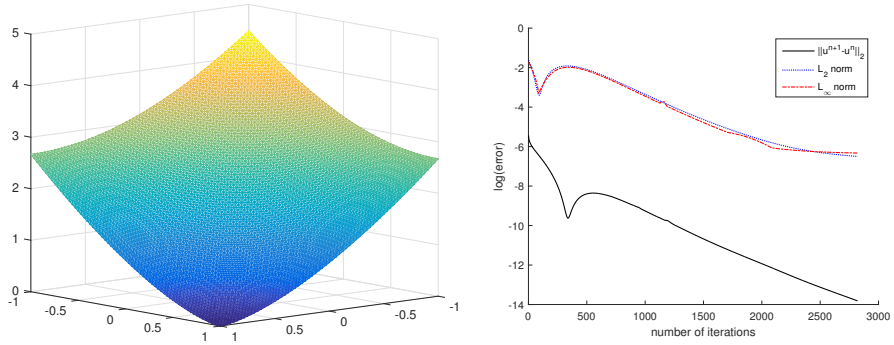


Figure 3.14: (Example 4) Graphs of the computed solution for $h = 0.0156$ and related convergence histories and approximation error behaviors(symmetric mesh).

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1768	271	9.78×10^{-7}	6.24×10^{-3}		7.97×10^{-3}	
0.0884	866	9.98×10^{-7}	2.68×10^{-3}	1.22	3.41×10^{-3}	1.22
0.0156	2820	9.99×10^{-7}	1.51×10^{-3}	0.83	1.79×10^{-3}	0.93

Table 3.8: (Example 4)Number of iterations and errors on symmetric meshes for different mesh sizes.

$$\begin{cases} \det \mathbf{D}^2 u = 1/r \text{ in } \Omega, \\ u = \frac{2\sqrt{2}}{3} r^{3/4} \text{ on } \partial\Omega, \end{cases} \tag{3.77}$$

with $\Omega = (-1,1)^2$ and $r = \sqrt{x_1^2 + x_2^2}$. Problem (3.77) has a unique convex solution given by $u = \frac{2\sqrt{2}}{3} r^{3/4}$; we observe that $f = 1/r$ blows up at $x = (0,0)$. All our computations have been done with regular meshes (like the one in Figure 3.1(a)),

using $\varepsilon = \varepsilon_1 = 2h^2$ as regularization parameters, and $\|u^{n+1} - u^n\|_2 < 10^{-5}$ as stopping criterion. Due (very likely) to the non-smoothness of f, the two-stage strategy does not pay off for problem (3.77), explaining why the results reported below have been obtained via the finite element analogue of scheme (3.11)-(3.13). The graph of the computed solution and the convergence history have been visualized on Figure 3.13, for $h = 0.0442$ (which corresponds to 64 grid points in each direction). In Table 3.7(a) we have reported for $h = 0.1768, 0.0884$ and 0.0442 (which correspond, respectively, to 16, 32, and 64 grid points in each direction) the number of iterations (time steps) necessary to achieve convergence, the L_2 and L_∞-norms of the approximation errors and the related rates of convergence. In [104] one discusses two methods for the numerical solution of fully nonlinear elliptic equations. These methods rely on those large stencil finite difference discretization schemes discussed in [38], and on accelerated gradient type algorithms à la Nesterov ([89, 111]) whose convergence rate is $O(1/k^2)$ where k denotes the number of iterations. For comparison purpose we have reported in Table 3.7(b) the results obtained by the two methods discussed in [104] using meshes of the same size as our finite element meshes. Actually the two methods discussed in [104] are also cascadic since a first approximate solution is computed on a coarse mesh and then interpolated on a mesh twice finer to initialize an iterative method. Tables 3.7(a) and 3.7(b) suggest that, for this test problem at least, our method is faster and more accurate than the ones discussed in [104].

In order to study the influence of the mesh on the solution process, we have used symmetric meshes (like the one in Figure 3.1(b)) to solve problem (3.77). The related results are reported in Table 3.8 and Figure 3.14, they have been obtained using $\|u^{n+1} - u^n\|_2 < 10^{-6}$ as stopping criterion.

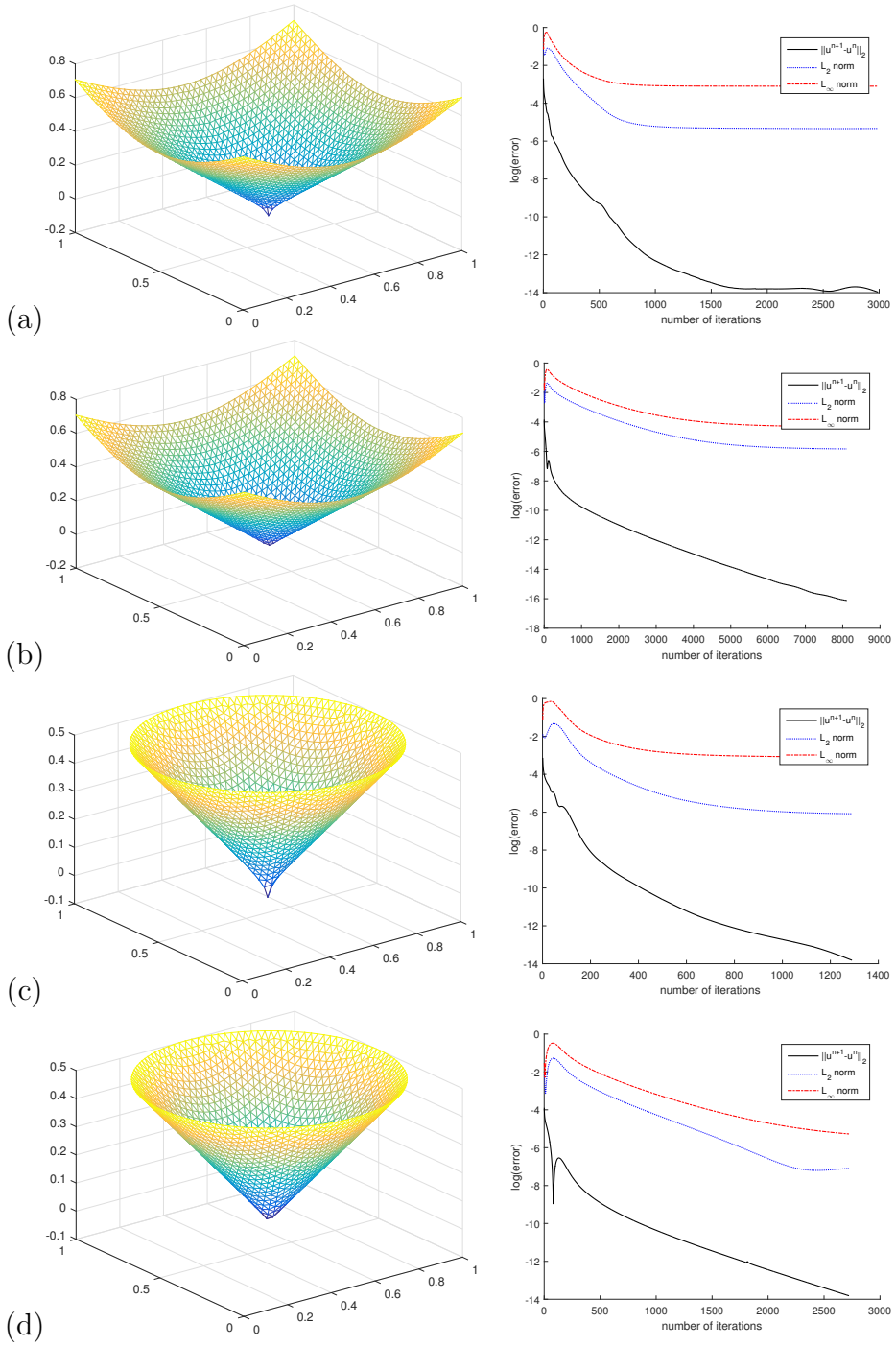


Figure 3.15: (Example 5) Graphs of the computed solutions obtained via the discrete variant of scheme (3.11)-(3.13) and visualization of the convergence behaviors obtained on: (a) (resp., (b)), the unit square for the regular mesh with $h = 0.0354$ and approximation (3.79) (resp., ((3.80)), (c) (resp., (d)) the half-unit disk for a non-uniform mesh with $h = 0.025$ and approximation (3.79) (resp., (3.80)).

(a)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1414	66	9.68×10^{-7}	2.32×10^{-2}		1.63×10^{-1}	
0.0707	295	9.89×10^{-7}	7.85×10^{-3}	1.56	8.76×10^{-2}	0.90
0.0354	2989	8.58×10^{-7}	4.86×10^{-3}	0.69	4.52×10^{-2}	0.95

(b)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1414	208	9.86×10^{-8}	6.85×10^{-3}		2.15×10^{-2}	
0.0707	1352	9.97×10^{-8}	3.52×10^{-3}	0.96	1.72×10^{-2}	0.32
0.0354	8112	1.00×10^{-7}	2.93×10^{-3}	0.26	1.33×10^{-2}	0.37

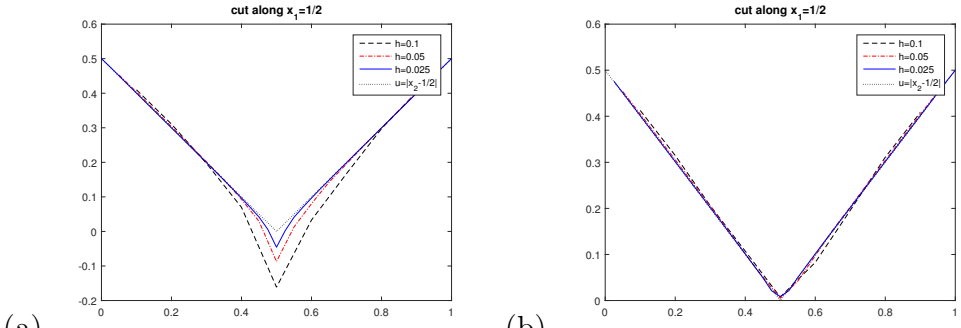
(c)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1	60	9.46×10^{-7}	2.16×10^{-2}		1.61×10^{-1}	
0.05	305	9.92×10^{-7}	6.54×10^{-3}	1.72	8.71×10^{-2}	0.88
0.025	1290	9.99×10^{-7}	2.29×10^{-3}	1.51	4.57×10^{-2}	0.93

(d)

h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2	rate	L_∞	rate
0.1	65	9.15×10^{-7}	6.54×10^{-3}		1.89×10^{-2}	
0.05	465	9.98×10^{-7}	2.68×10^{-3}	1.29	8.80×10^{-3}	1.10
0.025	2724	1.00×10^{-6}	8.37×10^{-4}	1.68	5.13×10^{-3}	0.78

Table 3.9: (Example 5) Number of iterations necessary to achieve convergence and related approximation errors for various values of h. (a) (resp., (b)) On the unit square with approximation (3.79) (resp., (3.80)) of the Dirac measure (regular meshes). (c) (resp., (d)) On the half-unit disk with approximation (3.79) (resp., (3.80)) of the Dirac measure (non-uniform meshes).



(a) (b)

Figure 3.16: (Example 5) Graphs of the restriction of the half-unit disk computed solutions to the line $x_1 = 1/2$ for different values of h. (a) Using approximation (3.79). (b) Using approximation (3.80).

3.6.6 Example 5

In order to further investigate the capabilities of our methodology, we have applied it to the solution of what we consider a demanding test problem, namely

$$\begin{cases} \det \mathbf{D}^2 u = \pi \delta_{(1/2,1/2)} \text{ in } \Omega, \\ u(x_1, x_2) = \sqrt{(x_1 - 1/2)^2 + (x_2 - 1/2)^2}, \forall (x_1, x_2) \in \partial\Omega, \end{cases} \tag{3.78}$$

where in (3.78), $\delta_{(1/2,1/2)}$ is the Dirac measure at $(1/2, 1/2)$ and Ω is either the square $(0,1)^2$ or the disk of radius $1/2$ centered at $(1/2, 1/2)$. The unique convex solution u of problem (3.78) is given by

$$u(x_1, x_2) = \sqrt{(x_1 - 1/2)^2 + (x_2 - 1/2)^2}, \forall (x_1, x_2) \in \Omega.$$

Two different approximations of the Dirac measure have been employed. The first one is defined by

$$f(x_1, x_2) = \begin{cases} (\frac{3}{|\omega_0|} + \epsilon_2)\pi & \text{if } x_1 = x_2 = \frac{1}{2}, \\ \epsilon_2 \pi & \text{otherwise} \end{cases} \tag{3.79}$$

where $|\omega_0|$ denotes the area of the polygonal ω_0, union of those triangle of $\mathcal{T}_h$ which have $(1/2, 1/2)$ as a common vertex.

To define the second approximation, we recall that

$$-\nabla^2 \ln \left(1/\sqrt{(x - 1/2)^2 + (y - 1/2)^2} \right) = 2\pi \delta_{(\frac{1}{2},\frac{1}{2})} \text{ (in the sense of distributions).}$$

Introducing a small parameter ϵ_3 we have

$$-\nabla^2 \ln \left(1/\sqrt{\epsilon_3^2 + (x - 1/2)^2 + (y - 1/2)^2} \right) = \frac{2\epsilon_3^2}{\left(\epsilon_3^2 + (x - 1/2)^2 + (y - 1/2)^2\right)^2},$$

suggesting as second approximation of the Dirac measure the function f_{ϵ_3} defined by

$$f_{\epsilon_3}(x_1, x_2) = \frac{\epsilon_3^2}{\left(\epsilon_3^2 + (x_1 - 1/2)^2 + (x_2 - 1/2)^2\right)^2}. \tag{3.80}$$

Our methodology has been tested, for several values of h, on two domains, namely, the square $(0,1) \times (0,1)$ (with $\varepsilon = h^2, \varepsilon_1 = h^2, \varepsilon_2 = h, \varepsilon_3 = h$) on regular meshes, and then the disk of radius $1/2$ centered at $(1/2, 1/2)$ (with $\varepsilon = h^2, \varepsilon_1 = h^2, \varepsilon_2 = h, \varepsilon_3 = h$) on non-uniform meshes. For the disk, one has $u|_{\partial\Omega} = 1/2$ as boundary condition. When solving problem (3.78) on the unit square, using the approximation (3.80) of the Dirac measure, we took $\|u^{n+1} - u^n\|_2 < 10^{-7}$ as stopping criterion; for the other situations associated with problem (3.78) we investigated, we used the less demanding stopping criterion $\|u^{n+1} - u^n\|_2 < 10^{-6}$ since it was bringing results of comparable quality. The graphs of the computed solutions and the related convergence histories have been reported on Figure 3.15. The number of iterations necessary to achieve convergence and the behaviors of the L^2 and approximation errors have been reported on Table 3.9. Approximation (3.79) of the Dirac measure provides a faster convergence, if measured in number of iterations, however, approximation (3.80) provides a significantly smaller approximation error, particularly close to the point $(1/2, 1/2)$ as shown in Figure 3.16.

Remark 3.6.3. *Approximation (3.80) of the Dirac measure involves only one small parameter, namely ε_3, necessarily a distance. Resolution considerations strongly suggest taking $\varepsilon_3 \approx h$. The good quality of the results obtained with $\varepsilon_3 = h$ confirm the soundness of our analysis.*

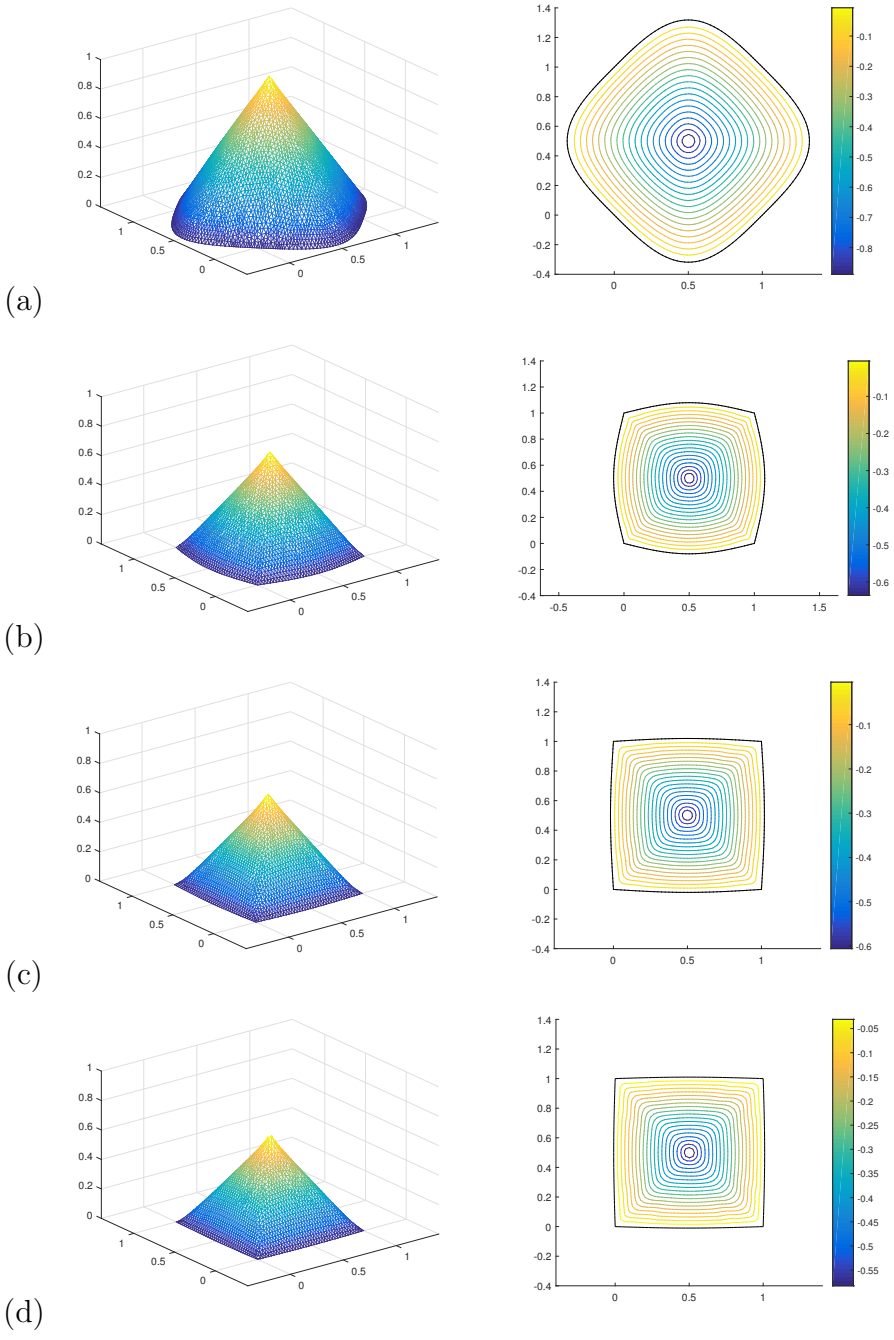
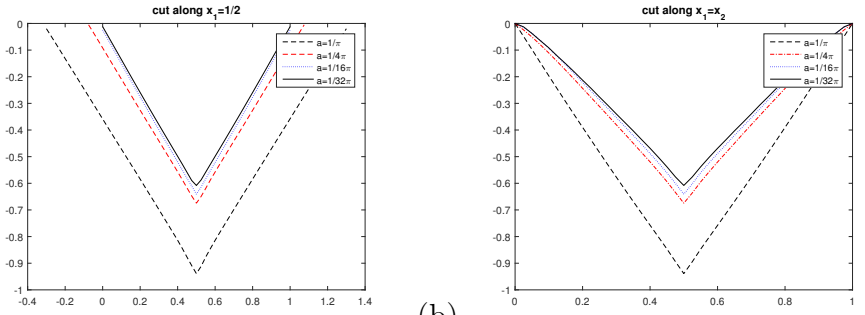


Figure 3.17: (Example 6) Graphs and contours of the computed approximate concave solutions of problem (72) for (a) $a = 1/\pi$, (b) $a = 1/4\pi$, (c) $a = 1/16\pi$, (d) $a = 1/32\pi$ and $h = 1/40$.



(a) (b)

Figure 3.18: (Example 6) Graphs of the restrictions of the computed approximate convex solutions of problem (72) to (a) the line $x_1 = 1/2$, and (b) the line $x_1 = x_2$, for $a = 1/\pi, 1/4\pi, 1/16\pi, 1/32\pi$ and $h = 1/40$.

3.6.7 Example 6

We investigate the solution of the following variant of problem (3.78) in Section 3.6.6

$$\begin{cases} \det \mathbf{D}^2 u = \pi \delta_{(1/2,1/2)} \text{ in } \Omega, \\ u = 0 \text{ on } \partial\Omega, \end{cases} \tag{3.81}$$

with $\Omega = (0,1)^2$. Problem (3.81) is challenging to our methodology since the right-hand side of the related Monge-Ampère equation is not a function but a measure vanishing in $\Omega \backslash \{1/2, 1/2\}$, the non-strict convexity of Ω creating additional problems. Indeed, when applying the one-stage algorithm, we discussed in the preceding sections, to the solution of the variant of problem (3.81) obtained by replacing $\pi\delta_{(1/2,1/2)}$ by its approximation defined by (3.80) (with $\epsilon_3 = h$), we could not obtain convergence. Since the divergence was mostly taking place at the corners of Ω, an obvious attempt at rescuing the situation was to replace the above domain by the family $\{\Omega_a\}_a$ defined by (3.75), still using the approximation (3.80) of $\pi\delta_{(1/2,1/2)}$. This (kind of) regularization of Ω seems to work, as shown by the results we obtained for $a = 1/\pi, 1/4\pi, 1/16\pi, 1/32\pi$ and $h = 1/40$. These results have been visualized on Figure 3.17 (resp., Figure 3.18) which shows the graphs and contours of the computed concave solutions (resp., the graphs of the computed convex solutions restricted to the lines $x_1 = 1/2$ and $x_1 = x_2$). These

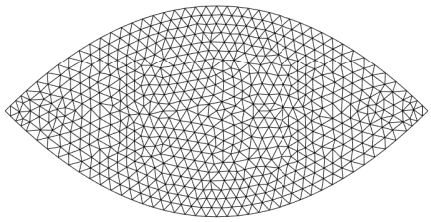


Figure 3.19: Eye-shape mesh with $h = 0.025$.

results strongly suggest that when $a \to 0$, one has convergence of the computed solutions to a piecewise affine function whose graph is an inverted pyramid.

3.6.8 Example 7

We will conclude our numerical experiments with the solution of (3.76) and (3.81) on the following eye-shape domain:

$$\Omega = \{\{x_1, x_2\} | -x_1(1-x_1) < x_2 < x_1(1-x_1), 0 < x_1 < 1\}.$$

This domain is strictly convex and with two corners. The graph of the domain with $h = 0.025$ is shown in Figure 3.19. When solving (3.81), the delta function is modified to be supported at $(1/2, 0)$ and the two approximations (3.79) and (3.80) are tested. The figures and contours of the three problems are shown in Figure 3.20. In Figure 3.20, the result of (3.76) is very smooth. For (3.81), similar phenomenon of the two approximation observed in Section 3.6.6 can be seen: a pin point appears by approximation (3.79) whereas approximation (3.80) provides more smooth solution.

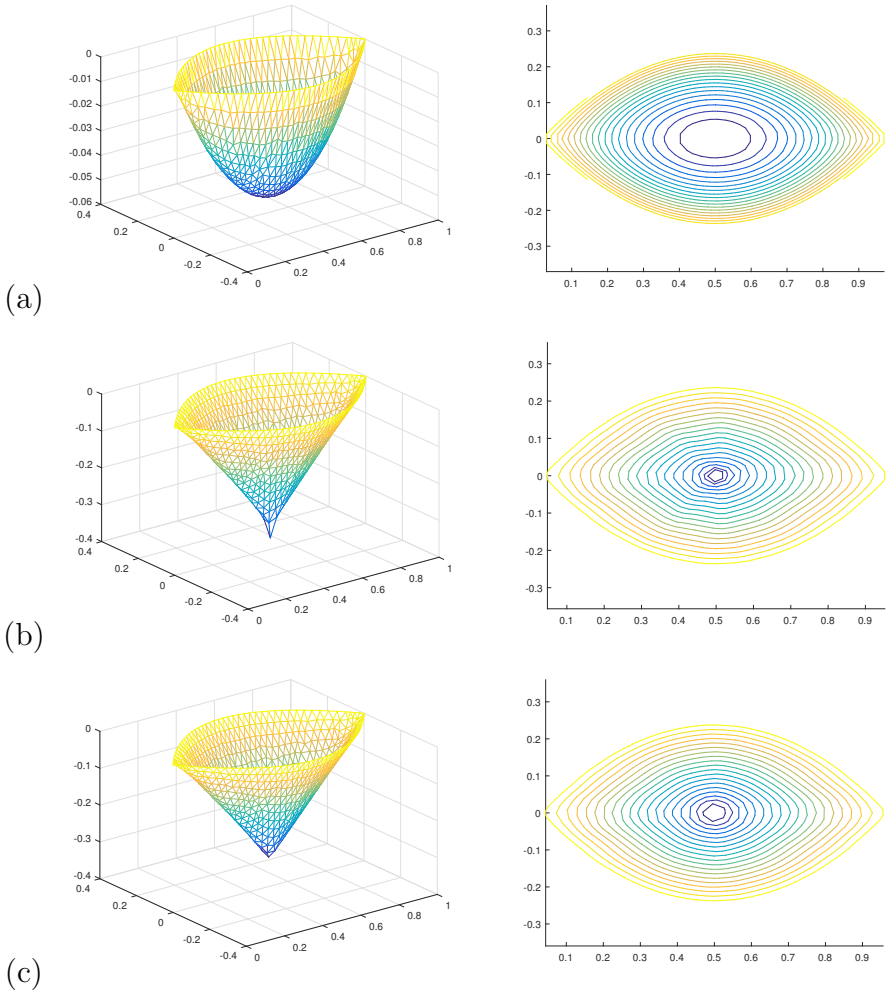


Figure 3.20: (Example 7, $h = 0.025$) Graph of the computed solution and its contours of (a) (3.76), (b)(3.81) with approximation (3.79), (c) (3.81) with approximation (3.80).

3.6.9 Further comments

More or less at the same time we started investigating the additional problems discussed in Section 3.6.4 and 3.6.7, our colleague X.C. Tai suggested us to have a look at the so-called '*heavy ball*' method introduced by B.T. Polyak in [102], as a possible tool to speed up the convergence of our time stepping methods to the solution of the discrete analogues of problem (3.1). To give an idea of the Polyak method, let us consider the following unconstrained minimization problem

$$\begin{cases} x \in X, \\ J(x) \leq J(y), \end{cases} \tag{3.82}$$

with $J : X \to \mathcal{R}$ convex and differentiable over the real Hilbert space X. Assuming it exists, the solution of problem (3.82) is characterized by

$$DJ(x) = 0, \tag{3.83}$$

where DJ is the differential of functional J. Suppose we use the gradient flow approach to solve (3.82) via (3.83): we associate thus to (3.83) the following initial value problem

$$\begin{cases} \frac{dx}{dt} + DJ(x) = 0 \text{ on } (0, +\infty), \\ x(0) = x_0. \end{cases} \tag{3.84}$$

The steady state solutions of (3.84) are solutions of (3.82), (3.83), justifying using (3.84) to solve (3.82), (3.83). Let $\tau(> 0)$ be a time discretization step; the simplest scheme we can think about for the time discretization of (3.84) is the

forward Euler one; it leads to the following algorithm of the gradient type:

$$\begin{cases} x^0 = x_0. \\ \text{For } n \geq 0, x^n \to x^{n+1} \text{ as follows:} \\ \frac{x^{n+1}-x^n}{\tau} + DJ(x^n) = 0. \end{cases} \tag{3.85}$$

In [102], B.T. Polyak suggested replacing (3.85) by

$$\begin{cases} x^0 = x_0, x^{-1} = x_1 (= x_0 \text{ for example}). \\ \text{For } n \geq 0, \{x^{n-1}, x^n\} \to x^{n+1} \text{ as follows:} \\ \frac{x^{n+1}-x^n}{\tau} + DJ(x^n) - \beta \frac{x^n - x^{n-1}}{\tau} = 0, \end{cases} \tag{3.86}$$

with $\beta \in (0, 1)$. Actually (3.86) can be rewritten as

$$\begin{cases} x^0 = x_0, x^{-1} = x_1 (= x_0 \text{ for example}). \\ \text{For } n \geq 0, \{x^{n-1}, x^n\} \to x^{n+1} \text{ as follows:} \\ (1+\beta)\frac{\tau}{2}\frac{x^{n+1}+x^{n-1}-2x^n}{\tau^2} + (1-\beta)\frac{x^{n+1}-x^{n-1}}{2\tau} + DJ(x^n) = 0, \end{cases} \tag{3.87}$$

a discrete analog of

$$\begin{cases} \epsilon \frac{d^2x}{dt^2} + (1-\beta)\frac{dx}{dt} + DJ(x) = 0 \text{ in } (0, +\infty), \\ x(0) = x_0, \frac{dx}{dt}(0) = p_0, \end{cases} \tag{3.88}$$

with $\epsilon = (1+\beta)\frac{\tau}{2}$ and $p_0 = 0$ if $x_1 = x_0$ (another possibility is to take $p_0 = -DJ(x_0)$). When we modify the final version of our draft, we find that (3.88) is pretty similar to what is proposed in [111], which is the limit of Nesterov's accelerated gradient method mentioned in Section 3.6.5 with the same convergence rate $O(1/k^2)$. From (3.88), it is clear that the Polyak trick introduces inertia in the numerical model, the added mass being $(1+\beta)\frac{\tau}{2}$. It is shown in [102] that there are situations where, for a well-chosen value of β, the Polyak

modification of algorithm (3.85) dramatically improves the speed of convergence to steady state solutions. Unfortunately, the improvements it brings to the algorithms discussed in the preceding sections are marginal (if any). This may be due to the fact that our algorithms are based on the implicit/explicit time discretization schemes, unlike the algorithms in [102] which are all related to fully explicit schemes. On the other hand, from the numerical results reported there, it seems that the algorithms discussed in [104] significantly benefited from convergence acceleration techniques strongly related to those discussed in [102] and made popular by Nesterov [89]

3.7 Conclusion

In this chapter, we have developed a relatively easy to implement finite element and operator-splitting based methodology for the numerical solution of the Monge-Ampère equation. The related method has been working well for various types of triangulations (structured and unstructured) and can handle curved boundary quite easily. We introduced also a Newton-like two-stage variant of our methodology, which accelerates significantly the convergence if the problem under consideration has a smooth convex solution. In the following chapters, a series of variant of this method is introduced to solve other problems of the Monge-Ampère type.

Chapter 4

The Minkowski Problem

4.1 Introduction

The Minkowski problem, named after Hermann Minkowski, is a fundamental problem in differential geometry. It asks for the construction of a strictly convex function with prescribed Gauss curvature.

Given a compact, strictly convex hypersurface u in $\mathbb{R}^n$, the Gaussian map $G(x)$ is a diffeomorphism from u to $\mathbf{S}^m$ such that $G(x)$ is the unit outward normal direction of u at x. Then the Gauss-Kronecker curvature K is the Jacobian determinant of the Gauss map of u. Minkowski stated that we must have

$$\int_{\mathbf{S}^n} x_i K^{-1} = 0 \tag{4.1}$$

with x_i being the coordinate function. Minkowski problem is the converse of the above problem: If a positive function K defined on $\mathbf{S}^n$ satisfies relation (4.1), can we find such hypersurface u? The existence and uniqueness of solutions were analyzed by Minkowski in [84, 85]. The regularity of the solution in two dimensions were proved by Lewy in [69, 70], Nirenberg in [91] and Pogorelov in

[100]. In high diemnsion, the regularity was analyzed by Cheng and Yau in [24], and Pogorelov in [101].

For this problem, only a few numerical algorithm can be found. The earliest one we can find is in [76, 77], which solved a related problem: reconstructing the surface from the extended Gaussian image. In [63], after a generalization of Minkowski's proof, the Minkowski problem is converted to an optimization problem and the algorithm solves a polyhedral version of the Minkowski problem. In [64], an algorithm based on Minkowski's isoperimetric inequality was introduced. The Minkowski problem is solved in finite function space in which truncated spherical harmonics series are used. In a more recent work, in [21], a level set based PDE method was designed. In this method, a flow is introduced. Given an initial implicitly defined surface, it evolves to the solution of the Minskowski problem under the level set framework. The method is discretized by finite difference method.

In the above description and analysis, the manifold described by u is assumed to be closed. Another type of the Minkowski problem is a Dirichlet problem. In the Dirichlet Minkowski problem, u is open and a Dirichelt boundary condition is given. This problem has been studied by many authors. The existence and uniqueness of the solution was studied by Bakelman in [4], Lions in [75] and Urbas in [117, 118, 119]. A necessary and sufficient condition of the classical solvability was proved by Trudinger and Urbas in [115].

In this work, we use an equivalent formulation of the Minkowksi problem and consider the Dirichlet problem. Let K be a given Gauss-Kronecker curvature and u be the solution to the corresponding Minkowski problem. Then u must satisfies

$$\frac{\det(\mathbf{D}^2 u)}{(1+|\nabla u|^2)^{(n+2)/2}} = K. \tag{4.2}$$

(4.2) is one type of the Monge-Ampère equation which in general can be written

as

$$\det(\mathbf{D}^2 u) = f. \tag{4.3}$$

In this chapter, based on the operator-splitting method introduced before, we introduce a new algorithm to solve the Dirichlet Minkowski problem.

This chapter is organized as follows: In Section 4.2, we state some theoretical results on the solution of the Dirichlet Minkowski problem. In Section 4.3, we show the divergence formulation of the problem and introduce the equivalent time dependent PDE system. The system is time discretized in Section 4.4 and space discretized in Section 4.5. The initialization and algorithm is summarized in Section 4.6. In Section 4.7, we show our numerical results of the performance of the algorithm. Conclusion follows in Section 4.8.

4.2 Problem description

The Minkowski problem tries to find a strictly convex surface with prescribed Gaussian curvature. In this work, we focus on the Minkowski problem with Dirichlet boundary condition:

$$\begin{cases} \det(\mathbf{D}^2 u) = K(1 + |\nabla u|^2)^{(n+2)/2}, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{4.4}$$

where Ω is the domain on which u is defined, $K \in C^{1,1}(\Omega) \cap C^{0,1}(\bar{\Omega})$ is a given positive function (the prescribed curvature), $g \in C^{1,1}$ is the boundary of the u and $\mathbf{D}^2 u$ is the Hessian matrix:

$$\mathbf{D}^2 u = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 \partial x_2} \\ \frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_2^2} \end{pmatrix}.$$

Here we state some theory regarding the existence, uniqueness and regularity of the solution to (4.4). The first one, from [115], regards on the classical solvability of (4.4)

Theorem 4.2.1. *Let Ω be a uniformly convex $C^{1,1}$ domain in $\mathbb{R}^n$ and K a positive function in $C^{1,1} \cap C^{0,1}(\bar{\Omega})$. Then the classical Dirichlet problem (4.4) is uniquely solvable for arbitrary $g \in C^{1,1}(\bar{\Omega})$, with convex solution $u \in C^2(\Omega) \cap C^{0,1}(\bar{\Omega})$, if and only if the following two conditions hold:*

$$\int_\Omega K dx < \int_{\mathbb{R}^n} (1+|p|^2)^{-1/(n+2)} dp, \tag{4.5}$$

$$K = 0 \text{ on } \partial\Omega. \tag{4.6}$$

To make sure the existence of the solution for arbitrary g, condition (4.6) is necessary. In [115], the authors stated that if K does not vanish on the boundary, there exists one g such that (4.4) has no solution.

For the limit case of condition (4.5):

$$\int_\Omega K dx = \int_{\mathbb{R}^n} (1+|p|^2)^{-1/(n+2)} dp, \tag{4.7}$$

the regularity of the solution is studied in [117, 118, 119] and we have the following theorem:

Theorem 4.2.2. *Let Ω be a uniformly convex domain with $C^{2,1}$ smooth boundary, and K a positive, C^2 smooth function, satisfying (4.7). Let u be a solution to the Dirichlet problem (4.4). Then*

(i) $u \in C^{1/2}(\Omega)$;

(ii) the graph of u is $C^{2,\alpha}$ smooth for some $\alpha \in (0,1)$;

(iii) the restriction of u on $\partial\Omega$ is $C^{1,\alpha}$ smooth;

(iv) if $\partial\Omega \in C^{k+1,\alpha}$ and $K \in C^{k-1,\alpha}, k \geq 2$, then the graph of u is $C^{k+1,\alpha}$ smooth and the restriction of u on $\partial\Omega$ is $C^{k+1,\alpha}$ smooth.

Concerning on the solution to the Minskowski problem, we refer interested readers to [116] for a complete discussion.

Remark 4.2.1. *In the above theorems, the conditions are restrictive and not easy to satisfy. To test the performance of our algorithm, we just choose some settings such that the smooth exact solution exists.*

4.3 A divergence formulation of problem and an associated initial value problem

In this work, we consider the two dimensional Minkowski problem

$$\begin{cases} \det(\mathbf{D}^2 u) = K(1 + |\nabla u|^2)^2, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{4.8}$$

Note that (4.8) can be rewritten as

$$\begin{cases} \det(\mathbf{D}^2 u) = K(1 + |\nabla u|^2)^2, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{4.9}$$

which is a kind of the Monge-Ampère type equation. Instead of solving (4.9) directly, we take advantage of the equivalence between (4.9) and a divergence form

$$\begin{cases} -\nabla \cdot (\mathrm{cof}(\mathbf{D}^2 u)\nabla u) + 2K(1 + |\nabla u|^2)^2 = 0, \\ u = g \text{ on } \partial\Omega. \end{cases} \tag{4.10}$$

In (4.10), $\mathrm{cof}(\mathbf{D}^2u)$ is the cofactor matrix of $\mathbf{D}^2u$ defined as

$$\mathrm{cof}(\mathbf{D}^2u) = \begin{pmatrix} \frac{\partial^2 u}{\partial x_2^2} & -\frac{\partial^2 u}{\partial x_1 \partial x_2} \\ -\frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_1^2} \end{pmatrix}.$$

Introducing the matrix valued variable $\mathbf{p}$ and the viscosity term $-\varepsilon\nabla^2 u$ and notice that

$$\nabla^2 u = \nabla \cdot (\nabla u),$$

we solve the regularized problem

$$\begin{cases} \begin{cases} -\nabla \cdot ((\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}))\, \nabla u) + 2K(1 + |\nabla u|^2)^2 = 0, \\ u = g \text{ on } \partial\Omega, \end{cases} \\ \mathbf{p} - \mathbf{D}^2 u = \mathbf{0}. \end{cases} \tag{4.11}$$

To solve (4.11), we associate it with time derivatives with of u and $\mathbf{p}$ to get an initial value problem:

Given u_0 and $\mathbf{p}_0$, solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot ((\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}))\, \nabla u) + 2K(1 + |\nabla u|^2)^2 = 0, \\ u = g \text{ on } \partial\Omega, \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma\,(\mathbf{p} - \mathbf{D}^2 u) = \mathbf{0} \end{cases} \tag{4.12}$$

to steady state. In (4.12), γ is a factor controlling the evolution speed of $\mathbf{p}$. In the evolution, we want $\mathbf{p}$ to evolve at approximately the same speed of u. A reasonable choice is

$$\gamma = \beta\lambda_0 \left(\varepsilon + \sqrt{\alpha}\right),$$

where λ_0 is the smallest eigenvalue of operator $-\nabla^2$ in $H_0^1(\Omega)$, α is the lower bound of K, and β is a constant of the order of 1.

4.4 Time discretization by operator splitting method

We use operator splitting method to discretize (4.12) in time. Among various splitting schemes, we adopt the famous Lie type splitting. Denote the time step by $\Delta t, t^n = n\Delta t$ and $u^n = u(n\Delta t)$. From $\{u^n, \mathbf{p}^n\}$ to $\{u^{n+1}, \mathbf{p}^{n+1}\}$ we evolve u and $\mathbf{p}$ in three steps: $\{u^n, \mathbf{p}^n\} \to \{u^{n+1/2}, \mathbf{p}^{n+1/2}\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$. The steps are described as follows:

Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot [(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u] + 2K(1 + |\nabla u|^2)^2 = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u = g \text{ on } \partial\Omega \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^n, \ \mathbf{p}(t^n) = \mathbf{p}^n, \end{cases} \tag{4.13}$$

and set

$$u^{n+1/2} = u(t^{n+1}), \ \mathbf{p}^{n+1/2} = \mathbf{p}^n. \tag{4.14}$$

Step 2:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/2} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/2}, \mathbf{p}(t^n) = \mathbf{p}^n, \end{cases} \tag{4.15}$$

and set

$$u^{n+1} = u^{n+1/2}, \mathbf{p}^{n+1} = P_+ \left[\mathbf{p}(t^{n+1})\right]. \tag{4.16}$$

In (4.15), $P_+(\cdot)$ is a projection operator which will be discussed later.

After this splitting, we still need to solve subproblems (4.13) and (4.15). There is no difficulty to solve (4.15) since we have the exact solution:

$$\mathbf{p}(t^{n+1}) = e^{-\gamma\Delta t}\mathbf{p}^n + \left(1 - e^{-\gamma\Delta t}\right)\mathbf{D}^2 u^{n+1}.$$

To solve (4.13), we use the semi-implicit scheme. We take the linear part of u implicitly and the nonlinear part of u explicitly. The treatment reads as follows:

$$\begin{cases} \frac{u^{n+1}-u^n}{\Delta t} - \nabla\cdot\left[\left(\varepsilon\mathbf{I} + \mathrm{cof}(\mathbf{p}^n)\right)\nabla u^{n+1}\right] + 2K(1+|\nabla u^n|^2)^2 = 0 \text{ in } \Omega\,, \\ u^{n+1} = g \text{ on } \partial\Omega. \end{cases}$$

Then our operator splitting scheme can be summarized as follows:

$$u^0 = u_0\,,\ \mathbf{p}^0 = \mathbf{p}_0\,. \tag{4.17}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as

$$\begin{cases} \frac{u^{n+1}-u^n}{\Delta t} - \nabla\cdot\left[\left(\varepsilon\mathbf{I} + \mathrm{cof}(\mathbf{p}^n)\right)\nabla u^{n+1}\right] + 2K(1+|\nabla u^n|^2)^2 = 0 \text{ in } \Omega, \\ u^{n+1} = g \text{ on } \partial\Omega, \end{cases} \tag{4.18}$$

$$\mathbf{p}^{n+1} = P_+\left[e^{-\gamma\Delta t}\mathbf{p}^n + \left(1 - e^{-\gamma\Delta t}\right)\mathbf{D}^2 u^{n+1}\right]. \tag{4.19}$$

4.5 Finite element implementation

The divergence form strongly suggests using finite element method to implement (4.18)-(4.19). Here we use the so called mixed finite element method: we use the same function space to approximate $u, \nabla u, \mathbf{D}^2 u$ and $\mathbf{p}$. We use the same finite element spaces in Chapter 3, piecewise affine functions. Details are the same in Chapter 3. Here we only mention how to approximate the first order derivatives using the same piecewise affine space.

4.5.1 Approximations of the two first order derivatives of u

For any $v \in V_h$, denote the approximation of $\frac{\partial v}{\partial x_i}$ by $D_{ih}(v)$ for $i = 1, 2$. We use the following relations:

$$\int_\Omega D_{ih}(v) w dx = \int_\Omega \frac{\partial v}{\partial x_i} w dx, i = 1, 2. \tag{4.20}$$

For practice importance, we state the calculations:
Denote the area of ω_k by $|\omega_k|$. Since w_k is only supported on ω_k, we have

$$\begin{cases} D_{ih}(v) \in V_h, \forall i = 1, 2, \\ D_{ih}(v)(Q_k) = \frac{3}{|\omega_k|} \int_{\omega_k} \frac{\partial v}{\partial x_i} w_k dx, \forall k = 1, 2, ..., N_h. \end{cases} \tag{4.21}$$

Remark 4.5.1. *On the regular mesh, (4.21) recovers the central difference approximation in the finite difference method on interior node and one sided approximation on boundary note.*

4.5.2 Implementation of scheme (4.17)-(4.19)

A fully discretized analogue of scheme (4.17)-(4.19) reads as:

$$u^0 = u_0 \in V_h, \mathbf{p}^0 = \mathbf{p}_0 \in \mathbf{Q}_h. \tag{4.22}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows Solve

$$\begin{cases} u^{n+1} \in V_{gh}, \\ \int_\Omega u^{n+1} v dx + \Delta t \int_\Omega \left(\varepsilon \mathbf{I} + \text{cof}(\mathbf{p}^n)\right) \nabla u^{n+1} \cdot \nabla v dx \\ \qquad = \int_\Omega u^n v dx - 2\Delta t K \int_\Omega (1 + |\nabla u^n|^2)^2 dx, \forall v \in V_{0h}, \end{cases} \tag{4.23}$$

and compute $\mathbf{p}^{n+1}$ via

$$\begin{cases} \forall k = 1, ..., N_h, \\ \mathbf{p}^{n+\frac{1}{2}}(Q_k) = e^{-\gamma \Delta t} \mathbf{p}^n(Q_k) + \left(1 - e^{-\gamma \Delta t}\right) \begin{pmatrix} D^2_{11h}\left(u^{n+1}\right)(Q_k) & D^2_{12h}\left(u^{n+1}\right)(Q_k) \\ D^2_{12h}\left(u^{n+1}\right)(Q_k) & D^2_{22h}\left(u^{n+1}\right)(Q_k) \end{pmatrix}, \\ \mathbf{p}^{n+1}(Q_k) = P_+\left[\mathbf{p}^{n+1/2}(Q_k)\right], \end{cases} \tag{4.24}$$

In (4.23), all integrations are computed using the trapezoidal rule. To compute the integration $\int_\Omega (1 + |\nabla u^n|^2)^2 dx$, we first compute ∇u^n using approximation (4.21) pointwise. Then we compute $(1+|\nabla u^n|^2)^2$ at all nodes and use trapezoidal rule to do the integration. In (4.24), $D^2_{ijh}(u^{n+1}), i, j = 1, 2$ is computed using approximation (3.38),(3.42).

4.6 Initialization

To initialize u_0 and $\mathbf{p}_0$, we solve the standard Monge-Ampère equation

$$\begin{cases} \det(\mathbf{D}^2 u_0) = K \\ u_0 = g \text{ on } \partial\Omega. \end{cases} \tag{4.25}$$

We use the method in [48] to solve (4.25). The method solves the initial value problem

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot ((\varepsilon \mathbf{I} + \text{cof}(\mathbf{p}))\nabla u) + 2K = 0, \\ u = g \text{ on } \partial\Omega, \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma(\mathbf{p} - \mathbf{D}^2 u) = \mathbf{0}, \end{cases} \tag{4.26}$$

to stead state. Denote the steady state of (4.26) by $\{u_*, \mathbf{p}_*\}$. We set $u_0 = u_*$. $\mathbf{p}_0$ is then computed as

$$\begin{cases} \mathbf{p}_0 = \mathbf{D}^2 u_0, \\ \frac{\partial \mathbf{p}}{\partial \mathbf{n}} = \mathbf{0}. \end{cases} \tag{4.27}$$

Our algorithm can be summarized as a two stage method:

Stage 1

In the algorithm in [48], set $\varepsilon_1 = \varepsilon_2 = h^2, dt = 2h^2$. Solve (4.26) until $\|u^{n+1} - u^n\|_2 < tol$ to get u_1. Compute $\mathbf{p}_1$ by (4.27).

Stage 2

With initial condition $u_1, \mathbf{p}_1$, solve (4.17)-(4.19) until steady state.

In the above algorithm, *tol* is some positive small number. The two-stage algorithm works but is inefficient. When we design this scheme, we want to use ε_1 and ε_2 of order $O(h^2)$. To use this setting and to make the two-stage algorithm converges, we need to set *tol* very small, say h^4, and we need to use a small time step in Stage-2, say h^2. To accelerate it, we propose a three stage algorithm:

Stage 1

In the algorithm in [48], set $\varepsilon_1 = \varepsilon_2 = h^2, dt = 2h^2$. Solve (4.26) until $\|u^{n+1} - u^n\|_2 < tol$ to get u_1. Compute $\mathbf{p}_1$ by (4.27).

Stage 2

With initial condition $u_1, \mathbf{p}_1$, set $\varepsilon_1 = \varepsilon_2 = h, dt = h^2$. Solve (4.17)-(4.19) until $\|u^{n+1} - u^n\|_2 < tol$ to get u_2. Compute $\mathbf{p}_2$ by (4.27).

Stage 3

With initial condition $u_2, \mathbf{p}_2$, set $\varepsilon_1 = \varepsilon_2 = h^2, dt = 2h^2$. Solve (4.17)-(4.19) until steady state.

4.7 Numerical experiments

In this section, we test the performance of our algorithm. Four different domains shown in Figure 3.1 are tested: (a) the regular mesh on a unit square, (b) the symmetric mesh on a unit square, (c) the unstructured mesh on a unit square and (d) the unstructured mesh on a half-unit circle. In all of our experiments, in stage 1, we use the method introduced in [48] to get the initial condition. We choose $\Delta t = 2h^2, \varepsilon_1 = \varepsilon_2 = h^2$ and $tol = h^3$. In stage 2, we use $\Delta t = 2h^2, \varepsilon_1 = \varepsilon_2 = h^2$.

4.7.1 Example 1

For the first example, we choose the exact solution u^* as a quadratic equation:

$$u^* = \alpha \left(x_1 - \frac{1}{2} \right)^2 + \frac{1}{\alpha} \left(x_2 - \frac{1}{2} \right)^2, \tag{4.28}$$

with boundary condition $g = u^*|_{\partial\Omega}$ and

$$K = \frac{4}{1 + 4\alpha \left(x_1 - \frac{1}{2} \right)^2 + \frac{4}{\alpha} \left(x_2 - \frac{1}{2} \right)^2}.$$

For all of the tests, we use the stopping criterion $\|u^{n+1} - u^n\| < 10^{-8}$.

In the first test, we choose $\alpha = 1$, then u^* is an isotropic function. The graph and error behavior of the 3-stage algorithm on different mesh is shown in Figure 4.1. The number of iteration and accuracy orders are shown in Table 4.1. The accuracy order for the L_2 and L_∞ norm is in general larger than 1.5. The performance comparison between the 2-stage algorithm and the 3-stage algorithm is shown in Table 4.2. We can see the L_2 and L_∞ errors are almost the same for both algorithms. But the number of iteration required by the 3-stage algorithm is only 60% of that of the 2-stage algorithm.

Then we use the two-stage algorithm with $\varepsilon_1 = \varepsilon_2 = 0$ in both stage. With this setting, the regularization effect is removed. For the quadratic function, the zero-Neumann boundary condition is exact for the three second order derivatives of u^*. So we expect our algorithm provides a more accurate result. Since we remove the regularization, we need a smaller time step. We choose $dt = h^2$ in both stage. On the unit square regular mesh with $h = 0.0354$, the error behavior is shown in Figure 4.2. We can see both L_2 and L_∞ errors converge to the machine precision.

In the third test, we choose $\alpha = 5$. Then u^* is highly anisotropic. We solve this problem on the unit square regular mesh by the three-stage algorithm. For this problem, we need a smaller stopping criterion, $\|u^{n+1} - u^n\|_2 < 10^{-9}$, to achieve the best accuracy. The number of iterations necessary to satisfy the stopping criterion and approximation error accuracy is shown in Table 4.3. Its graph of the computed solution and its error behavior is shown in Figure 4.3

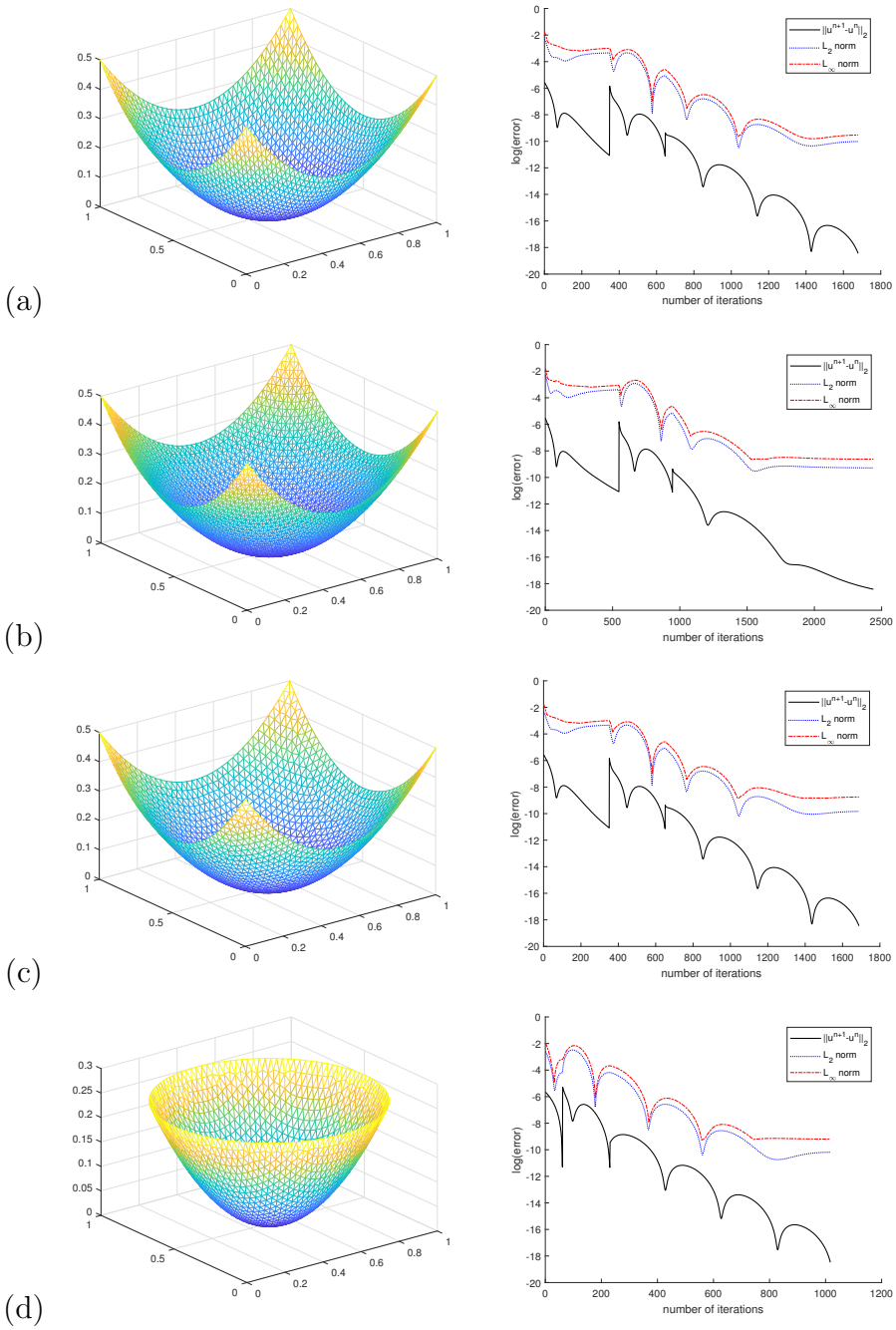


Figure 4.1: (Example 1, $\alpha = 1$) Graphs of the computed solution by the 3-stage algorithm and related error behavior on (a) unit square regular mesh, (b) unite square symmetric mesh, (c) unite square unstructured mesh, (d) half-unit circle triangulation.

(a)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1414	138	9.59×10^{-9}	6.07×10^{-4}		9.86×10^{-4}	
0.0707	476	9.91×10^{-9}	1.66×10^{-4}	1.87	2.72×10^{-4}	2.12
0.0354	1679	9.78×10^{-9}	4.47×10^{-5}	1.89	7.28×10^{-5}	1.64

(b)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	231	9.92×10^{-9}	1.37×10^{-3}		2.64×10^{-3}	
0.05	762	9.99×10^{-9}	3.54×10^{-4}	1.95	6.89×10^{-4}	1.94
0.025	2439	9.99×10^{-9}	9.30×10^{-5}	1.93	1.79×10^{-4}	1.94

(c)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	138	3.59×10^{-9}	8.30×10^{-4}		2.11×10^{-3}	
0.05	474	9.16×10^{-9}	1.68×10^{-4}	2.30	5.60×10^{-4}	1.91
0.025	1688	9.71×10^{-9}	5.38×10^{-5}	1.64	1.58×10^{-4}	1.83

(d)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	95	7.52×10^{-9}	6.10×10^{-4}		1.20×10^{-3}	
0.05	297	8.02×10^{-9}	1.64×10^{-4}	1.90	4.55×10^{-4}	1.40
0.025	1017	9.42×10^{-8}	3.77×10^{-5}	2.12	1.02×10^{-4}	2.16

Table 4.1: (Example 1, $\alpha = 1$) Number of iterations necessary to converge, approximation errors and accuracy orders on (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square unstructured mesh and (d) the half-unit disk mesh.

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	2811	9.65×10^{-9}	3.40×10^{-5}	5.79×10^{-5}
(ii)	0.0354	1679	9.78×10^{-9}	4.47×10^{-5}	7.28×10^{-3}
(iii)	0.025	4403	9.98×10^{-9}	9.32×10^{-5}	1.79×10^{-4}
(iv)	0.025	2439	9.99×10^{-9}	9.30×10^{-5}	1.79×10^{-4}

Table 4.2: (Example 1) Comparison of the two-stage and three-stage algorithms for the meshes of Figure 3.1(a) and 3.1(b): (i) Mesh (a) with the two-stage scheme. (ii) Mesh (a) with the three-stage strategy. (iii) Mesh (b) with the two-stage scheme. (iv) Mesh (b) with the three-stage strategy.

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1414	1321	9.96×10^{-10}	4.10×10^{-4}		6.63×10^{-4}	
0.0707	4514	9.98×10^{-10}	1.05×10^{-4}	1.97	1.66×10^{-4}	2.00
0.0354	15169	9.99×10^{-9}	2.73×10^{-5}	2.00	4.15×10^{-5}	2.00

Table 4.3: (Example 1, $\alpha = 5$) Number of iterations necessary to converge, approximation errors and accuracy orders on the unit square regular mesh.

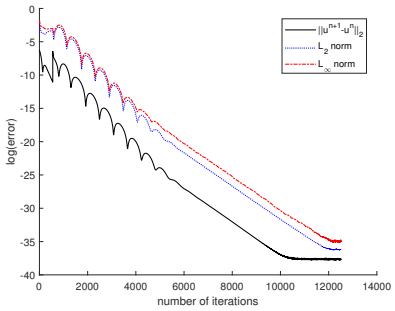


Figure 4.2: (Example 1, $\alpha = 1$) The error behavior by the two-stage algorithm with $\varepsilon_1 = \varepsilon_2 = 0$ in both stage on the unit square regular mesh .

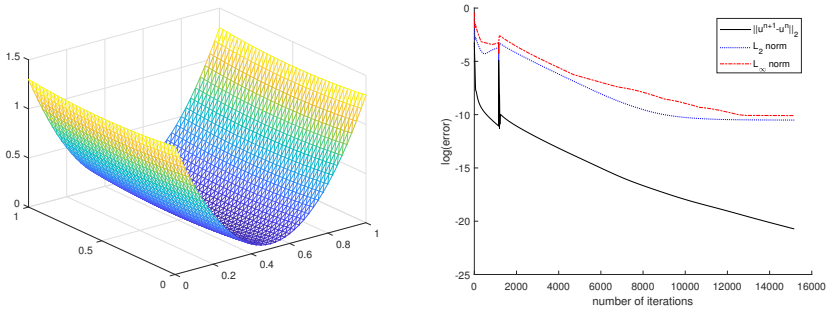


Figure 4.3: (Example 1, $\alpha = 5$) Graph of the computed result by the three-stage algorithm and the associated error behavior on the unit square regular mesh.

4.7.2 Example 2

In the second example, we choose the exact solution u^* as an exponential function:

$$u^* = e^{r^2}, \tag{4.29}$$

with $g = u^*|_\Omega$ and

$$K = \frac{4e^{2r^2}(1+2r^2)}{(1+4r^2e^{2r^2})^2}.$$

In (4.29), $r^2 = (x_1 - 1/2)^2 + (x_2 - 1/2)^2$. For this example, we use stopping criterion $\|u^{n+1} - u^n\|_2 < 10^{-7}$. The graph of the computed solution by the three-stage algorithm and spatial approximation error behavior is shown in Figure 4.4. The errors and our results are shown in Table 4.4. Generally the accuracy orders are larger than 1.5.

For this example, the comparison between the performance of the two-stage

algorithm and the three stage algorithm on the regular mesh and symmetric mesh is shown in Table 4.5. The L_2 and L_∞ errors are similar by the number of iteration required by the three-stage algorithm is almost half of that of the two-stage algorithm.

(a)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1414	130	6.09×10^{-8}	8.04×10^{-3}		1.52×10^{-2}	
0.0707	519	8.82×10^{-8}	2.78×10^{-3}	1.53	4.99×10^{-3}	1.61
0.0354	1963	9.80×10^{-8}	7.39×10^{-4}	1.91	1.32×10^{-3}	1.92

(b)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	176	9.99×10^{-8}	1.11×10^{-2}		1.95×10^{-2}	
0.05	575	9.77×10^{-8}	4.01×10^{-3}	1.47	6.66×10^{-3}	1.55
0.025	2430	9.47×10^{-8}	1.17×10^{-3}	1.78	1.88×10^{-3}	1.82

(c)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	129	6.05×10^{-8}	8.97×10^{-3}		1.66×10^{-2}	
0.05	516	9.32×10^{-8}	2.92×10^{-3}	1.62	5.47×10^{-3}	1.60
0.025	1961	9.51×10^{-8}	8.09×10^{-4}	1.85	1.48×10^{-3}	1.89

(d)

h	N	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	69	8.74×10^{-8}	5.30×10^{-3}		1.15×10^{-2}	
0.05	312	7.83×10^{-8}	1.78×10^{-3}	1.57	3.65×10^{-3}	1.66
0.025	1284	9.40×10^{-8}	4.88×10^{-4}	1.87	9.75×10^{-4}	1.90

Table 4.4: (Example 2) Number of iterations necessary to converge, approximation errors and accuracy orders on (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square unstructured mesh and (d) the half-unit disk mesh.

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	3853	9.90×10^{-8}	7.35×10^{-4}	1.31×10^{-3}
(ii)	0.0354	1963	9.80×10^{-8}	7.39×10^{-4}	1.32×10^{-3}
(iii)	0.025	4251	9.99×10^{-8}	1.14×10^{-3}	1.85×10^{-3}
(iv)	0.025	2430	9.47×10^{-8}	1.17×10^{-3}	1.88×10^{-3}

Table 4.5: (Example 2) Comparison of the two-stage and three-stage algorithms for the meshes of Figure 3.1(a) and 3.1(b): (i) Mesh (a) with the two-stage scheme. (ii) Mesh (a) with the three-stage strategy. (iii) Mesh (b) with the two-stage scheme. (iv) Mesh (b) with the three-stage strategy.

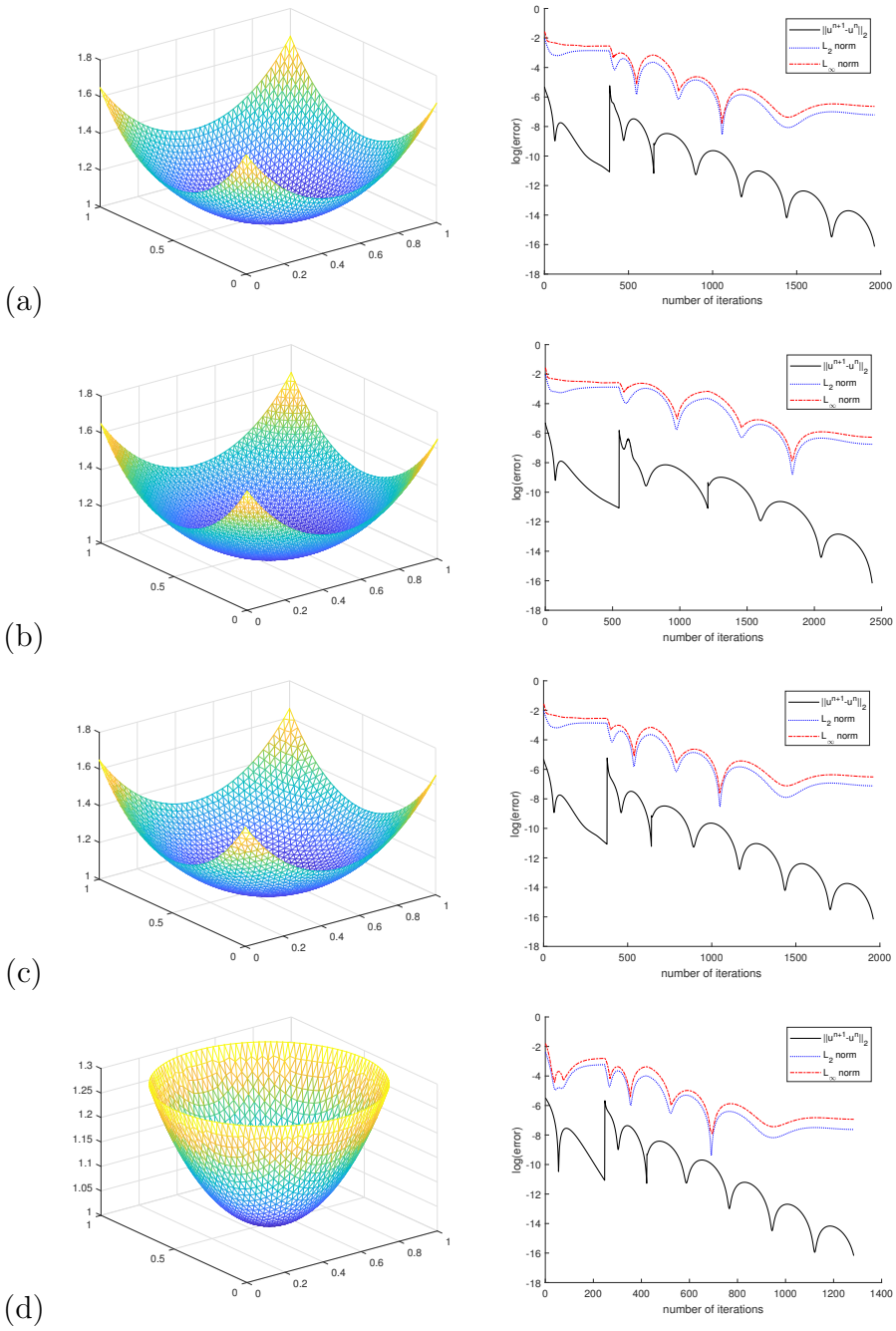


Figure 4.4: (Example 2) Graphs of the computed solution by the 3-stage algorithm and related error behavior on (a) unit square regular mesh, (b) unite square symmetric mesh, (c) unite square unstructured mesh, (d) half-unit circle triangulation.

4.7.3 Example 3

In this example,

$$\begin{cases} g = 0 \text{ on } \partial\Omega, \\ K = \lambda, \end{cases} \tag{4.30}$$

with λ varies from 0.1 to 1 with 0.1 increment each time. As λ increases, the curvature of the solution should goes larger, or the minimum value of the solution should go smaller. On the regular mesh with $h = 0.0354$, the figure of the solution of each λ is shown in Figure 4.5. In Figure 4.5, we also show the cross sections of some selected λ along $x_1 = x_2$ and $x_1 = 1/2$. The same to our expectation, as λ increases, the minimum value of the solution goes smaller.

4.8 Conclusion

In this work, we have proposed two algorithms to solve the Dirichlet Minkowski problem in dimension two. Our algorithms are easy to implement since only a linear PDE system is to be solved and the basis function is piecewise linear function. In our numerical experiments, when the exact solution is given, the accuracy order of the L_2 and L_∞ error is larger than 1.5. To achieve the same accuracy, the number of iterations needed by the three-stage algorithm is only 60 percent of that of the two-stage algorithm. This algorithm can be easily extended to high dimension problem.

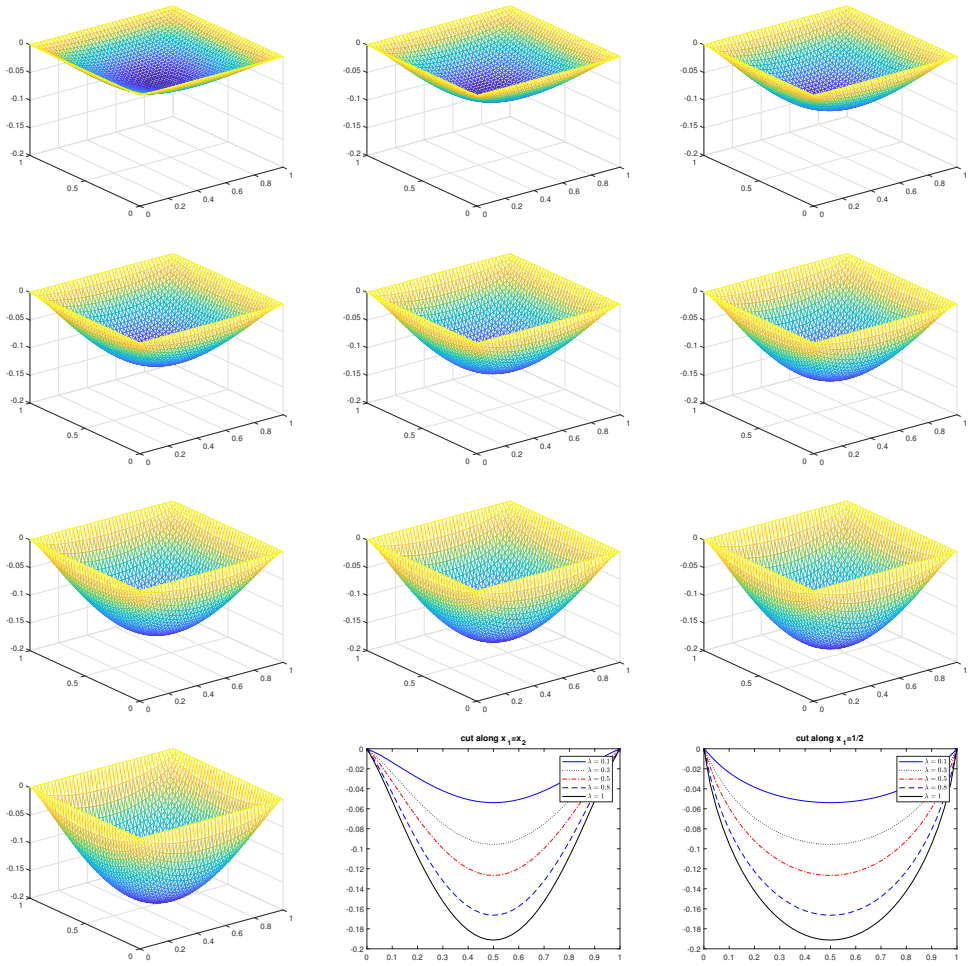


Figure 4.5: (Example 3) On the regular mesh with $h = 0.0354$, solution of each λ with λ increase from 0.1 to 1 and cross section along $x_1 = x_2$ and $x_1 = 1/2$.

Chapter 5

The Obstacle Monge-Ampère Equation

The obstacle problem for Monge-Ampère equation with Dirichlet boundary condition is defined as

$$\begin{cases} \det \mathbf{D}^2 u = f\chi(u > 0) \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{5.1}$$

where in (5.1):

- Ω is bounded convex domain of $\mathbb{R}^2$ and $\partial\Omega$ is the boundary of Ω.
- $f \in L^\infty(\Omega)$ and verifies $0 < \alpha \leq f \leq \beta < +\infty$.
- $\chi(u > 0)$ is the characteristic function of the set $\{x | x \in \Omega, u(x) > 0\}$.

The obstacle here is the plane $x_3 = 0$ and convex solution u is expected. This obstacle problem is studied by O. Savin in [103] and the regularity of the free boundary $\partial\{u = 0\}$ is proved to be $C^{1,\alpha}$ with α being positive small. With the existence of the characteristic function of u on the right hand side, u will always be non-negative. Applications of the obstacle problem for Monge-Ampère

equation can also be found in [103].

5.1 A divergence formulation of problem and an associated initial value problem

For the obstacle problem, an alternative formula to (5.1) is given as

$$\begin{cases} \det \mathbf{D}^2 u = \partial j_+(u) \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{5.2}$$

where $\partial j_+(u)$ is the sub-differential at u of the convex functional j_+ defined as

$$j_+(v) = \int_\Omega f v^+ dx.$$

with $v^+ = \max(0, v)$. Using the identity $v^+ = \frac{1}{2}(v + |v|)$, (5.2) can be further simplified as

$$\begin{cases} \det \mathbf{D}^2 u = \frac{1}{2}\left[f + \partial j(u)\right] \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{5.3}$$

where the convex functional $j(v)$ is defined by

$$j(v) = \int_\Omega f|v| dx.$$

Assume u has enough regularity, using the cofactor function, (5.3) can be written in the variational form

$$\begin{cases} u \in H^1(\Omega), u = g \text{ on } \partial\Omega, \\ \int_\Omega \operatorname{cof}(\mathbf{D}^2 u)\, \nabla u \cdot \nabla v dx + \int_\Omega f v dx + \langle \partial j(u), v\rangle = 0, \forall v \in H_0^1(\Omega). \end{cases} \tag{5.4}$$

To decouple differential operators and the Monge-Ampère nonlinearity, we take advantage of the equivalence between (5.3) and the following system:

$$\begin{cases} \begin{cases} -\nabla \cdot (\mathrm{cof}(\mathbf{p})\nabla u) + f + \partial j(u) \ni 0, \\ u = g \text{ on } \partial\Omega, \end{cases} \\ \mathbf{p} - \mathbf{D}^2 u = \mathbf{0}. \end{cases} \tag{5.5}$$

Since the elliptic operator in (5.3), (5.4), and (5.5) may degenerate at those parts of Ω where the graph of u touches the obstacle, we regularize the problem by addition of $-\varepsilon\nabla^2 u$(boundary layer considerations suggest taking ϵ of the order of h^2, h being a space discretizatoin step). After regularization, (5.5) becomes

$$\begin{cases} \begin{cases} -\varepsilon\nabla^2 u - \nabla \cdot (\mathrm{cof}(\mathbf{p})\nabla u) + f + \partial j(u) \ni 0, \\ u = g \text{ on } \partial\Omega, \end{cases} \\ \mathbf{p} - \mathbf{D}^2 u = \mathbf{0}. \end{cases} \tag{5.6}$$

In order to capture the solution of (5.6),we associate the following ***initial value problem*** (***flow*** in the Dynamic System terminology):

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot [(\varepsilon\mathbf{I} + \mathrm{cof}(\mathbf{p}))\, \nabla u] + f + \partial j(u) \ni 0, (\text{ for } t > 0), \\ u = g \text{ on } \partial\Omega \times (0, +\infty), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma\,(\mathbf{p} - \mathbf{D}^2 u) = \mathbf{0} \text{ in } \Omega \times (0, +\infty), \\ u(0) = u_0,\ \mathbf{p}(0) = \mathbf{p}_0, \end{cases} \tag{5.7}$$

with γ a positive constant (above and below, $\phi(t)$ denotes the function $x \to \phi(x,t)$). Problem (5.7) has a ***visco-elasticity*** flavor.

Concerning γ, the idea is to pick its value so that $\mathbf{p}$ evolves roughly at the same speed as u. Taking advantage of the fact that $\mathbf{p}$ and $\mathrm{cof}(\mathbf{p})$ have the same

eigenvalues we suggest to take

$$\gamma = \beta\lambda_0\sqrt{\alpha}$$

where β is some coefficient, λ_0 is the smallest eigenvalues of $-\nabla^2$ operator in $H_0^1(\Omega)$ and α is the lowest bound of the function f over Ω.

For the initial condition $\{u_0, \mathbf{p}_0\}$ we suggest to take u_0 the solution of the following linear Dirichlet problem:

$$\begin{cases} \nabla^2 u_0 = 2\lambda\sqrt{f} \text{ in } \Omega, \\ u_0 = g \text{ on } \partial\Omega, \end{cases} \tag{5.8}$$

with proper λ.

Concerning $\mathbf{p}_0$ an obvious choice is to take

$$\mathbf{p}_0 = \mathbf{D}^2 u_0. \tag{5.9}$$

An alternative choice being $\mathbf{p}_0 = \lambda\sqrt{f}\mathbf{I}$.

5.2 Time discretization by operator-splitting method

Given a function u, the operator splitting scheme of the ***Lie type*** is stated in the following(where $\Delta t(> 0)$ is a time-discretization step and $t^n = n\Delta t$):

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{5.10}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \left\{u^{n+1/3}, \mathbf{p}^{n+1/3}\right\} \to \left\{u^{n+2/3}, \mathbf{p}^{n+2/3}\right\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

Fractional Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n)\right) \nabla u\right] + f = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u = g \text{ on } \partial\Omega \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^n, \mathbf{p}(t^n) = \mathbf{p}^n \end{cases} \tag{5.11}$$

and set

$$u^{n+1/3} = u(t^{n+1}), \mathbf{p}^{n+1/3} = \mathbf{p}^n. \tag{5.12}$$

Fractional Step 2:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/3} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/2}, \end{cases} \tag{5.13}$$

and set

$$u^{n+2/3} = u^{n+1/3}, \mathbf{p}^{n+2/3} = \mathbf{p}(t^{n+1}). \tag{5.14}$$

Fractional Step 3:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} + \partial j(u) \ni 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+2/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+2/3}, \end{cases} \tag{5.15}$$

and set

$$u^{n+1} = u(t^{n+1}), \mathbf{p}^{n+1} = P_+\left(\mathbf{p}^{n+2/3}\right). \tag{5.16}$$

$P_+(\cdot)$ is the eigenvalue projection operator as before. Scheme (8.14)-(5.16) is first-order in time at best. To use this scheme, we have to solve the sub-initial value problems (5.11), (5.13) and (5.15). For (5.13), solving it is easy since it has a closed form solution. To discretize the other two, we advocate using the backward Euler scheme for better stability and larger time step. The resulting scheme reads as follows (using a more compact notation):

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{5.17}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] + f = 0 \text{ in } \Omega, \\ u^{n+1/2} = g \text{ on } \partial\Omega, \end{cases} \tag{5.18}$$

$$\mathbf{p}^{n+1} = e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2}, \tag{5.19}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\Delta t} + \partial j(u^{n+1}) \ni 0. \tag{5.20}$$

If the matrix-valued function $\mathbf{p}^n$ is positive semi-definite, the elliptic problem (5.18) is formally well-posed. To enforce the positivity of $\mathbf{p}^n$, we apply eigenvalue projection to (5.19). Then we have our first algorithm:

Algorithm OB1

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{5.21}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] + f = 0 \text{ in } \Omega, \\ u^{n+1/2} = g \text{ on } \partial\Omega, \end{cases} \tag{5.22}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{5.23}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\Delta t} + \partial j(u^{n+1}) \ni 0. \tag{5.24}$$

where $P_+(\cdot)$ is the projection operator which will be discussed later. The good news is that the unique solution of problem (5.24) has a closed form representation given by:

$$u^{n+1}(x) = u^{n+1/2}(x) \left(1 - \frac{\Delta t}{|u^{n+1/2}(x)|} f(x) \right)^+ \text{ a.e. in } \Omega. \tag{5.25}$$

It follows from (5.25) that $u^{n+1}(x) = 0$ if $\left|u^{n+1/2}\right| \leq \Delta t f(x)$.

Remark 5.2.1. *The rich structure of problem (5.7) implies that various operator-splitting schemes are applicable to its numerical solution. Among them, we would like to mention the following variant of scheme (5.21)-(5.24):*

Algorithm OB2

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{5.26}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] = 0 \ in \ \Omega, \\ u^{n+1/2} = g \ on \ \partial\Omega, \end{cases} \tag{5.27}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{5.28}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\Delta t} + \partial j(u^{n+1}) + f \ni 0, \tag{5.29}$$

The solution of (5.29) is given by

$$u^{n+1}(x) = \begin{cases} \max \left[u^{n+1/2}(x) - 2\Delta t f(x), 0 \right] & , \text{ if } u^{n+1/2} > 0, \\ u^{n+1/2} & , \text{ if } u^{n+1/2} \leq 0. \end{cases} \tag{5.30}$$

Comparing (5.30) to (5.25), the truncation region is shifted from $(-\Delta t f(x), \Delta t f(x))$ *to* $(0, 2\Delta t f(x))$.

Remark 5.2.2. *When we solve (5.24) in Algorithm OB1 using (5.25) or solve (5.29) in Algorithm OB2 using (5.30), the dependence of error on time step is obvious for points near the boundary. There are three reasons lead to this phenomenon:*

1. *the starting point* $u^{n+1/2}$ *is smooth(result by solving linear system (5.21) or (5.26)),*
2. *we are using a Dirichlet boundary condition,*
3. *only interior points' value are modified.*

The above three reasons make the discontinuity more obvious on the boundary.

The space discretization is the same as that mentioned in Chapter 3 and we do not repeat it here.

5.3 Numerical experiments

In this section, several examples are tested on different meshes: regular mesh on square, symmetric mesh on square, irregular mesh on square and irregular mesh on the unit disk. In our experiments, we choose $\lambda = 1, \gamma = 1/4$ with time step $dt = h^2/8$. Without specification, Algorithm 1 is used.

5.3.1 Example 1

In our first test problem, we use $f = 256$, g is defined by $g = u_0|_{\partial\Omega}$ where

$$u_0(x, y) = 8\left[\alpha\left(x - \frac{1}{2}\right)^2 + \frac{1}{\alpha}\left(y - \frac{1}{2}\right)^2\right] - 1.$$

α is a parameter no less than 1. We have $\det \mathbf{D}^2 u_0 = 256 (= f)$. u_0 is the exact solution to the standard Monge-Ampère equation without obstacle.

First we choose $\alpha = 1$. Then u_0 is an isotropic function. We did test of this problem on the four meshes listed in Figure (3.1). The results and error behavior are shown in Figure (5.1). The contour and plot of cross section of our results on a square with regular mesh and on a disk is shown in Figure 5.2. We can see our result is smooth and above the obstacle. Then we choose $\alpha = 1.2$ and test our algorithm on regular mesh and symmetric mesh. The results, contour and cross section is shown in Figure 5.3.

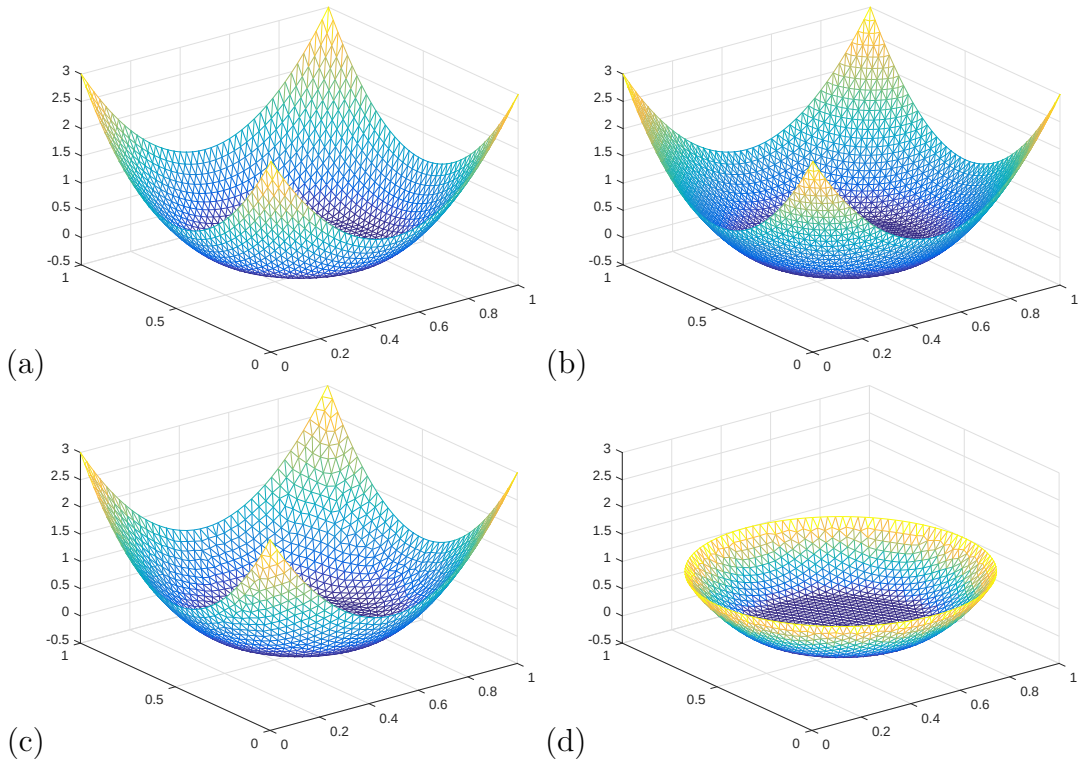


Figure 5.1: (Example 1, $\alpha = 1$) Computed results on (a) on a unit square with regular mesh, (b) on a unit square with symmetric mesh, (c) on a unit square with unstructured mesh and (d) on a half unit disk with unstructured.

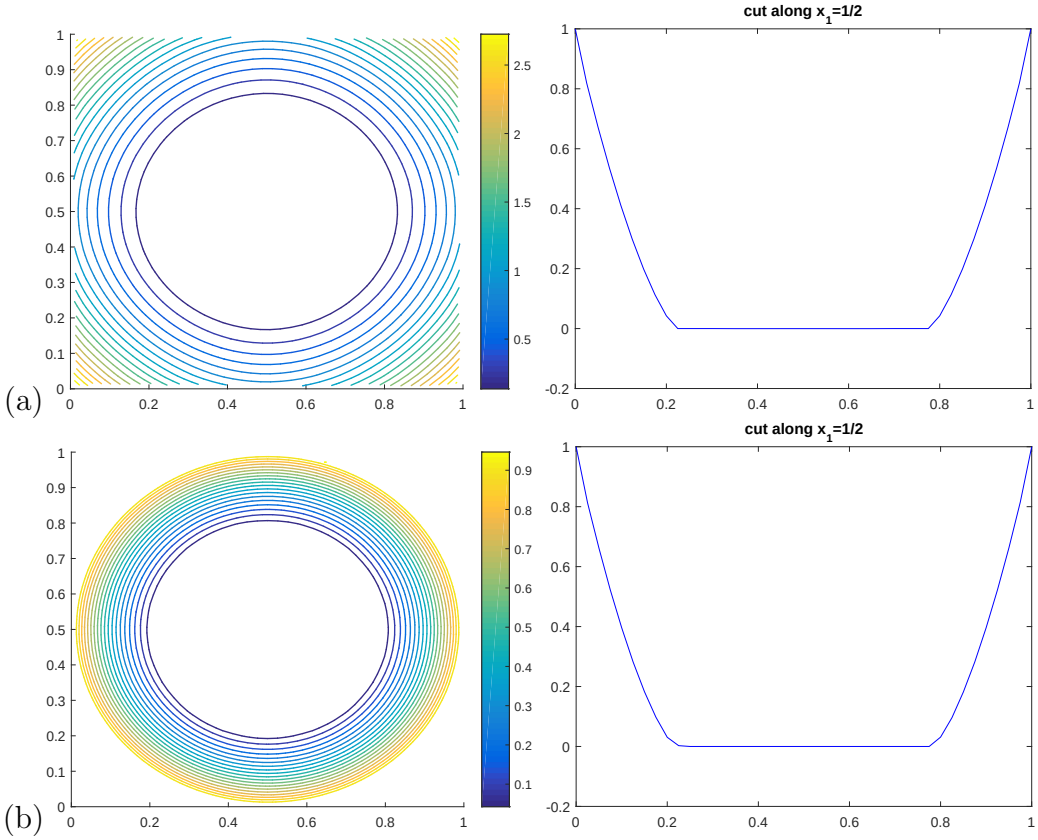


Figure 5.2: (Example 1, $\alpha = 1$) Contour and cross section along $x_1 = 1/2$ of the computed result on (a) a unit square with regular mesh and (b) a half unit disk with unstructured mesh.

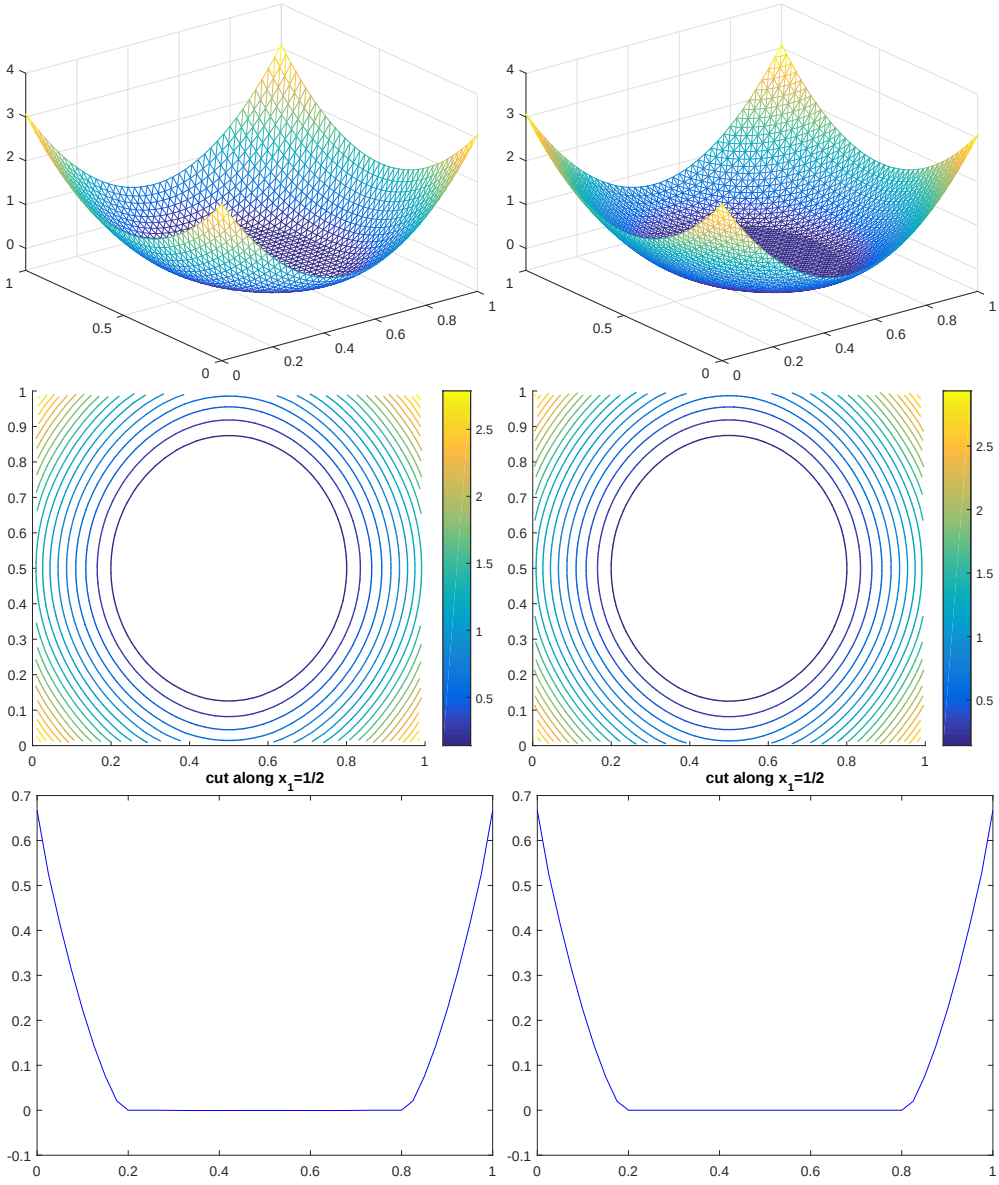


Figure 5.3: (Example 1, $\alpha = 1.2$) Computed results, contour and cross section on regular mesh and symmetric mesh. The first column is on a unit square with regular mesh. The second column is on a unit square with symmetric mesh.

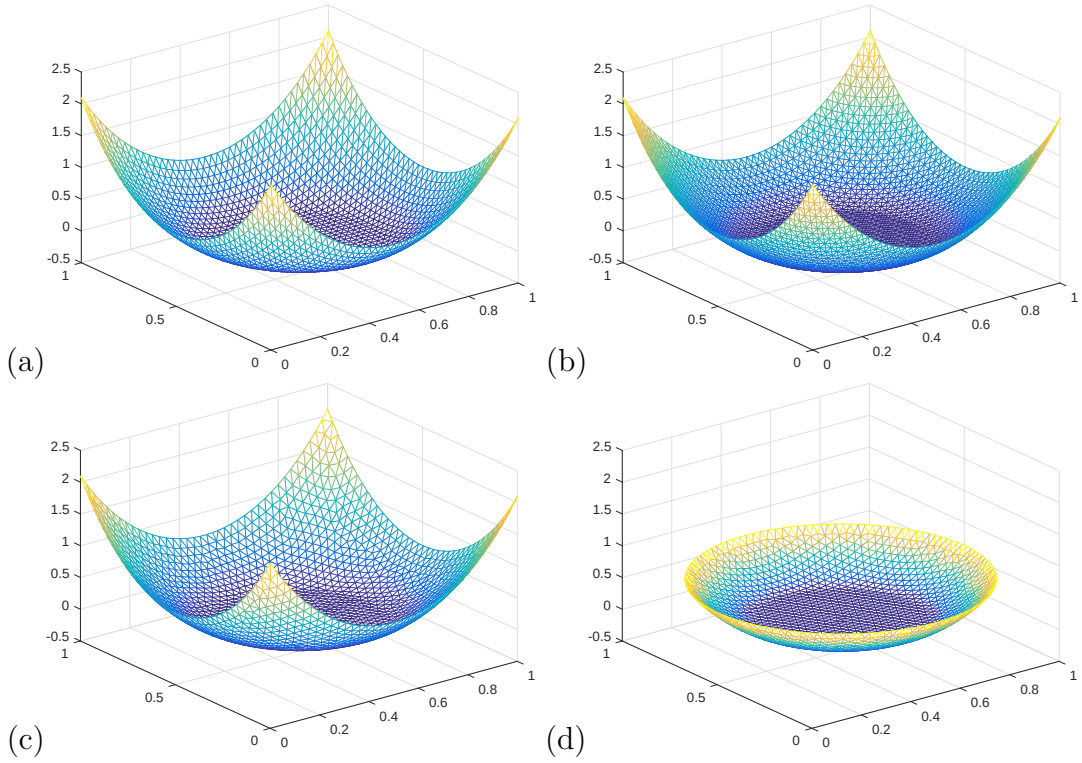


Figure 5.4: (Example 2) Computed results on (a) on a unit square with regular mesh, (b) on a unit square with symmetric mesh, (c) on a unit square with unstructured mesh and (d) on a half unit disk with unstructured.

5.3.2 Example 2

Compared to the first test problem the two modifications we are making are:

$$f(x,y) = 64e^{2r^2}\left(1+2r^2\right),$$

and

$$u_0(x,y) = 4e^{r^2} - \frac{9}{2},$$

with $r^2 = (x-1/2)^2 + (y-1/2)^2$. We have $\det \mathbf{D}^2 u_0 = f$. The result and error behavior of this example is shown in Figure (5.4). The time step is $\Delta t = h^2/8$. The contour and cross section on a square and circle is shown in Figure 5.5.

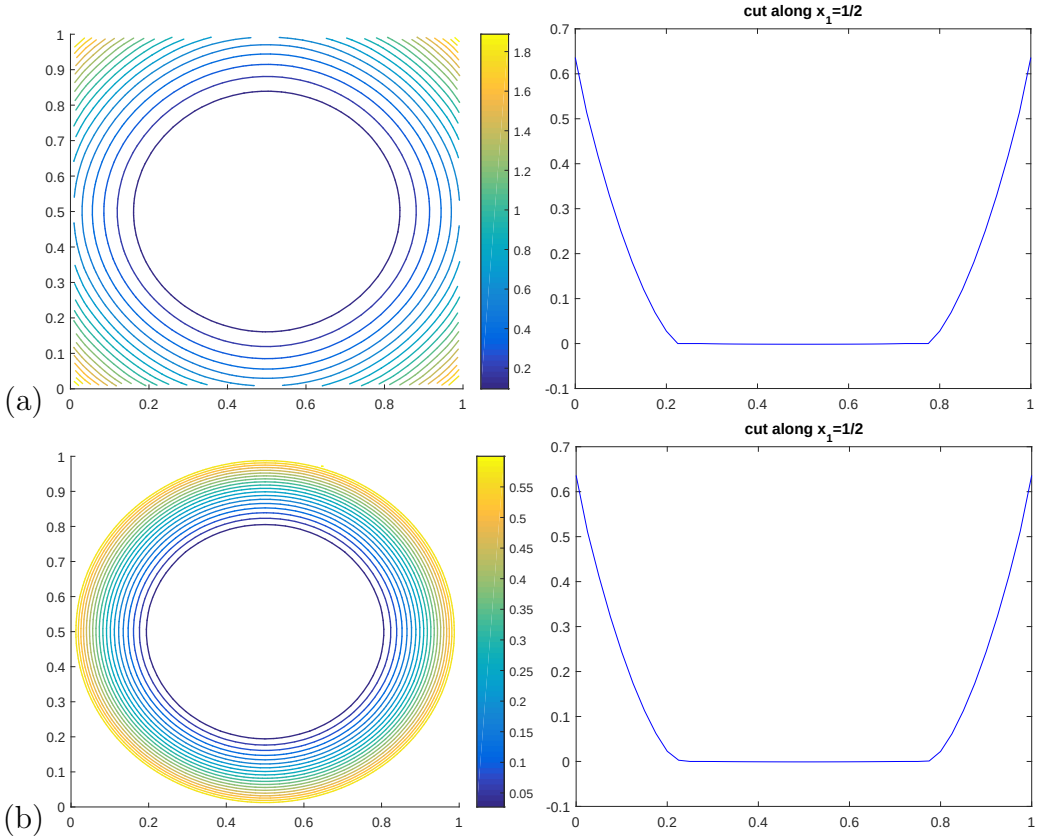


Figure 5.5: (Example 2) Contour and cross section along $x_1 = 1/2$ of the computed result on (a) a unit square with regular mesh and (b) a half unit disk with unstructured mesh.

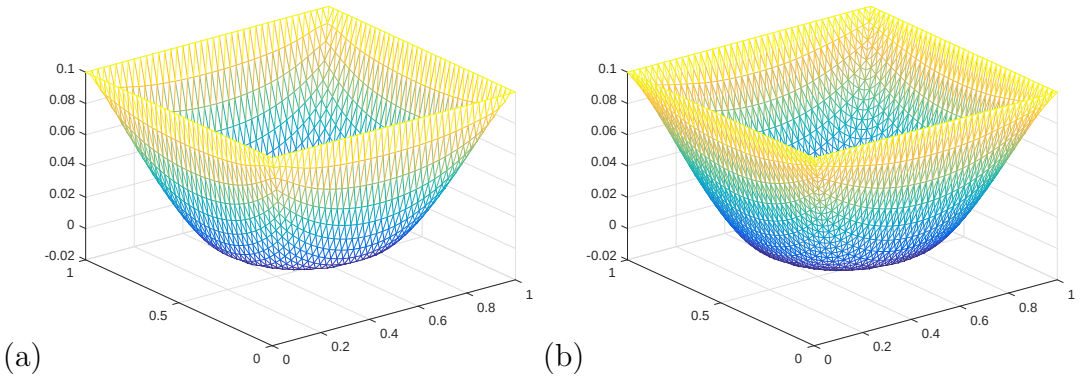


Figure 5.6: (Example 3) Computed results on (a) a unit square with regular mesh and (b) a unit square with symmetric mesh.

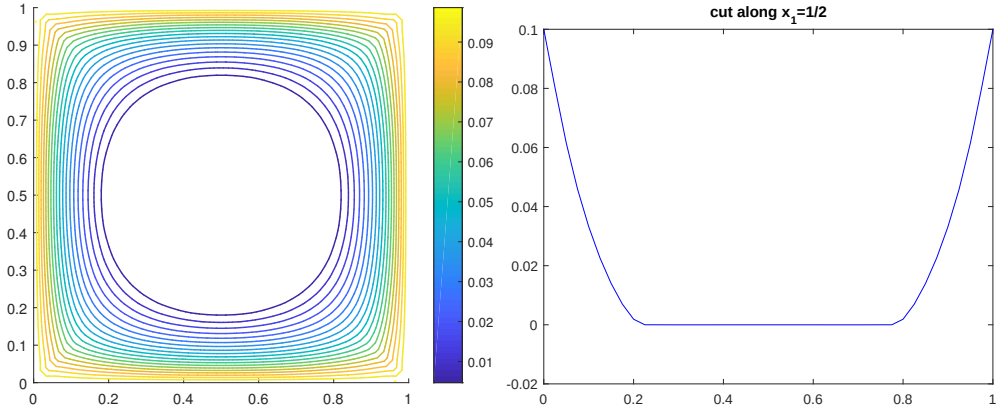


Figure 5.7: (Example 3) Contour and cross section along $x_1 = 1/2$ of the computed result on regular mesh.

5.3.3 Example 3

In this test, we use

$$f = 1 \text{ on } \Omega, g = 0.1 \text{ on } \partial\Omega.$$

Without obstacle, his problem has no classic solution but has a viscosity solution. Following the discussion in [48], we set $\epsilon = \epsilon_1 = 10^{-3}$. The result is shown in Figure 5.6. The contour and cross section on regular mesh is shown in Figure 5.7.

5.3.4 Example 4

We test

$$g = \frac{2\sqrt{2}}{3}\left(\left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2\right)^{3/4},$$
$$f = \frac{1}{\sqrt{(x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2}}.$$

In this example, f has a singular point at $\left(\frac{1}{2}, \frac{1}{2}\right)$. We approximate f at this point by $\frac{1}{h}$. The results by Algorithm OB1 on different mesh is shown in Figure 5.8. The contour and cross section on a square and on a disk is shown in Figure 5.9. Overall, the results are good. But from the cross section in Figure 5.9, there are some oscillations near the singular point. This oscillation disappear if we use Algorithm OB2. The cross section of the result by Algorithm OB2 on a square and on a disk is shown in Figure 5.9. Viewing from the bottom, the comparison of the results between the two algorithms are shown in Figure 5.11.

5.4 Conclusion

We have applied the operator-splitting finite element based method to solve the two dimensional obstacle Monge-Ampère problem with the Dirichlet boundary condition. According to different splitting ideas, two algorithms are introduced. If f and g are smooth, then both Algorithms provides good smooth result. The obstacle of the computed solution is the plane $x_3 = 0$ as expected. But if f has singularity in the obstacle region of the solution, Algorithm OB1 may have small oscillation in that region and Algorithm OB2 is better.

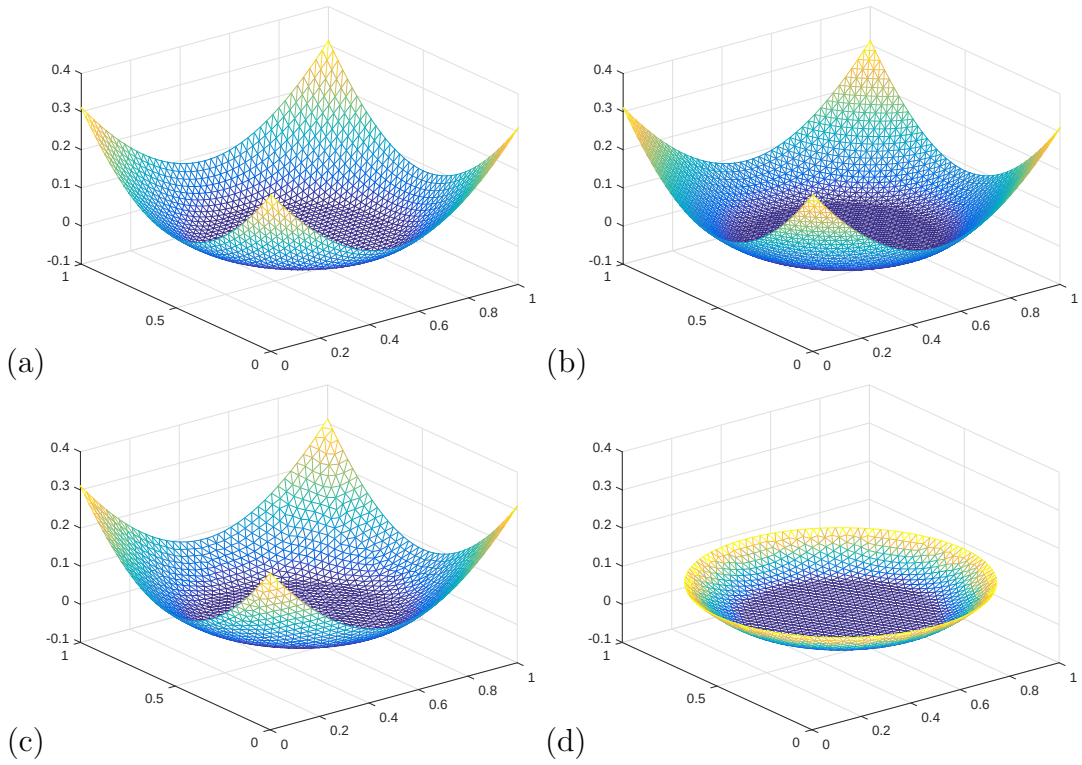


Figure 5.8: (Example 4) By Algorithm OB1, computed results on (a) on a unit square with regular mesh, (b) on a unit square with symmetric mesh, (c) on a unit square with unstructured mesh and (d) on a half unit disk with unstructured.

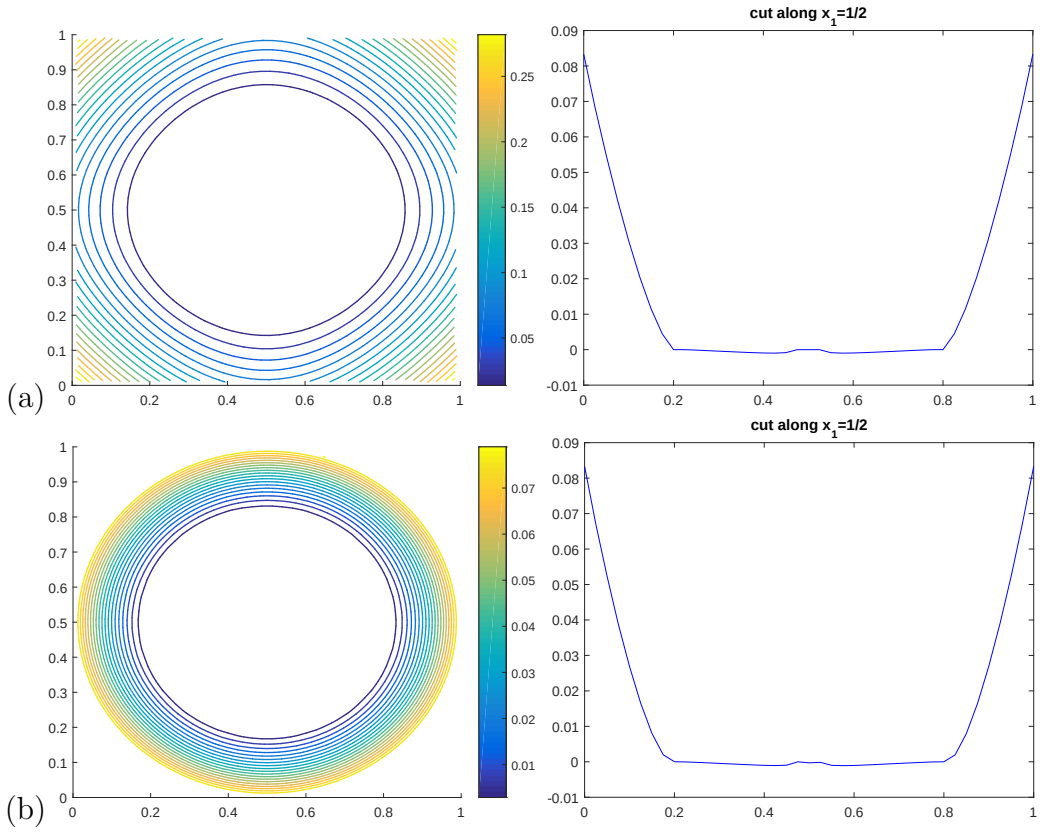


Figure 5.9: (Example 4) By Algorithm OB1, contour and cross section along $x_1 = 1/2$ of the computed result on (a) a unit square with regular mesh and (b) a half unit disk with unstructured mesh.

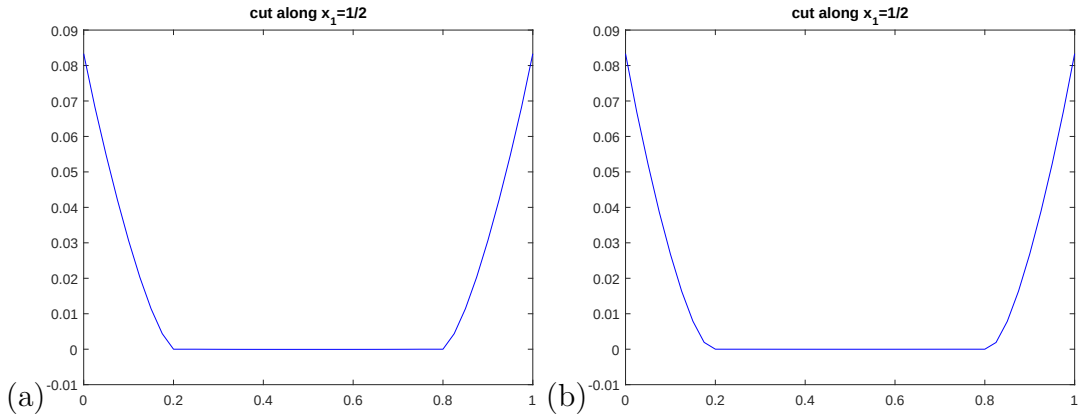


Figure 5.10: (Example 4) By Algorithm OB2, contour and cross section along $x_1 = 1/2$ on (a) a unit square with regular mesh and (b) a half unit disk with unstructured mesh.

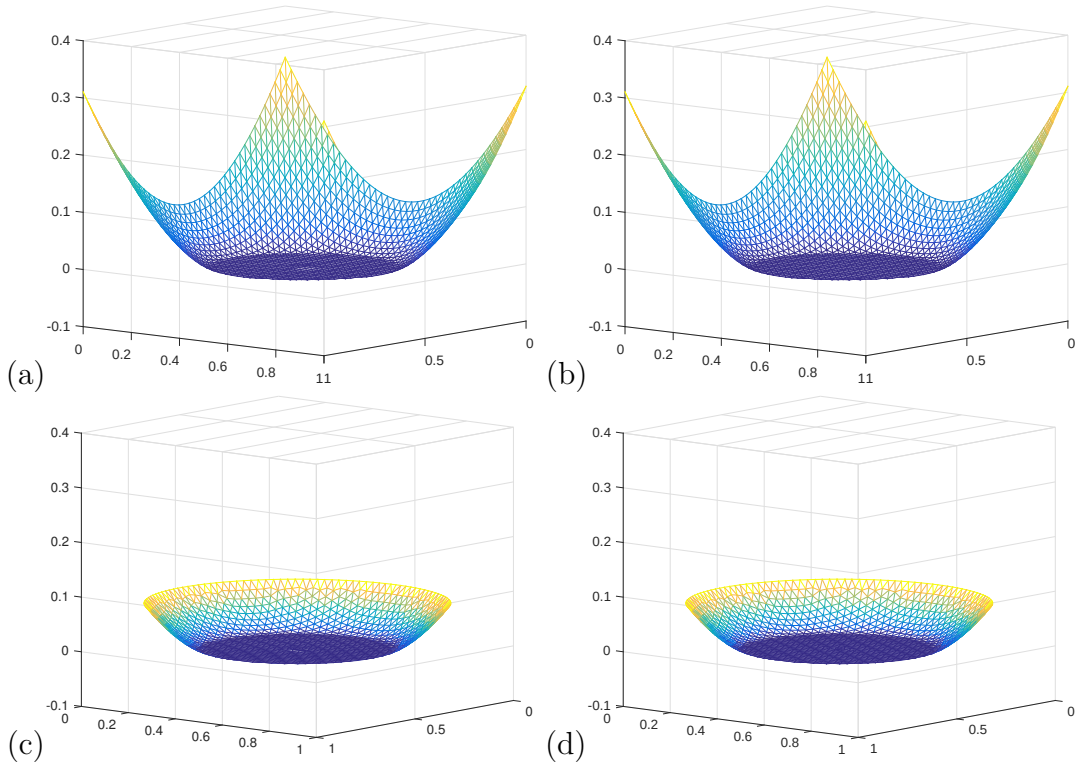


Figure 5.11: (Example 4) Computed results on different mesh. (a) Results by Algorithm OB1 on a unit square with regular mesh. (b) Results by Algorithm OB2 on a unit square with regular mesh. (c) Results by Algorithm OB1 on a half unit disk with unstructured mesh. (d) Results by Algorithm OB2 on a half unit disk with unstructured mesh.

Chapter 6

The Degenerate Monge-Ampère Equation

The degenerate Monge-Ampère equation reads as follows:

$$\begin{cases} \det \mathbf{D}^2 u = u^s \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{6.1}$$

with $0 < s < 2$. u is required to be a non-negative weakly convex function. Here Ω is assumed to be a strictly convex smooth domain. When the solution u is strictly positive, then (6.1) is strictly elliptic and the solution is proved to be C^∞ on Ω, see [17]. If $u = 0$ on a convex subdomain $\Lambda(u)$ of Ω, denote the free boundary by $\Gamma(u) = \partial\Lambda(u)$, it has been proved that u is C^∞ on $\Omega\backslash\Lambda(u)$. And the regularity of $\Gamma(u)$ and the optimal regularity of u up to the interface is proved in [25].

(6.1) is a more general equation of the obstacle problem (5.1). Letting $s \to 0^+$, (6.1) converges to (5.1) with $f = 1$.

6.1 A divergence formulation of problem and an associated initial value problem

Taking advantage of the equivalence between the canonical Monge-Ampère equation and the divergence form formulation, (6.1) can be written as

$$\begin{cases} -\nabla \cdot [\mathrm{cof}\,(\mathbf{D}^2 u)\,\nabla u] + 2u^s = 0 \text{ in } \Omega, \\ u = g \text{ on } \partial\Omega, \end{cases} \tag{6.2}$$

Introduce regularization term and the time derivative of u and $\mathbf{p}$, we get the initial value PDE system:

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot [(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}))\,\nabla u] + 2u^s = 0, (\text{ for } t > 0), \\ u = g \text{ on } \partial\Omega \times (0, +\infty), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma\,(\mathbf{p} - \mathbf{D}^2 u) = \mathbf{0} \text{ in } \Omega \times (0, +\infty), \\ u(0) = u_0,\ \mathbf{p}(0) = \mathbf{p}_0, \end{cases} \tag{6.3}$$

We use the same initial condition in the obstacle problem to initialize (6.3).

6.2 Time discretization by operator-splitting method

Problem (6.3) is time discretized by operator-splitting method. Each iteration consists of three sub-steps:

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{6.4}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \left\{u^{n+1/3}, \mathbf{p}^{n+1/3}\right\} \to \left\{u^{n+2/3}, \mathbf{p}^{n+2/3}\right\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

Fractional Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n) \right) \nabla u \right] = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u = g \text{ on } \partial\Omega \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^n, \mathbf{p}(t^n) = \mathbf{p}^n \end{cases} \tag{6.5}$$

and set

$$u^{n+1/3} = u(t^{n+1}), \mathbf{p}^{n+1/3} = \mathbf{p}^n. \tag{6.6}$$

Fractional Step 2:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/3} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/2}, \end{cases} \tag{6.7}$$

and set

$$u^{n+2/3} = u^{n+1/3}, \mathbf{p}^{n+2/3} = \mathbf{p}(t^{n+1}). \tag{6.8}$$

Fractional Step 3:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} + 2u^s = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+2/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+2/3}, \end{cases} \tag{6.9}$$

and set

$$u^{n+1} = u(t^{n+1}), \mathbf{p}^{n+1} = P_+ \left(\mathbf{p}^{n+2/3} \right). \tag{6.10}$$

$P_+(\cdot)$ is the eigenvalue projection operator as before. To discretize (6.4)-(6.10), we need to solve three PDE's: (6.5), (6.7) and (6.9). (6.5) and (6.7) appears in the algorithm of the obstacle problem and we use the same way to handle it. For (6.9), we use the backward Euler method to discretize it:

$$\frac{u^{n+1}-u^n}{\Delta t}+2(u^{n+1})^s=0 \tag{6.11}$$

Then the discretized algorithm is

$$u^0=u_0,\mathbf{p}^0=\mathbf{p}_0. \tag{6.12}$$

For $n\geq 0$, $\{u^n,\mathbf{p}^n\}\to u^{n+1/2}\to\{u^{n+1},\mathbf{p}^{n+1}\}$ as follows

$$\begin{cases}\frac{u^{n+1/2}-u^n}{\Delta t}-\nabla\cdot\left[\left(\varepsilon\mathbf{I}+\operatorname{cof}(\mathbf{p}^n)\right)\nabla u^{n+1/2}\right]=0 \text{ in } \Omega,\\ u^{n+1/2}=g \text{ on } \partial\Omega,\end{cases} \tag{6.13}$$

$$\mathbf{p}^{n+1}=P_+\left(e^{-\gamma\Delta t}\mathbf{p}^n+\left(1-e^{-\gamma\Delta t}\right)\mathbf{D}^2u^{n+1/2}\right), \tag{6.14}$$

$$\frac{u^{n+1}-u^{n+1/2}}{\Delta t}+2(u^{n+1})^s=0. \tag{6.15}$$

Concerning on the non-negative solution of (6.15) , we have the following theorem

Theorem 6.2.1. *If $u^{n+1/2}$ is non-negative, (6.11) has only one non-negative solution.*

Proof. The proof is simple. (6.11) can be rewritten as

$$2\Delta t(u^{n+1})^s+u^{n+1}-u^{n+1/2}=0.$$

Define $h(u^{n+1})=2\Delta t(u^{n+1})^s+u^{n+1}-u^{n+1/2}$. Then the non-negative solution of (6.11) is the non-negative root of $h(u^{n+1})$. We first prove that h has at most

one non-negative root. Computing the derivative of h we get

$$h'(u^{n+1}) = 2s\Delta t(u^{n+1})^{s-1} + 1.$$

So h' is positive for non-negative u^{n+1} and thus h is strictly increasing for non-negative u^{n+1}. h at most have one positive root.

Now we prove h has at least one non-negative root. Since $u^{n+1/2} \geq 0$ and we only care the value of h for $u^{n+1} \geq 0$, we have $h(0) = -u^{n+1/2} \leq 0$. From the above proof, we know $h(u^{n+1})$ is monotonic increasing for $u^{n+1} \geq 0$ and $h(u^{n+1}) \to +\infty$ as $u^{n+1} \to +\infty$, there must be a u^{n+1} such that $h(u^{n+1}) = 0$. So $h(u^{n+1})$ at least has one non-negative root. So $h(u^{n+1})$ has only one non-negative root. □

Since we want to find non-negative solution u, we enforce the non-negativity of $u^{n+1/2}$ by replacing (6.11) by

$$\frac{u^{n+1} - (u^{n+1/2})_+}{\Delta t} + 2(u^{n+1})^s = 0 \tag{6.16}$$

where $(u^{n+1/2})_+ = \max(u^{n+1/2}, 0)$. We use the Newton's method to find the positive root of (6.16) in each iteration. If Δt is small, u^{n+1} should not be too far away from u^n. So a good initialization of the Newton's method is $(u^{n+1/2})_+$. Incorporating the eigenvalue projection procedure on $\mathbf{p}$, the scheme is summarized as

Algorithm DG

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{6.17}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] = 0 \text{ in } \Omega, \\ u^{n+1/2} = g \text{ on } \partial\Omega, \end{cases} \tag{6.18}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{6.19}$$

$$\frac{u^{n+1} - (u^{n+1/2})_+}{\Delta t} + 2(u^{n+1})^s = 0. \tag{6.20}$$

6.3 Relation to the obstacle problem

In the above discussion, we assume $0 < s < 2$. Since the obstacle problem (with $f = 1$) discussed in Chapter 5 is the limit case of $s \to 0^+$, we analyze the limit of Algorithm DG in this section. We only need to discuss the limit case of the solution to (6.20).

(6.20) can be written as

$$u^{n+1} + 2\Delta t (u^{n+1})^s = (u^{n+1/2})_+. \tag{6.21}$$

If $u^{n+1/2} \leq 0$, then $(u^{n+1/2})_+ = 0$ and $u^{n+1} = 0$. If $u^{n+1/2} \geq 2\Delta t$, $u^{n+1} = u^{n+1/2} - 2\Delta t \geq 0$. If $0 < u^{n+1/2} < 2\Delta t$, for given $u^{n+1/2}$, let u^{n+1} satisfies (6.21). we have u^{n+1} goes to 0 as s goes to 0^+. So in this case we just assign $u^{n+1} = 0$.

Remark 6.3.1. *Given $u^{n+1} \geq 0$, we have*

$$\lim_{s \to 0^+} u^{n+1} + 2\Delta t (u^{n+1})^s = \begin{cases} 0, & \text{if } u^{n+1} = 0, \\ u^{n+1} + 2\Delta t, & \text{if } u^{n+1} > 0. \end{cases} \tag{6.22}$$

So actually there is no u^{n+1} satisfies (6.21) in the limit for $0 < u^{n+1/2} < 2\Delta t$. The above strategy has error of order $O(\Delta t)$.

Then the solution of (6.20) as $s \to 0^+$ can be summarized as follows

$$u^{n+1} = \begin{cases} u^{n+1/2} - 2\Delta t, & \text{if } u^{n+1/2} \geq 2\Delta t, \\ 0, & \text{if } u^{n+1/2} < 2\Delta t. \end{cases} \tag{6.23}$$

Recall the solution (5.30) (with $f = 1$) of the third step of Algorithm OB2 can be written as

$$u^{n+1}(x) = \begin{cases} u^{n+1/2} - 2\Delta t, & \text{if } 2\Delta t < u^{n+1/2}, \\ 0, & \text{if } 0 < u^{n+1/2} \leq 2\Delta t, \\ u^{n+1/2}, & \text{if } u^{n+1/2} \leq 0. \end{cases} \tag{6.24}$$

(6.23) is a truncated version of (6.24) to keep the strictly positivity of u^{n+1}.

The algorithm for the obstacle problem (5.1) induced from Algorithm DG is

Algorithm DGOB

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{6.25}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] = 0 \text{ in } \Omega, \\ u^{n+1/2} = g \text{ on } \partial\Omega, \end{cases} \tag{6.26}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{6.27}$$

$$u^{n+1} = \begin{cases} u^{n+1/2} - 2\Delta t f, & \text{if } u^{n+1/2} \geq 2\Delta t, \\ 0, & \text{if } u^{n+1/2} < 2\Delta t. \end{cases} \tag{6.28}$$

6.4 Numerical experiments

In this section, we test the performance of Algorithm DG and Algorithm DGOB. Four meshes shown in Figure 3.1 are used. In the first example, we choose $\varepsilon = \varepsilon_1 = h^2, \Delta t = 8h^2$. In the second example, we use $\varepsilon = \varepsilon_1 = h^2, \Delta t = h^2/8$.

6.4.1 Example 1

In the first example, we test the accuracy order of the algorithm. Since we have

$$\det(\mathbf{D}^2 u) = \frac{u' u''}{r}$$

where $u = u(r)$ with $r = |x - x_0|$. We choose

$$g = 180^{-\frac{3}{2}} r^6. \tag{6.29}$$

Then g is the exact solution to (6.29) with $k = 4/3$. The graph of the computed solutions on different mesh are shown in Figure 6.1. The number of iterations necessary to converge, L_2 and L_∞ errors and accuracy orders are shown in Table 6.1. Generally, our algorithm is better than first order.

(a)

h	N	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.2828	73	9.16×10^{-9}	6.30×10^{-4}		4.57×10^{-4}	
0.1414	434	9.86×10^{-9}	3.11×10^{-4}	1.02	2.42×10^{-4}	0.92
0.0707	1955	9.98×10^{-9}	1.16×10^{-4}	1.42	8.74×10^{-5}	1.47

(b)

h	N	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	73	9.34×10^{-9}	6.55×10^{-4}		4.72×10^{-4}	
0.05	435	9.89×10^{-9}	3.12×10^{-4}	1.07	2.47×10^{-4}	0.93
0.025	1961	9.99×10^{-9}	1.10×10^{-4}	1.50	8.32×10^{-5}	1.57

(c)

h	N	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	73	9.39×10^{-9}	6.40×10^{-4}		4.67×10^{-4}	
0.05	434	9.84×10^{-9}	3.15×10^{-4}	1.02	2.47×10^{-4}	0.92
0.025	1954	9.99×10^{-9}	1.17×10^{-4}	1.42	8.76×10^{-5}	1.50

(d)

h	N	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
0.1	66	9.03×10^{-9}	3.62×10^{-4}		2.48×10^{-4}	
0.05	414	9.91×10^{-9}	1.90×10^{-4}	0.93	1.35×10^{-4}	0.88
0.025	1965	9.99×10^{-9}	4.14×10^{-5}	2.20	3.33×10^{-5}	2.02

Table 6.1: (Example 1) Number of iterations necessary to converge, approximation errors and accuracy orders on (a) the square with regular mesh, (b) the square with symmetric mesh, (c) the square unstructured mesh and (d) the unit disk with unstructured mesh.

6.4.2 Example 2

In the second example, we test Algorithm DGOB to see its performance to solve the obstacle problem. We set $f = 256, g = u_0|_{\partial\Omega}$ where

$$u_0(x,y) = 8\left[\alpha\left(x-\frac{1}{2}\right)^2 + \frac{1}{\alpha}\left(y-\frac{1}{2}\right)^2\right] - 1.$$

This is the first test example in Chapter 5. The computed results by Algorithm DGOB on different meshes are shown in Figure 6.2. The performance is as good

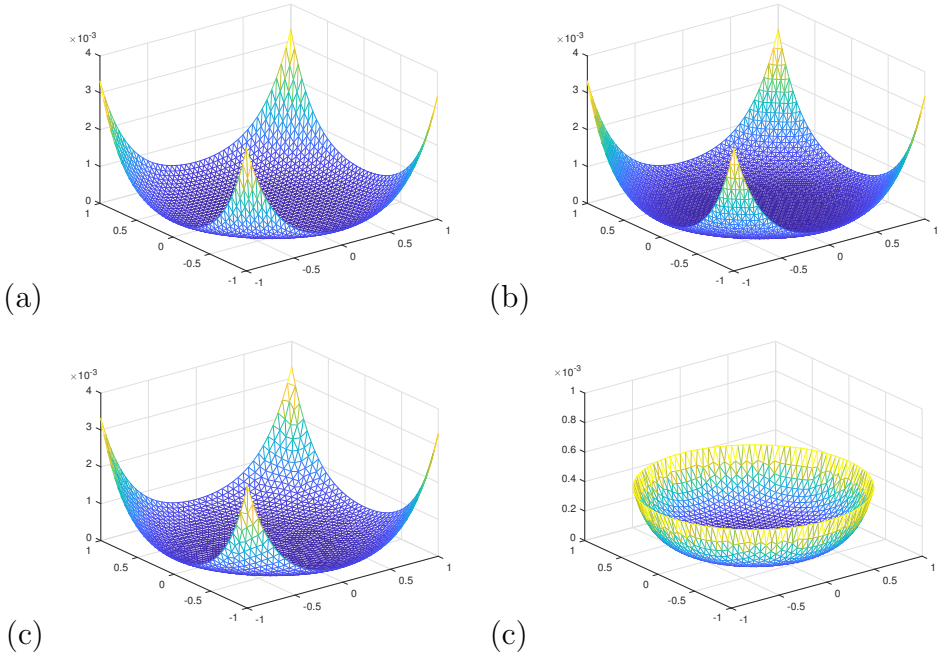


Figure 6.1: (Example 1) Computed results on (a) the square with regular mesh, (b) the square with symmetric mesh, (c) the square unstructured mesh and (d) the unit disk with unstructured mesh.

as Algorithm OB1.

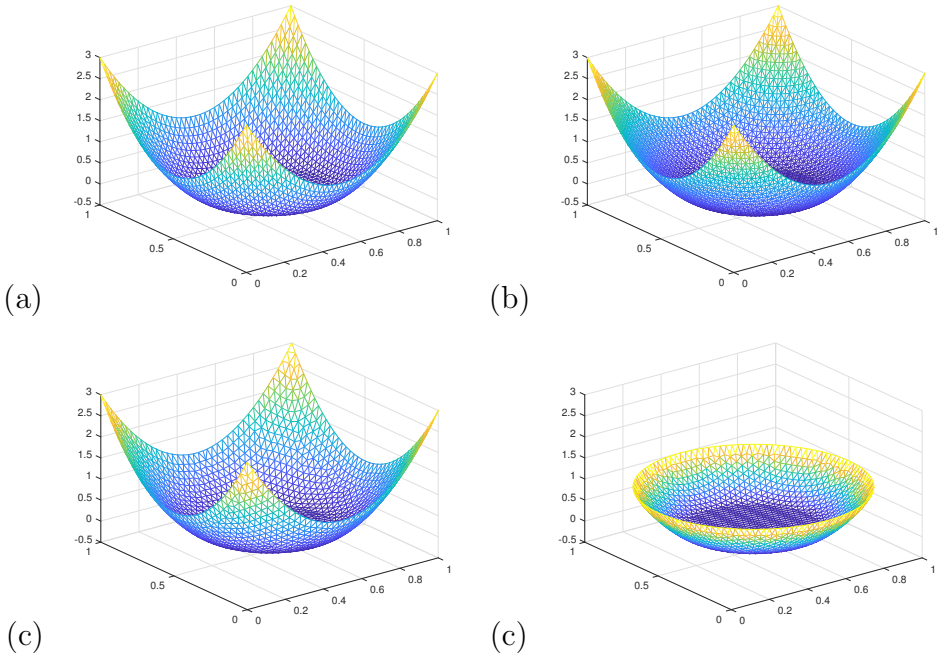


Figure 6.2: (Example 1) Computed results on (a) the square with regular mesh, (b) the square with symmetric mesh, (c) the square unstructured mesh and (d) the unit disk with unstructured mesh.

6.5 Conclusion

In this chapter, we have developed and operator-splitting finite element based method, Algorithm DG, to solve the degenerate Mogne-Ampère equation. Our method is easy to implement and the L_2 and L_∞ accuracy order are better than 1. For the extreme case of the degenerate equation, the obstacle problem, another algorithm, Algorithm DGOB, is also designed in the sense of the limit case of Algorithm DG. This new algorithm can be taken as an truncated version of Algorithm OB2 in Chapter 5. Its performance is similar to our results in Chapter 5.

Chapter 7

The Monge-Ampère Equation with the Neumann Boundary Condition

7.1 Introduction

In previous chapters, we introduced operator-splitting algorithm for various Monge-Ampère problems with the Dirichlet boundary condition. In this chapter, we consider the two dimensional Monge-Ampère equation with the Neumann boundary condition. As we mentioned before, Monge-Ampère equation arises in differential geometry and optimal transport. The Minkowski problem is a problem in differential geometry and it has the Dirichlet boundary condition. The optimal transport has a similar formulation but with the Neumann boundary condition.

The L_2 optimal transport problem is as follows: Given two density ρ_1, ρ_2 in $\mathbb{R}^d, d \geq 1$. The L_2 optimal transport is a map $\mathbf{y} = \phi(\mathbf{x}), \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ transport ρ_1

to ρ_2 and minimize the cost functional

$$c(\phi) = \int_{\mathbb{R}^d} |\phi(\mathbf{x}) - \mathbf{x}|^2 \rho_1 d\mathbf{x}. \tag{7.1}$$

Since $\phi(\mathbf{x})$ transport a density distribution to another one, it should be mass conservative:

$$\int_{\phi^{-1}(\Omega)} \rho_1(\mathbf{x}) d\mathbf{x} = \int_{\Omega} \rho_2(\mathbf{y}) d\mathbf{y}, \forall \Omega \in \mathbb{R}^d.$$

If ϕ is bijective, it can be shown that ϕ satisfies

$$\rho_2(\phi(\mathbf{x})) \det\left(\frac{\partial \phi}{\partial \mathbf{x}}\right) = \rho_1(\mathbf{x}), \mathbf{x} \in \Omega \tag{7.2}$$

where $\frac{\partial \phi}{\partial \mathbf{x}}$ is the Jacobian matrix of ϕ and Ω is a set containing the support of ρ_1 and ρ_2. The existence and properties of the solution is discussed in [61, 10] and is summarized in the following theorem:

Theorem 7.1.1. *There is a unique optimal transport mapping $\phi(\mathbf{x})$ from ρ_1 to ρ_2. Moreover, there is a convex potential $u(\mathbf{x})$ such that $\phi(\mathbf{x}) = \nabla u(\mathbf{x})$ and problem (7.2) can be written as a Monge-Amère equation*

$$\det(D^2 u) = \frac{\rho_1}{\rho_2(\nabla u)}. \tag{7.3}$$

The optimal transport has been applied on various problems. Since most assumptions are made on the mapping ϕ instead of the potential u, when we solve (7.3), the boundary conditions are always the Neumann boundary condition. In this chapter, we introduce an operator-splitting algorithm for the simplest Monge-Ampère equation. The algorithm can be easily extended to more general problems.

7.2 Two initial value PDE systems with divergence form

In this work, we consider the canonical two dimensional Monge-Ampère problem with the Neumann boundary condition:

$$\begin{cases} \det \mathbf{D}^2 u = f, \\ \frac{\partial u}{\partial \mathbf{n}} = l \text{ on } \Omega. \end{cases} \tag{7.4}$$

In (7.4), Ω is our computational domain, f is some positive function, $\mathbf{n}$ is the outward normal vector of Ω, l is some given function defined on $\partial\Omega$ and $\mathbf{D}^2 u$ is the Hessian matrix of u:

$$\mathbf{D}^2 u = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 \partial x_2} \\ \frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_2^2} \end{pmatrix}.$$

To solve (7.4), we recall the divergence initial value PDE system for the two dimensional canonical Monge-Ampère equation with the Dirichlet boundary condition (3.5), but replace the boundary condition by the Neumann one:

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}) \right) \nabla u \right] + 2f = 0 \text{ in } \Omega \times (0, +\infty), \\ \frac{\partial u}{\partial \mathbf{n}} = l \text{ on } \partial\Omega \times (0, +\infty), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \left(\mathbf{p} - \mathbf{D}^2 u \right) = \mathbf{0} \text{ in } \Omega \times (0, +\infty), \\ u(0) = u_0, \ \mathbf{p}(0) = \mathbf{p}_0, \end{cases} \tag{7.5}$$

Remark 7.2.1. *The idea of (7.5) is the same to our previous algorithm. But we cannot directly use integration by parts to implement the Neumann boundary condition since we have a matrix multiplication inside the divergence form. Our strategy is introduced in the following subsections.*

7.2.1 The first formulation

In (7.5), it is difficult to implement the boundary condition using integration by parts, as what has been used by most traditional finite element based algorithm. Our first try is to replace the Neumann boundary problem in (7.5) by two time dependent Dirichlet boundary problem:

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot ((\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}))\, \nabla u) = -2f \text{ in } \mathring{\Omega}, \\ \frac{\partial u}{\partial t} + \eta(\nabla u \cdot \mathbf{n} - l) = 0 \text{ on } \partial\Omega, \\ \frac{\partial p}{\partial t} + \gamma\,(\mathbf{p} - \mathbf{D}^2 u) = 0. \end{cases} \tag{7.6}$$

Here we use $\mathring{\Omega}$ to denote the interior area of Ω. The general idea is to solve (7.6) iteratively to steady state. In each iteration, u is updated in the prey-predator sense: first fix the value of u on boundary and update its interior value by solving the first PDE for Δt, the discrete time step. This is the first Dirichlet problem. Then fix the interior value of u and update its boundary value by solving the second PDE for Δt. This is the second Dirichlet problem. Details will be discussed in the next section.

Remark 7.2.2. *By this formulation, since the boundary value of u evolves in time, it may not strictly satisfy the Neumann boundary condition.*

7.2.2 The second formulation

In some applications of the Monge-Ampère equation, u is the potential of some quantity and what one cares is ∇u instead of u. Sometimes the condition $\frac{\partial u}{\partial \mathbf{n}} = l$ must be satisfied, for example in the moving mesh application, nodes (represented as ∇u) on the boundary are assumed to move along the boundary. In the second formulation, we want the Neumann boundary condition to be strictly satisfied.

The simplest way to achieve that is to replace the second equation in (7.6) by

$$\frac{\partial u}{\partial \mathbf{n}} = l. \tag{7.7}$$

By using (7.7), we enforce u to strictly satisfy the Neumann boundary condition after each iteration. Then the PDE system becomes

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left(\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}) \right) \nabla u \right) = -2f \text{ in } \mathring{\Omega}, \\ \nabla u \cdot \mathbf{n} = l \text{ on } \partial\Omega, \\ \frac{\partial p}{\partial t} + \gamma \left(\mathbf{p} - \mathbf{D}^2 u \right) = 0. \end{cases} \tag{7.8}$$

The Neumann boundary problem of u is now converted to a time-dependent and a time-independent Dirichlet boundary problem.

7.3 Time discretization by operator-splitting method

We first show the discretized scheme for (7.6). Each iteration is decomposed into three steps:

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{7.9}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \left\{u^{n+1/3}, \mathbf{p}^{n+1/3}\right\} \to \left\{u^{n+2/3}, \mathbf{p}^{n+2/3}\right\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

Fractional Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n) \right) \nabla u \right] + 2f = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u = u^n \text{ on } \partial\Omega \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^n, \mathbf{p}(t^n) = \mathbf{p}^n \end{cases} \tag{7.10}$$

and set

$$u^{n+1/3} = u(t^{n+1}), \mathbf{p}^{n+1/3} = \mathbf{p}^n. \tag{7.11}$$

Fractional Step 2:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} + \eta(\nabla u \cdot \mathbf{n}) = l \text{ in } \partial\Omega \times (t^n, t^{n+1}), \\ u = u^{n+1/3} \text{ on } \mathring{\Omega} \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/3}, \end{cases} \tag{7.12}$$

and set

$$u^{n+2/3} = u(t^{n+1}), \mathbf{p}^{n+2/3} = \mathbf{p}(t^{n+1}). \tag{7.13}$$

Fractional Step 3:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/3} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/2}, \end{cases} \tag{7.14}$$

and set

$$u^{n+1} = u(t^{n+1}), \mathbf{p}^{n+1} = P_+ \left(\mathbf{p}(t^{n+1}) \right). \tag{7.15}$$

$P_+(\cdot)$ is the eigenvalue projection operator as before. Steps (7.9)-(7.15) can be discretized in usual ways as those previous discussed algorithms. Since the solution of (7.4) is not unique, we need to fix some point to make sure our algorithm converge to some solution. So we choose a point $\mathbf{x}_0$ to make sure $u(\mathbf{x}_0) = 0$. To achieve that, after each iteration, we subtract $u^{n+1}(\mathbf{x}_0)$ from u^{n+1}: $u^{n+1} = u^{n+1} - u^{n+1}(\mathbf{x}_0)$. The resulting algorithm is summarized as follows:

Algorithm NU1

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{7.16}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] + 2f = 0 \text{ in } \Omega, \\ u^{n+1/2} = u^n \text{ on } \partial\Omega, \end{cases} \tag{7.17}$$

$$\begin{cases} \frac{u^{n+1}-u^{n+1/2}}{\Delta t} + \eta(\nabla u^{n+1} \cdot \mathbf{n}) = l \text{ in } \partial\Omega, \\ u^{n+1} = u^{n+1/2} \text{ on } \mathring{\Omega}, \end{cases} \tag{7.18}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{7.19}$$

$$u^{n+1} = u^{n+1} - u^{n+1}(\mathbf{x}_0). \tag{7.20}$$

To discretize the second system (7.8), we only need to replace the second sub-step (7.12) of the first system by

---

Fractional Step 2 of the Second System:

Solve

$$\begin{cases} \nabla u^{n+1/3} \cdot \mathbf{n} = l \text{ in } \partial\Omega, \\ u = u^{n+1/3} \text{ on } \mathring{\Omega}, \frac{\partial \mathbf{p}}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/3}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/3}, \end{cases} \tag{7.21}$$

and set

$$u^{n+2/3} = u(t^{n+1}), \mathbf{p}^{n+2/3} = \mathbf{p}(t^{n+1}). \tag{7.22}$$

The second algorithm is

Algorithm NU2

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{7.23}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to u^{n+1/2} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1/2}-u^n}{\Delta t} - \nabla \cdot \left[(\varepsilon \mathbf{I} + \text{cof}(\mathbf{p}^n)) \nabla u^{n+1/2} \right] + 2f = 0 \text{ in } \Omega, \\ u^{n+1/2} = u^n \text{ on } \partial\Omega, \end{cases} \tag{7.24}$$

$$\begin{cases} \nabla u^{n+1} \cdot \mathbf{n} = l \text{ in } \partial\Omega, \\ u^{n+1} = u^{n+1/2} \text{ on } \mathring{\Omega}, \end{cases} \tag{7.25}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right), \tag{7.26}$$

$$u^{n+1} = u^{n+1} - u^{n+1}(\mathbf{x}_0) \tag{7.27}$$

7.4 Finite element implementation

We use the same finite element space as previous sections. The discretization of the first and third step of the two algorithms, (7.17), (7.19), and (7.25), (7.26),

are the same to some previous discussed algorithms. Here we only discuss how to implement the two second steps (7.18) and (7.25).

The key point is to implement the dot product. Then a linear system can be formed and solved implicitly. The method we use here in some sense is a kind of mixed finite element method. In general, the first derivatives of u is approximated by piecewise linear polynomials. Use $\{b_1, b_2\}$ to denote the approximation of $\{u_x, u_y\}$:

$$\begin{cases} \int_\Omega b_1(v)wdx = \int_\Omega u_x(v)wdx, \\ \int_\Omega b_2(v)wdx = \int_\Omega u_y(v)wdx, \end{cases} \tag{7.28}$$

for any $w \in H^1(\Omega)$. v can be any nodes in our discretization and w is basis function. So $\{b_1, b_2\}$ are linear combinations of u on the nodes. Then $\nabla u \cdot \mathbf{n}$ can be approximated as

$$\nabla u \cdot \mathbf{n} = b_1 n_1 + b_2 n_2$$

which is linear combination of b_1 and b_2. The normal direction (n_1, n_2) is approximated by $(\tilde{n_1}, \tilde{n_2})$:

$$\begin{cases} \int_{\partial\Omega} \tilde{n}_1(v)wds = \int_{\partial\Omega} n_1(v)wds, \\ \int_{\partial\Omega} \tilde{n}_2(v)wds = \int_{\partial\Omega} n_2(v)wds, \end{cases} \tag{7.29}$$

for any $w \in H^1(\Omega)$. Then we use the approximation

$$\nabla u \cdot \mathbf{n} \approx b_1 \tilde{n}_1 + b_2 \tilde{n}_2. \tag{7.30}$$

Following (7.28)-(7.30), we can derive a matrix $\mathbf{M}$ such that (7.30) can be computed as

$$\nabla u \cdot \mathbf{n} \approx b_1 \tilde{n}_1 + b_2 \tilde{n}_2 = \mathbf{M}u. \tag{7.31}$$

Then (7.18) and (7.25) can be solved easily.

7.5 Numerical experiments

In this section, we test the performance of the two algorithms. Most examples are solved by Algorithm NU1. For some selected tests we compare the two algorithms.

7.5.1 Example 1

In this example, the domain is $\Omega = (0,1)^2$. We set the example such that the exact solution is

$$u^* = 8\left[\left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2\right] - 3. \tag{7.32}$$

Then $g(0,0) = 0$ and $f = 256$. The boundary condition is computed from the exact solution. During our computation, we force $u(0,0) = u^*(0,0) = 0$. First we set $\varepsilon = \varepsilon_1 = h^{1.5}$. The stopping criterion is chosen as $\|u^{n+1} - u^n\|_2 < 10^{-7}$. But for the Algorithm NU on the regular mesh on unit square with $h = 0.0354$, we need $\|u^{n+1} - u^n\|_2 < 10^{-8}$ to achieve the best accuracy. The computed solution and the convergence behavior on different mesh is shown in Figure 7.1. The L_2 and L_∞ error and their corresponding convergence rate is shown in Table 7.1. Generally the convergence rate is larger than 1.5. Only on the unstructured mesh on a unit square the negative convergence rate appears. We think that maybe due to the mesh is too coarse. With this setting, Algorithm NU2 does not converge on all kinds of meshes. To compare the performance of the two algorithms, we set $\varepsilon = \varepsilon_1 = h$. The comparison is shown in Table 3.2. Algorithm NU1 has higher accuracy but needs more iterations to achieve the best accuracy than Algorithm NU2. But Algorithm NU1 is more robust and we can use smaller ε and ε_1 to get better accuracy and convergence rate.

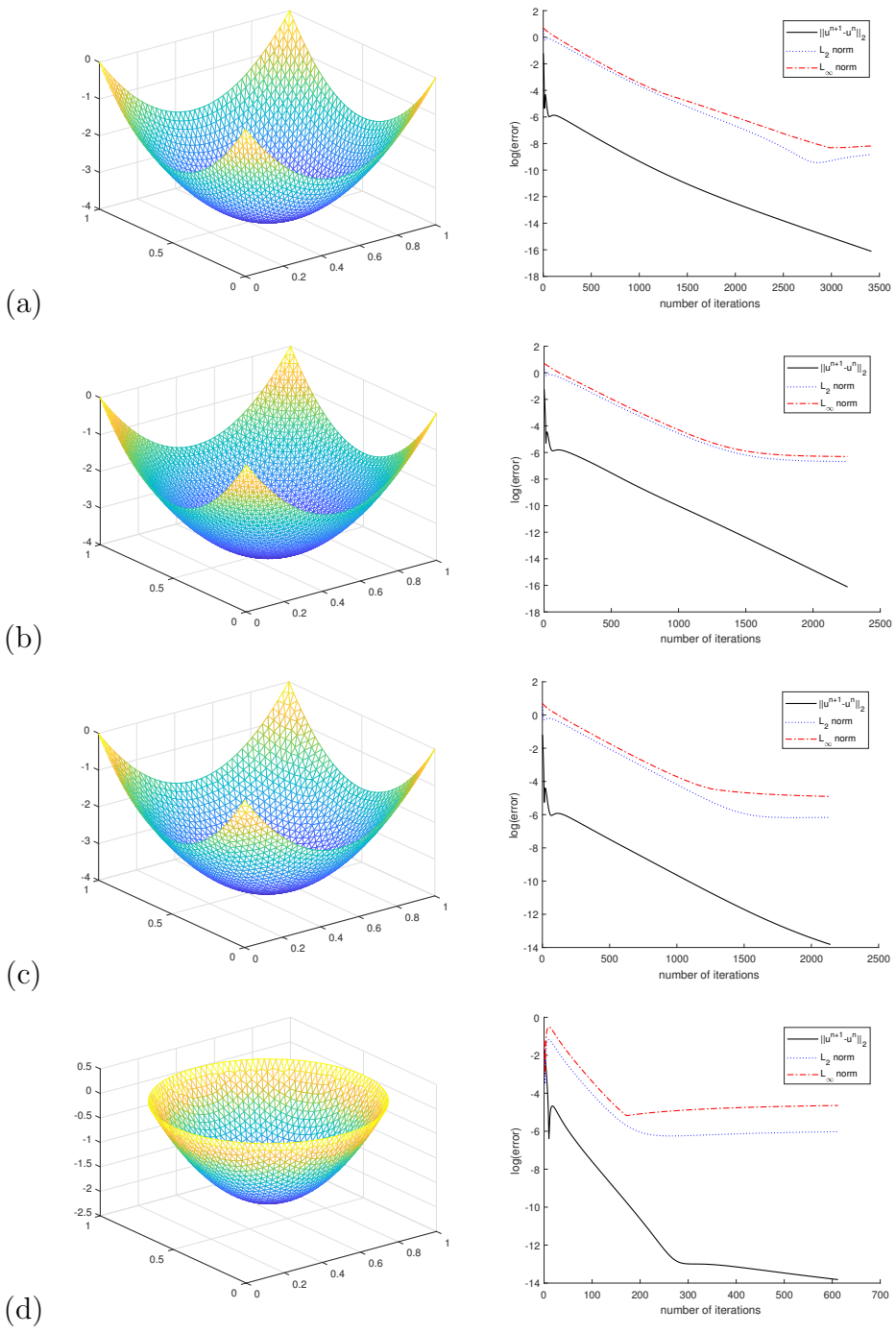


Figure 7.1: (Example 1, $\varepsilon = \varepsilon = h^{1.5}$) Graphs of the computed solutions of Algorithm NU1 and related convergence behaviors on: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square non-uniform mesh, and (d) the half-unit disk triangulation.

(a)

$h(h^2)$	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1414	286	9.76×10^{-7}	4.61×10^{-3}		1.39×10^{-2}	
0.0707	864	9.98×10^{-7}	7.45×10^{-4}	2.63	1.81×10^{-3}	2.94
0.0354	3417	9.99×10^{-8}	1.45×10^{-4}	2.36	2.80×10^{-4}	2.69

(b)

$h(h)$	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	194	9.68×10^{-7}	2.26×10^{-2}		3.66×10^{-2}	
0.05	577	9.87×10^{-7}	5.38×10^{-3}	2.07	8.06×10^{-3}	2.18
0.025	2257	9.95×10^{-8}	1.26×10^{-3}	2.09	1.83×10^{-3}	2.14

(c)

$h(h)$	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	234	9.86×10^{-7}	1.48×10^{-2}		6.12×10^{-2}	
0.05	793	9.97×10^{-7}	1.59×10^{-2}	-0.10	4.00×10^{-2}	0.61
0.025	2412	9.99×10^{-7}	2.10×10^{-3}	2.92	7.42×10^{-3}	2.43

(d)

$h(h)$	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	154	9.74×10^{-7}	4.89×10^{-2}		8.24×10^{-2}	
0.05	452	9.92×10^{-7}	6.97×10^{-3}	2.81	2.79×10^{-2}	1.56
0.025	612	9.99×10^{-7}	2.42×10^{-3}	1.53	9.63×10^{-3}	1.53

Table 7.1: (Example 1, $\varepsilon = \varepsilon = h^{1.5}$) Number of iterations necessary for the convergence of the scheme, approximation errors and space approximation convergence rates for: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square unstructured mesh, and (d) the half-unit disk triangulation.

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	3486	9.99×10^{-8}	2.70×10^{-3}	3.21×10^{-3}
(ii)	0.0354	460	9.98×10^{-7}	5.40×10^{-3}	1.00×10^{-2}
(iii)	0.025	1997	9.99×10^{-7}	3.68×10^{-3}	5.28×10^{-3}
(iv)	0.025	1708	9.99×10^{-7}	1.01×10^{-1}	1.87×10^{-1}

Table 7.2: (Example 1, $\varepsilon = \varepsilon_1 = h$) Comparison between Algorithm NU1 and Algorithm NU2 on the unit square regular mesh with $h = 0.0354$ and the unit square symmetric mesh with $h = 0.025$: (i) Regular mesh with Algorithm NU1. (ii) Regular mesh with Algorithm NU2. (iii) Symmetric mesh with Algorithm NU1. (iv) Symmetric mesh with Algorithm NU2.

7.5.2 Example 2

In the second example, we test problem with exact solution being the exponential function

$$u^* = 4e^{r^2} - 4e^{\frac{1}{2}} \tag{7.33}$$

where $r = [(x_1 - 1/2)^2 + (x_2 - 1/2)^2]^{1/2}$. Then we have $f = 64e^{2r^2}(1 + 2r)$. The boundary condition is computed from the exact solution. During the iterations, we force $u(0, 0) = u^*(0, 0) = 0$. First we test the performance of Algorithm NU1 with $\varepsilon = \varepsilon_1 = h^{1.5}$ and stopping criterion $\|u^{n+1} - u^n\|_2 < 10^{-6}$. The computed result and convergence behavior is shown in Figure 7.2. The L_2 and L_∞ errors and their corresponding convergence rate are shown in Table 7.3. Generally the convergence rate is larger than 1. For this example, with $\varepsilon = \varepsilon_1 = h^{1.5}$ Algorithm NU2 works but a smaller time step is required and it converges very slow. To compare the performance, we use $\varepsilon = \varepsilon_1 = h$. The comparison on regular and symmetric mesh on a unit square is shown in Table 7.4. There is no big difference on accuracy between these two algorithm. Algorithm NU2 is more efficient on solving this example on regular mesh.

(a)

$h(h^2)$	Iterations	$\|u^{n+1} - u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1414	246	9.71×10^{-7}	9.57×10^{-2}		2.13×10^{-1}	
0.0707	732	9.89×10^{-7}	3.98×10^{-2}	1.27	9.25×10^{-2}	1.20
0.0354	2106	9.99×10^{-8}	2.77×10^{-2}	0.52	4.49×10^{-2}	1.04

(b)

$h(h)$	Iterations	$\|u^{n+1} - u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	159	9.79×10^{-7}	8.59×10^{-2}		1.46×10^{-1}	
0.05	466	9.89×10^{-7}	3.80×10^{-2}	1.18	6.44×10^{-2}	1.18
0.025	1421	9.96×10^{-7}	1.11×10^{-2}	1.78	2.04×10^{-2}	1.66

(c)

$h(h)$	Iterations	$\|u^{n+1} - u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	226	9.70×10^{-7}	7.73×10^{-2}		1.30×10^{-1}	
0.05	753	9.94×10^{-7}	1.61×10^{-2}	2.26	3.64×10^{-2}	1.84
0.025	1798	9.99×10^{-7}	6.04×10^{-3}	1.41	1.60×10^{-2}	1.19

(d)

$h(h)$	Iterations	$\|u^{n+1} - u^n\|_2$	L_2 norm	rate	L_∞ norm	rate
0.1	150	9.92×10^{-7}	3.32×10^{-2}		6.78×10^{-2}	
0.05	426	9.91×10^{-7}	1.13×10^{-2}	1.55	2.60×10^{-2}	1.38
0.025	549	9.97×10^{-7}	3.91×10^{-3}	1.53	1.06×10^{-2}	1.29

Table 7.3: (Example 2, $\varepsilon = \varepsilon = h^{1.5}$) Number of iterations necessary for the convergence of the scheme, approximation errors and space approximation convergence rates for: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square unstructured mesh, and (d) the half-unit disk triangulation.

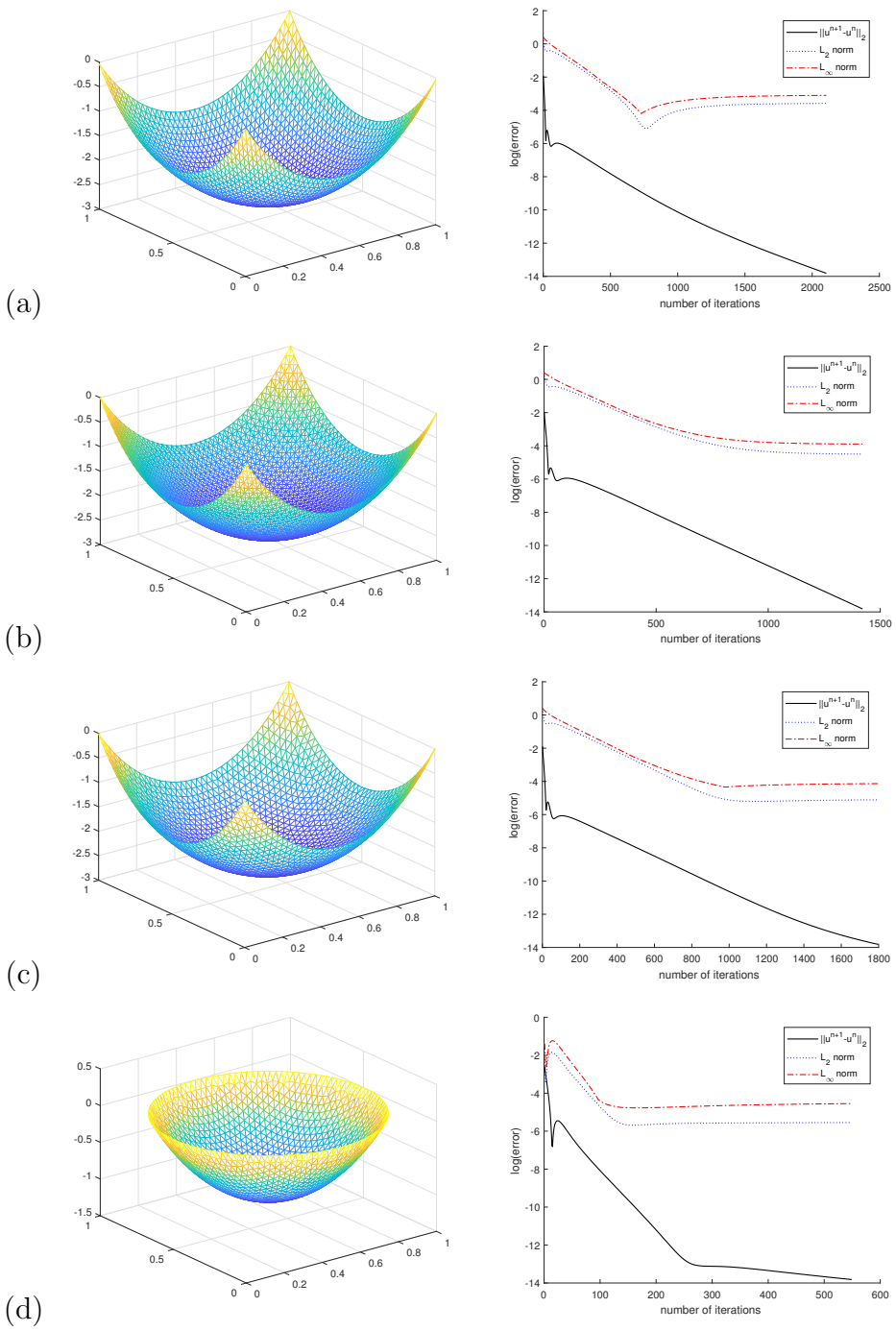


Figure 7.2: (Example 2, $\varepsilon = \varepsilon = h^{1.5}$) Graphs of the computed solutions Algorithm NU1 and related convergence behaviors for: (a) the unit square regular mesh, (b) the unit square symmetric mesh, (c) the unit square non-uniform mesh, and (d) the half-unit disk triangulation.

	h	Iterations	$\|u^{n+1}-u^n\|_2$	L_2 norm	L_∞ norm
(i)	0.0354	2146	9.98×10^{-7}	3.82×10^{-2}	7.85×10^{-2}
(ii)	0.0354	450	9.98×10^{-7}	3.17×10^{-2}	6.59×10^{-2}
(iii)	0.025	1598	9.99×10^{-7}	6.53×10^{-2}	1.13×10^{-1}
(iv)	0.025	1371	9.96×10^{-7}	9.69×10^{-2}	1.44×10^{-1}

Table 7.4: (Example 2, $\varepsilon = \varepsilon_1 = h$) Comparison between Algorithm NU1 and Algorithm NU 2 on the unit square regular mesh with $h = 0.0354$ and the unit square symmetric mesh with $h = 0.025$: (i) Regular mesh with Algorithm NU1. (ii) Regular mesh with Algorithm NU2. (iii) Symmetric mesh with Algorithm NU1. (iv) Symmetric mesh with Algorithm NU2.

7.5.3 Example 3

In this example, we show an application of our algorithms. We use our algorithm to generate adaptive mesh. The adaptive mesh generated using the Monge-Ampère equation was introduced in [112]. Here we test one of their examples. In this problem, two mesh grid density distributions ρ_1, ρ_2 are given. We are given the grids location following ρ_1 and want to transport it to follow ρ_2. We choose $\Omega = (0,1)^2, \rho_1(\mathbf{x}) = 1\ \forall \mathbf{x} \in \Omega$. Define

$$k = 1 + 2\exp\left(-C\left|(x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 - 0.09\right|\right). \tag{7.34}$$

Then we choose

$$\rho_2 = \frac{k}{\int_\Omega k d\mathbf{x}}. \tag{7.35}$$

This choice of ρ_2 has largest density on a circle centered at $(1/2, 1/2)$. Here C is a parameter controlling the distribution, the larger C is, the larger density on the circle.

Since the regular mesh and the Cartesian mesh uses the same grid points, we only consider regular mesh. We solve (7.3) with $\nabla u = (x_1, x_2)$ in (7.35). So we have

$$f = \frac{1}{\rho_2(\nabla u)}.$$

The strategy used here is similar to the our method solving the Minkowski problem. We use u^n from the previous step to compute f and then apply Algorithm NU1 or NU2 to update u^{n+1}. Our initial condition is computed from the standard Cartesian mesh $\{\mathbf{z}_i\}_{i=1}^N$:

$$u^0(\mathbf{z}_i) = \frac{1}{2}\|\mathbf{z}_i\|^2$$

where $\mathbf{z}_i = (z_{i1}, z_{i2})$ is the location of the i^{th} grid point. Then we have $\mathbf{z}_i = \nabla u(\mathbf{z}_i)$. The grid points and mesh computed from the initial condition is shown in Figure 7.3. When computing the gradient on the boundary, we only have information on one side, so the error is larger than interior value. The boundary condition used here is the Neumann boundary condition:

$$\nabla u \cdot \mathbf{n} = \mathbf{z} \cdot \mathbf{n}, \ \forall \mathbf{z} \text{ on } \partial\Omega,$$

where $\mathbf{n}$ is the unit outward direction. With this boundary condition, grid points on the boundary is only allowed to move along the boundary. Before and after the transportation, we keep the connectivity between the grid points.

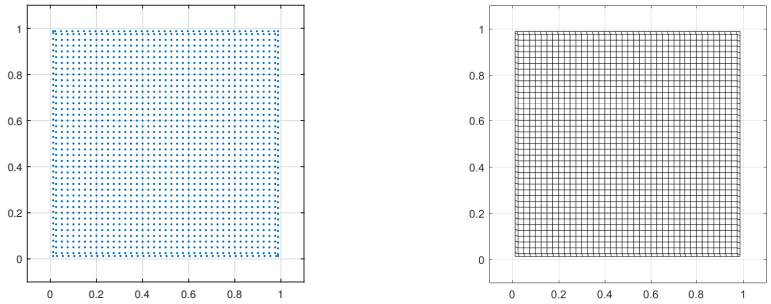
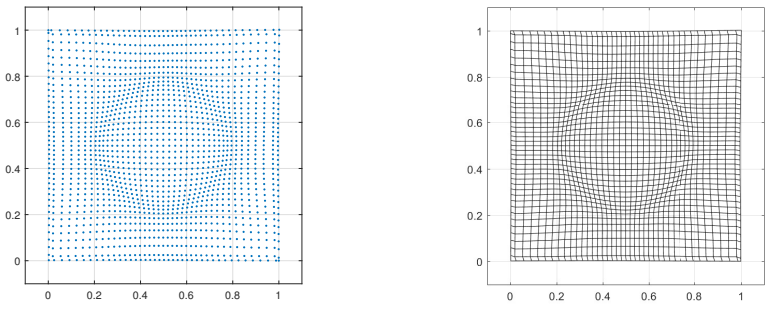


Figure 7.3: (Example 3) Grid points and mesh computed from u^0.

First we use a small $C = 10$. With $\varepsilon = \varepsilon_1 = h^{1.5}$, both algorithms works. The resulting grid points and meshes are shown in Figure 7.4. Both algorithm generate the adaptive mesh well. Although Algorithm NU1 cannot make the solution strictly satisfy the boundary condition, the error is very small. Grid points on the boundary almost stay on the boundary after the transportation.

(a)



(b)

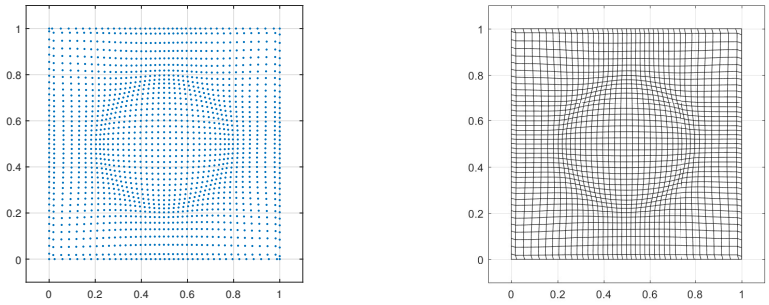


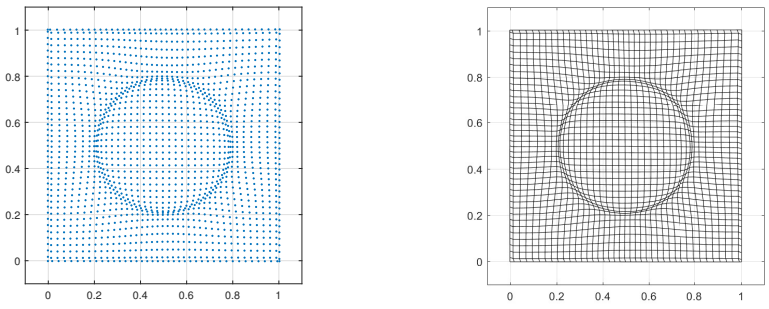
Figure 7.4: (Example 3, $C = 10$) Adaptive grid points and mesh generated by (a) Algorithm NU1, (b) Algorithm NU2.

Then we use a larger $C = 50$ to see the performance of the two algorithms. With $\varepsilon = \varepsilon_1 = h^{1.5}$, both algorithms works. The resulting grid points and meshes are shown in Figure 7.5. With this C, the grid points are more dense near the circle in the results by both algorithms. By Algorithm NU1, grid points on the boundary also almost stay on the boundary. For both choices of C, there is no line cross in the mesh plot by both algorithms.

7.6 Conclusion

In this chapter, we applied our previous introduced Monge-Ampère equation solver on the Neumann Monge-Ampère equation problems. We design two algorithms: one is robust and works with smaller smoothing factor, $\varepsilon, \varepsilon_1$, on all kinds of meshes. But the solution may not strictly satisfy the boundary condition. The accuracy order is larger than one in general. The other one is less robust but its

(a)



(b)

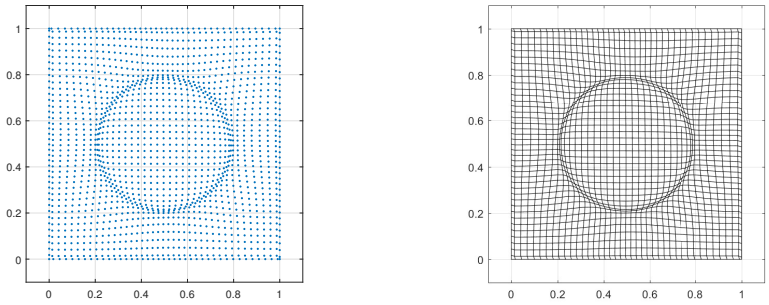


Figure 7.5: (Example 3, $C = 10$) Adaptive grid points and mesh generated by (a) Algorithm NU1, (b) Algorithm NU2.

solution strictly satisfies the boundary condition. We have used both algorithms to generate adaptive mesh. Both of them perform good.

Chapter 8

Three Dimensional Monge-Ampère Equation

8.1 Introduction

In previous chapters, all methods are proposed to solve two dimensional Monge-Amère related problems. In this chapter, we consider the Monge-Ampère equation in three dimension. Consider the Monge-Ampère equation with Dirichlet boundary condtion in dimension three:

$$\begin{cases} \det \mathbf{D}^2 u = f \text{ in } \Omega, \\ u|_{\partial\Omega} = g \end{cases} \tag{8.1}$$

where Ω is a bounded domain in $\mathbb{R}^3$, D^2u is the Hessian matrix of u:

$$\mathbf{D}^2 u = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2} & \frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_1 \partial x_3} \\ \frac{\partial^2 u}{\partial x_1 \partial x_2} & \frac{\partial^2 u}{\partial x_2^2} & \frac{\partial^2 u}{\partial x_2 \partial x_3} \\ \frac{\partial^2 u}{\partial x_3 \partial x_1} & \frac{\partial^2 u}{\partial x_3 \partial x_3} & \frac{\partial^2 u}{\partial x_3^2} \end{pmatrix},$$

f and g are given functions with $f > 0$.

Due to the importance of the Monge-Ampère equation, lots of numerical algorithms has been developed recently. But most of them focus on the two dimensional problem. The three dimensional is more complicated since it includes cubic terms whereas there are only quadratic terms in the two dimensional problem. Here is a brief literature review on numerical methods solving three dimensional Monge-Ampère equations. In [35], a vanishing moment method is introduced to solve second order fully nonlinear PDEs. Similar to the viscosity solution idea, a biharmonic term is added to the PDE and then solved by preconditioned Newton method. In [12], solved by vanishing moment method with Newton method, a penalization method is designed by extending the ideas in [11]. The error bound is proved with polynomial finite element space with order equal or larger than 3. Another vanishing moment based method using Morley finite element [83] is proposed in [88]. The estimation of error bound is given with condition on mesh size. In [38, 39], two finite difference method based on wide stencil [92] is proposed for solving Monge-Ampère equation in dimension two and three. But due to the employment of the wide-stencil, they are not simple to solve problem on irregular domain. In [86], a hybrid method using the wide-stencil and power diagrams [2] is designed. In [108], a least square method for σ_2 problem(which can be written as three dimensional Monge-Ampère equation) is proposed by extending idea in [29]. In [9], a projection method is introduced using C^1 finite element. But it is difficult to construct C^1 finite element spaces in 3 dimension.

The idea here is simple. First we rewrite (8.1) in another form using the cofactor matrix of $\mathbf{D}^2 u$. Then the new equation is solved in an operator-splitting way discretized by finite element method. To enforce the convexity, an eigenvalue projection method using hard thresholding is used. The implementation is pretty easy. Our method can solve Monge-Ampère equation with smooth classical solution or with singularities in dimension three.

This work is an extension of the idea in Chapter 3. Among those methods mentioned above, most numerical experiments are conducted on a cube. Only in [12] an experiment on an ellipsoid is mentioned. Using our method, problem on regular domain such as a cube or irregular domain such as a sphere can be solved easily.

8.2 A divergence formulation of problem and an associated initial value problem

In this section, we derive our initial value problem associated to (8.1). Let $A = \{a_{ij}\}, i, j = 1, 2, 3$. The cofactor matrix associated to A is defined as

$$\operatorname{cof}(A) = \frac{1}{\det(A)} \begin{pmatrix} a_{22}a_{33} - a_{23}a_{32} & a_{23}a_{31} - a_{21}a_{33} & a_{21}a_{32} - a_{22}a_{31} \\ a_{13}a_{32} - a_{12}a_{33} & a_{11}a_{33} - a_{13}a_{31} & a_{12}a_{31} - a_{11}a_{32} \\ a_{12}a_{23} - a_{13}a_{22} & a_{13}a_{21} - a_{11}a_{23} & a_{11}a_{22} - a_{12}a_{21} \end{pmatrix}. \quad (8.2)$$

Using (8.2), the left hand side of (8.1) can be written in divergence form:

$$\det \mathbf{D}^2 u = \frac{1}{3} \nabla \cdot \left(\operatorname{cof} \left(\mathbf{D}^2 u \right) \nabla u \right).$$

The Monge-Ampère equation in dimension 3 can then be written as

$$\begin{cases} -\nabla \cdot \left(\operatorname{cof} \left(\mathbf{D}^2 u \right) \nabla u \right) + 3f = 0 \text{ in } \Omega, \\ u|_{\partial\Omega} = g. \end{cases} \quad (8.3)$$

By some steps of integration by parts, it can be easily shown that the solution to (8.3) is a stationary point of the energy

$$I(u) = \int_\Omega \operatorname{cof} \left(\mathbf{D}^2 u \right) \nabla u \cdot \nabla u dx + 12 \int_\Omega f u dx. \quad (8.4)$$

The simplest Hilbert space to solve for u is H^2. Define

$$V_g = \{v | v \in H^2, v|_{\partial\Omega=g}\}.$$

Let ϕ be a stationary point of $I(u)$. We have $I'(u) = 0$ which leads to the weak formulation of (8.1):

$$\begin{cases} \phi \in V_g, \\ \int_\Omega \mathrm{cof}\,(\mathbf{D}^2\phi)\,\nabla\phi \cdot \nabla v dx + 3\int_\Omega f\phi dx = 0, \forall v \in V_g. \end{cases} \tag{8.5}$$

We have the following theorem on the positive definiteness preserve of the cofactor operator:

Theorem 8.2.1. *Let* $\mathbf{p}$ *be a* 3×3 *matrix and* $\mathrm{cof}(\mathbf{p})$ *be the cofactor matrix of* $\mathbf{p}$. $\mathrm{cof}(\mathbf{p})$ *is positive definite if and only if* $\mathbf{p}$ *is either positive definite or negative definite.*

Proof. Denote the eigenvalues of $\mathbf{p}$ by l_1, l_2, l_3 with eigenvectors v_1, v_2, v_3 respectively. Denote eigenvalues of $\mathrm{cof}(\mathbf{p})$ by k_1, k_2, k_3. Since $\mathrm{cof}(\mathbf{p})$ is the cofactor matrix of $\mathbf{p}$, we have

$$\mathbf{p}^{-1} = \frac{1}{\det(\mathbf{p})}\mathrm{cof}(\mathbf{p}). \tag{8.6}$$

Since

$$\mathbf{p}v_i = l_i v_i,$$

we have

$$\mathbf{p}^{-1}v_i = \frac{1}{l_i}v_i$$

for $i = 1, 2, 3$.

Multiply (8.6) by v_i on the right, we have

$$\frac{1}{l_i}v_i = \mathbf{p}^{-1}v_i = \frac{1}{\det(\mathbf{p})}\mathrm{cof}(\mathbf{p})v_i$$

for $i = 1, 2, 3$.

The above equation implies

$$k_i = \frac{\det(\mathbf{p})}{l_i}, i = 1, 2, 3. \tag{8.7}$$

First we prove the sufficient part.

If $\mathbf{p}$ is positive definite, $\det(\mathbf{p}) > 0$ and $l_i > 0$ for $i = 1, 2, 3$. So $k_i = \frac{\det(\mathbf{p})}{l_i} > 0$ for $i = 1, 2, 3$ and $\mathrm{cof}(\mathbf{p})$ is positive definite. If $\mathbf{p}$ is negative definite, $\det(\mathbf{p}) < 0$ and $l_i < 0$ for $i = 1, 2, 3$. So $k_i = \frac{\det(\mathbf{p})}{l_i} > 0$ for $i = 1, 2, 3$ and $\mathrm{cof}(\mathbf{p})$ is positive definite.

For the necessary part, assume $\mathrm{cof}(\mathbf{p})$ is positive definite. Then $k_i > 0$ for $i = 1, 2, 3$. Since (8.7),

$$\frac{\det(\mathbf{p})}{l_i} > 0 \tag{8.8}$$

and l_i's have the same sign for $i = 1, 2, 3$. If they all are positive, $\det(\mathbf{p}) > 0$ and (8.8) is satisfied. $\mathbf{p}$ is positive definite. If they all are negative, $\det(\mathbf{p}) < 0$ and (8.8) is satisfied. $\mathbf{p}$ is negative definite. □

If u is convex, then $\mathbf{D}^2u$ is positive definite and $\mathrm{cof}(\mathbf{D}^2u)$ is positive definite.

Remark 8.2.1. *By solving (8.1), we want to find convex solution u. Since we have $f > 0$ on Ω, u must be strictly convex and $\mathbf{D}^2u$ is positive definite.*

Remark 8.2.2. *Let $p_0 = (a, b, c) \in \mathbb{R}^3$. Define $r^2 = (x_1-a)^2+(x_2-b)^2+(x_3-c)^2$ and $u(r)$ is a function of r. Then we have*

$$\det(\mathbf{D}^2u) = u'' \left(\frac{u'}{r}\right)^2.$$

When solving the Monge-Ampère equation, we want to find convex solution. So $\mathbf{D}^2u$ is positive definite and (8.3) is an elliptic PDE of u. Before go to the initial value problem, we work on the nonlinearity of (8.3) first.

To handle the nonlinearity in (8.3), we use a new variable $\mathbf{p}$ to replace $\mathbf{D}^2u$. Then solving (8.3) is equivalent to solve the following system:

$$\begin{cases} -\nabla \cdot (\mathbf{p}\nabla u) + 3f = 0, \\ \mathbf{p} - \mathbf{D}^2 u = 0, \\ u|_{\partial\Omega} = g. \end{cases} \tag{8.9}$$

Introducing artificial time t and viscosity term $-\Delta u$, we get the initial value problem:

$$\begin{cases} \frac{\partial u}{\partial t} - \epsilon\Delta u - \nabla \cdot (\mathbf{p}\nabla u) + 3f = 0, \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma\left(\mathbf{p} - \mathbf{D}^2 u\right) = 0, \\ u|_{\partial\Omega} = g, \\ u(0) = u_0, \mathbf{p}(0) = \mathbf{p}_0, \end{cases} \tag{8.10}$$

where ϵ is a small positive number and γ is a positive parameter. The solution to the original Monge-Ampère equation is the steady state of u in (8.10).

Note that $-\Delta u = -\nabla \cdot (\nabla u)$, (8.10) can be rewritten as

$$\begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot (\epsilon\mathbf{I} + \mathbf{p}\nabla u) + 3f = 0, \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma\left(\mathbf{p} - \mathbf{D}^2 u\right) = 0, \\ u|_{\partial\Omega} = g, \\ u(0) = u_0, \mathbf{p}(0) = \mathbf{p}_0. \end{cases} \tag{8.11}$$

When solving (8.11), we want $\mathbf{p}$ to evolve roughly with the same rate as that of u. Denote the three eigenvalues of $\mathbf{D}^2u$ by l_1, l_2, l_3. Denote the arithmetic mean and algebraic mean of these three eigenvalues by $\bar{l}$ and $\tilde{l}$ respectively:

$$\bar{l} = \frac{l_1 + l_2 + l_3}{3}, \tilde{l} = (l_1 l_2 l_3)^{1/3}.$$

Also, denote the algebraic mean of the eigenvalues of $\mathrm{cof}\,(\mathbf{p})$ by $\tilde{k}$. Since

$$\mathbf{p}^{-1} = \frac{1}{\det(\mathbf{p})}\mathrm{cof}\,(\mathbf{p}),$$

we have

$$\begin{aligned} &\det(\mathrm{cof}\,(\mathbf{p})) = (\det(\mathbf{p}))^2 \\ \Rightarrow\quad &\tilde{k}^3 = \tilde{l}^{3\cdot 2} \Rightarrow \tilde{k} = \tilde{l}^2. \end{aligned}$$

Let λ_0 be the smallest eigenvalue of $-\Delta$ and α be the smallest value of f. From (8.1), we have $l^3 = l_1 l_2 l_3 = f$. Then the smallest rate at which u evolves is

$$\lambda_0 k_{\min} = \lambda_0 l_{\min}^2 = \lambda_0 \alpha^{\frac{2}{3}},$$

where $k_{\min}$ and $l_{\min}$ are the smallest value of k and l in Ω respectively. We choose $\gamma = \kappa\lambda_0\alpha^{\frac{2}{3}}$ where κ is some parameter of order $O(1)$.
To enforce the convexity of u, we use two techniques:

- Use good initial condition u_0 and $\mathbf{p}_0$.
- Enforce the positivity of $\mathbf{p}^n, \mathbf{p}^n$ being intermediate state.

We now discus how to choose our initial condition. Enforcing the positivity of $\mathbf{p}^n$ will be discussed in Section 8.4.7.
For the initial condition u_0, it is reasonable to approximate $\bar{l}$ by $\tilde{l}$. Consider the Laplacian operator Δ, we have

$$\Delta u = l_1 + l_2 + l_3 = 3\bar{l} \approx 3\tilde{l} = 3f^{1/3}.$$

So we choose u_0 as solution to the following Laplace equation:

$$\Delta u = 3\beta f^{1/3}, \tag{8.12}$$

where β is some factor. And we choose $\mathbf{p}_0$ as the Hessian matrix of u_0:

$$\mathbf{p}_0 = \mathbf{D}^2 u_0. \tag{8.13}$$

8.3 Time discretization by operator-splitting method

Operator splitting method is very useful in solving complicate PDE. A great variety of schemes have be designed. For a deep discussion on Operator splitting method, we refer interested readers to [47]. In this work to solve (8.11), we use the ***Lie type*** operator splitting:

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{8.14}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \left\{u^{n+1/2}, \mathbf{p}^{n+1/2}\right\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

Fractional Step 1:

Solve

$$\begin{cases} \begin{cases} \frac{\partial u}{\partial t} - \nabla \cdot \left[\left(\varepsilon \mathbf{I} + \operatorname{cof}(\mathbf{p}^n)\right) \nabla u\right] + 3f = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ u = g \text{ on } \partial\Omega \times (t^n, t^{n+1}), \end{cases} \\ \frac{\partial \mathbf{p}}{\partial t} = \mathbf{0} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^n, \mathbf{p}(t^n) = \mathbf{p}^n \end{cases} \tag{8.15}$$

and set

$$u^{n+1/2} = u(t^{n+1}), \mathbf{p}^{n+1/2} = \mathbf{p}(t^{n+1}). \tag{8.16}$$

Fractional Step 2:

Solve

$$\begin{cases} \frac{\partial u}{\partial t} = 0 \text{ in } \Omega \times (t^n, t^{n+1}), \\ \frac{\partial \mathbf{p}}{\partial t} + \gamma \mathbf{p} = \gamma \mathbf{D}^2 u^{n+1/2} \text{ in } \Omega \times (t^n, t^{n+1}), \\ u(t^n) = u^{n+1/2}, \mathbf{p}(t^n) = \mathbf{p}^{n+1/2}, \end{cases} \tag{8.17}$$

and set

$$u^{n+1} = u(t^n), \mathbf{p}^{n+1} = \mathbf{p}(t^{n+1}). \tag{8.18}$$

System (8.15)-(8.17) is first order accurate. The crucial problem is to solve (8.15) and (8.17). There is no difficulty in solving (8.17) since the analytical solution can be found. (8.15) is linear in u and can be solved implicitly. Here we advocate Backward Euler method. The resulting scheme is as follows:

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{8.19}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1}-u^n}{\Delta t} - \nabla \cdot [(\varepsilon \mathbf{I} + \text{cof}(\mathbf{p}^n)) \nabla u^{n+1}] + 3f = 0 \text{ in } \Omega, \\ u^{n+1} = g \text{ on } \partial\Omega, \end{cases} \tag{8.20}$$

$$\mathbf{p}^{n+1} = e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2}. \tag{8.21}$$

To enforce the positivity of $\mathbf{p}^n$, we modify (8.21) by projecting it to a 3×3 semi-positive definite matrix space. Denote such projection operator by $P_+(\cdot)$, our scheme becomes

$$u^0 = u_0, \mathbf{p}^0 = \mathbf{p}_0. \tag{8.22}$$

For $n \geq 0$, $\{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as follows

$$\begin{cases} \frac{u^{n+1}-u^n}{\Delta t} - \nabla \cdot [(\varepsilon \mathbf{I} + \mathrm{cof}(\mathbf{p}^n)) \nabla u^{n+1}] + 3f = 0 \text{ in } \Omega, \\ u^{n+1} = g \text{ on } \partial\Omega, \end{cases} \tag{8.23}$$

$$\mathbf{p}^{n+1} = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n + \left(1 - e^{-\gamma \Delta t}\right) \mathbf{D}^2 u^{n+1/2} \right). \tag{8.24}$$

The definition of $P_+(\cdot)$ will be discussed in Section 8.4.7 which is similar to its definition in dimension two.

8.4 Finite element implementation

The divergence form in (8.20) suggests to use finite element method to implement system(8.20)-(8.21). In this work, we use tetrahedral mesh with piecewise affine basis function. In this section, we first define some finite element spaces which will be used, then we derive the finite element implementation of the approximation of the second order derivatives and scheme (8.23)-(8.24)

8.4.1 Basic finite element spaces

Assume Ω is a polyhedra domain, we divide Ω into small tetrahedrons with edges around h(in our experiments the edge length is between $0.8h \sim 1.9h$). Denote the collection of these tetrahedron by $\mathcal{T}_h$. Then the finite dimensional vector space V_h is defined as

$$V_h = \{v | v \in C^0(\bar{\Omega}), v|_T \in P_1, \forall T \in \mathcal{T}_h\},$$

where P_1 is the space of polynomials with three variables of degree≤ 1.

Denote the set of vertices of tetrahedral in $\mathcal{T}_h$ by Σ_h:

$$\Sigma_h = \{Q_i\}_{i=1}^{N_h}$$

where Q_i is vertex of tetrahedron and N_h is the number of vertex. Associate each Q_i the unique basis function w_i defined in Ω:

$$\begin{cases} w_i \in V_h, \\ w_i(Q_i) = 1, \\ w_i(Q_j) = 0, \forall j = 1, ..., N_h, j \neq i. \end{cases}$$

We have

$$v = \sum_{i=1}^{N_h} v(Q_i) w_i, \forall v \in V_h,$$

which shows dim $V_h = N_h$. Note that the support of w_i is the collection of tetrahedrons with the common vertex Q_i.

Assume $g \in C^0(\partial\Omega)$, define V_{gh} as

$$V_{gh} = \{v | v \in V_h, v(Q_i) = g(Q_i)\} \text{ if vertex } Q_i \in \partial\Omega.$$

$v \in V_{gh}$ is equivalent to $v \in V_h$ with $v = g$ on $\partial\Omega$.

We define another finite dimensional matrix-valued space $\mathbf{Q}_h$ to approximate the (symmetric) matrix value function $\mathbf{p}$

$$\mathbf{Q}_{0h} = \left\{ \mathbf{q} | \mathbf{q} = \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix} \in (V_{0h})^{3\times 3}, q_{12} = q_{21}, q_{13} = q_{31}, q_{23} = q_{32} \right\}$$

where V_{0h} is equivalent to V_{gh} with $g = 0$.

Remark 8.4.1. *The space $\mathbf{Q}_{0h}$ is not a good space to approximation to $\mathbf{p}$ since*

most problem does not have zero Dirichlet boundary condition. This issue will be discussed in Section 8.4.3. Here we use it temporally to derive our formulation.

8.4.2 Finite element approximation of second order derivatives

Consider a function $\psi \in H^2(\Omega)$, according to divergence theorem, we have

$$\forall \phi \in H_0^1(\Omega), \begin{cases} \forall i = 1,2,3,\ j = 1,2,3, \\ \int_\Omega \frac{\partial^2 \psi}{\partial x_i \partial x_j} \phi dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial \psi}{\partial x_i} \frac{\partial \phi}{\partial x_j} + \frac{\partial \psi}{\partial x_j} \frac{\partial \phi}{\partial x_i} \right] dx. \end{cases} \tag{8.25}$$

Now consider $v \in V_h$, we use D^2_{ijh} to denote the discrete approximation of $\frac{\partial^2}{\partial x_i \partial x_j}$. According to (8.25), we approximate $\frac{\partial^2 v}{\partial x_i \partial x_j}$ by

$$\begin{cases} \forall i = 1,2,3,\ j = 1,2,3, \\ D^2_{ijh}(v) \in V_{0h}, \forall w \in V_{0h} \\ \int_\Omega D^2_{ijh}(v) w dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j} \frac{\partial w}{\partial x_i} \right] dx. \end{cases} \tag{8.26}$$

For a given function f, the trapzoidal rule in on a tetrahedral E with vertex q_1, q_2, q_3, q_4 is given as

$$\int_E f dx = \frac{f(q_1) + f(q_2) + f(q_3) + f(q_4)}{4} |E|$$

where $|E|$ denotes the volume of E.

We use trapezoidal rule to approximate the integral on the left hand side in (8.26). Denote the set of vertices of tetrahedron in $\mathcal{T}_h$ which are not on $\partial\Omega$ by Σ_{0h}:

$$\Sigma_{0h} = \{Q_i\}_{i=1}^{N_{0h}}.$$

If $Q_i \in \Sigma_h$, denote the union of tetrahedron in $\mathcal{T}_h$ having Q_i as a common vertex by ω_i and denote the volume of ω_i by $|\omega_i|$. Then we have

$$\text{support of } w_i = \bar{\omega}_i.$$

Applying trapezoidal rule to the integral on the left in (8.26), we get

$$\begin{cases} \forall i = 1,2,3,\ j = 1,2,3, \\ D^2_{ijh}(v) \in V_{0h}, \forall w \in V_{0h} \\ D^2_{ijh}(v)(Q_k) = -\frac{4}{2|\omega_k|}\int_{\omega_k}\left[\frac{\partial v}{\partial x_i}\frac{\partial w_k}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w_k}{\partial x_i}\right]dx, \forall k = 1,2,...,N_{0h}. \end{cases} \tag{8.27}$$

In (8.27), since v and w_k are piecewise linear in Ω, their first order derivatives are piecewise constants. h is very small compared with the size of Ω, ω_k is the union of a small number of tetrahedrons. It is very simple to compute $D^2_{ijh}(v)(Q_k)$. In [15], the author mentioned that using approximation (8.27) in 2D to solve Monge-Ampère equation, the convergence properties of their method strongly depends on the triangulation of the domain. Similar phenomenon is observed in Chapter 3 for 2D Monge-Ampère equation. To resolve this problem, a smoothed version of the approximation is used. In this work, for 3D Monge-Ampère problem with classical solution, approximation (8.27) is fine, our scheme works well on various domain but a small time step is needed. With the smoothness, we can use a large time step and our algorithm converges much faster. For solution with singular point, the smoothness on the approximation on the second order derivatives is necessary to make our scheme convergent. Here we introduce the smoothing procedure in 3D situation. Instead of solving (8.26), we add a viscosity term on the left hand side:

$$\begin{cases} \forall i = 1,2,3,\ j = 1,2,3,\ D^2_{ijh} \in V_{0h}, \\ \int_\Omega D^2_{ijh}(v)w dx + \epsilon_1 \int_\Omega \nabla D^2_{ijh}(v)\cdot\nabla w dx = -\frac{1}{2}\int_\Omega \left[\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w}{\partial x_i}\right], \forall w \in V_{0h}, \end{cases} \tag{8.28}$$

where ϵ_1 is a positive small parameter. Note that the order of complexity of solving (8.28) is the same as that of solving (8.20), so the complexity order does not change.

8.4.3 On the boundary condition of the approximation of the second derivatives

As we mentioned before, $\mathbf{Q}_{0h}$ is not a good space to approximate $\mathbf{p}$ since it assumes zero Dirichlet boundary condition of $\mathbf{p}$ which is not appropriate for most situation. We need to propose a more natural boundary condition. Following our another work on 2D Monge-Ampère equation, we use the zero Neumann boundary condition:

$$\frac{\partial D^2_{ijh}(v)}{\partial \mathbf{n}} = 0, \forall x \in \partial\Omega \tag{8.29}$$

where $\mathbf{n}$ is the outward normal of $\partial\Omega$. Therefore, we modify (8.26) to

$$\begin{cases} \forall i = 1,2,3,\ j = 1,2,3, \\ D^2_{ijh}(v) \in V_{0h}, \forall w \in V_h \\ \int_\Omega D^2_{ijh}(v) w dx = -\frac{1}{2}\int_\Omega \left[\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j}\frac{\partial w}{\partial x_i}\right] dx, \\ \frac{\partial D^2_{ijh}(v)}{\partial \mathbf{n}} = 0, \forall x \in \partial\Omega. \end{cases} \tag{8.30}$$

Similar to what has been done in [48], we compute the interior point and boundary point of $D^2_{ijh}(v)$ using different conditions.

For interior point, we compute $D^2_{ijh}(v)$ according to (8.27) or (8.28). For boundary point, we compute it according to (8.29):

$$0 = \int_{\partial\Omega} \frac{\partial D^2_{ijh}(v)}{\partial \mathbf{n}} w dx = \int_\Omega \nabla \cdot \left(\nabla D^2_{ijh}(v) w\right) dx \tag{8.31}$$

$$= \int_\Omega \Delta D^2_{ijh}(v) w dx + \int_\Omega \nabla D^2_{ijh}(v) \cdot \nabla w dx. \tag{8.32}$$

Assume D^2_{ijh} is harmonic, $\Delta D^2_{ijh} = 0$, in each tetrahedral, the formula to compute boundary points reduces to

$$\int_\Omega \nabla D^2_{ijh}(v) \cdot \nabla w dx = 0. \tag{8.33}$$

The idea is summarized as follows:

1. At an interior point, solve

$$\int_\Omega D^2_{ijh}(v) w dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j} \frac{\partial w}{\partial x_i} \right] dx \tag{8.34}$$

   for (8.26) or solve

$$\begin{cases} \forall i = 1,2,3,\ j = 1,2,3,\ D^2_{ijh} \in V_{0h}, \\ \int_\Omega D^2_{ijh}(v) w dx + \epsilon_1 \int_\Omega \nabla D^2_{ijh}(v) \cdot \nabla w dx = -\frac{1}{2} \int_\Omega \left[\frac{\partial v}{\partial x_i} \frac{\partial w}{\partial x_j} + \frac{\partial v}{\partial x_j} \frac{\partial w}{\partial x_i} \right], \forall w \in V_{0h}, \end{cases} \tag{8.35}$$

   for (8.28).

2. At a boundary point, solve

$$\int_\Omega \nabla D^2_{ijh}(v) \cdot \nabla w dx = 0. \tag{8.36}$$

Equation (8.34)-(8.36) form a linear system of $D^2_{ijh}(v)$ and can be solved easily.

Remark 8.4.2. *Similar to the two dimensional problem, D^2_{ijh} is not harmonic here. This is a kind of variational crime.*

8.4.4 Finite element implementation of scheme (8.22)-(8.24)

We use backward Euler method to discretize (8.23) and approximate D^2u^{n+1} in (8.24) using (8.28) and (8.33). The discrete analogue of scheme (8.22)-(8.24) is given as

$$u^0 = u_{0h}(\in V_{gh}), \mathbf{p}^0 = \mathbf{p}_{0h}(\in \mathbf{Q}_h). \tag{8.37}$$

For $n \geq 0, \{u^n, \mathbf{p}^n\} \to \{u^{n+1}, \mathbf{p}^{n+1}\}$ as

$$\begin{cases} u^{n+1} \in V_{gh}, \\ \int_\Omega u^{n+1} v dx + \Delta t \int_\Omega \left(\epsilon \mathbf{I} + \text{cof}(\mathbf{p}^n)\right) \nabla u^{n+1} \cdot \nabla v dx = \\ \int_\Omega u^n v dx - 3\Delta t \int_\Omega f_h v dx, \forall v \in V_{0h}, \end{cases} \tag{8.38}$$

$$\begin{cases} \mathbf{p}^{n+1}(Q_i) = P_+ \left(e^{-\gamma \Delta t} \mathbf{p}^n(Q_i) + (1 - e^{-\gamma \Delta t}) \mathbf{D}_h^2(u^{n+1})(Q_i)\right), \\ \forall i = 1, ..., N_{0h}, \end{cases} \tag{8.39}$$

where in (8.38) f_h is a continuous approximation of f and in (8.39)

$$\mathbf{D}_h^2(u^{n+1})(Q_i) = \begin{pmatrix} D_{11h}^2(u^{n+1})(Q_i) & D_{12h}^2(u^{n+1})(Q_i) & D_{13h}^2(u^{n+1})(Q_i) \\ D_{21h}^2(u^{n+1})(Q_i) & D_{22h}^2(u^{n+1})(Q_i) & D_{23h}^2(u^{n+1})(Q_i) \\ D_{31h}^2(u^{n+1})(Q_i) & D_{32h}^2(u^{n+1})(Q_i) & D_{33h}^2(u^{n+1})(Q_i) \end{pmatrix}$$

computed using (8.34) and (8.35). The solution to (8.38) will be discussed in Section 8.4.6 and details of projection $P_+(\cdot)$ will be discussed in Section 8.4.7.

8.4.5 Initialization of scheme (8.37)-(8.39)

The initialization of (8.37)-(8.39) is the discrete analogue of (8.12) and (8.13):

$$\begin{cases} u_{0h} \in V_{gh}, \\ \int_\Omega \nabla u_{0h} \cdot \nabla v dx = -3\beta \int_\Omega f_h^{1/3} v dx, \forall v \in V_{0h}, \end{cases} \tag{8.40}$$

for u_{0h} and

$$\mathbf{p}_{0h} = \sum_{k=1}^{N_h} \begin{pmatrix} D^2_{11h}(u_{0h})(Q_k) & D^2_{12h}(u_{0h})(Q_k) & D^2_{13h}(u_{0h})(Q_k) \\ D^2_{21h}(u_{0h})(Q_k) & D^2_{22h}(u_{0h})(Q_k) & D^2_{23h}(u_{0h})(Q_k) \\ D^2_{31h}(u_{0h})(Q_k) & D^2_{32h}(u_{0h})(Q_k) & D^2_{33h}(u_{0h})(Q_k) \end{pmatrix} w_k. \tag{8.41}$$

for $\mathbf{p}_0$.

In (8.40), the integral on the right hand side can be approximated by the trapezoidal rule:

$$\int_\Omega f_h^{1/3} v dx \approx \frac{1}{4} \sum_{k=1}^{N_{0h}} |\omega_i| f_h^{1/3}(Q_k) v(Q_k).$$

In (8.41), $D^2_{ijh}(u_{0h})(Q_k)$'s are computed using (8.34)-(8.36) for $i = 1, 2, 3,\ j = 1, 2, 3$.

8.4.6 Solution to variational problem (8.38)

The general form of variational problem (8.38) can be written as

$$\begin{cases} \psi \in V_{gh} \\ \alpha \int_\Omega \psi \phi dx + \beta \int_\Omega \mathbf{M} \nabla \psi \cdot \nabla \phi dx = \gamma \int_\Omega \tilde{f} \phi dx, \forall \phi \in V_{0h}. \end{cases} \tag{8.42}$$

In (8.42), $\tilde{f}$ is a continuous given function, $\alpha(\geq 0), \beta(> 0)$ and γ are constants. $\mathbf{M}$ is a continuous piecewise affine symmetric matrix-valued function.

If $\mathbf{M}$ is pointwise positive semi definite, then (8.42) has a unique solution. Here

we assume $\mathbf{M}$ has such property.

Applying trapezoidal rule to the first and third integral,we get

$$\begin{cases} \psi \in V_{gh}, \\ \frac{\alpha}{4}\sum_{i=1}^{N_{0h}} |\omega_i|\psi(Q_i)\phi(Q_i) + \beta\int_\Omega \mathbf{M}\nabla\psi\cdot\nabla\phi dx = \frac{\gamma}{4}\sum_{i=1}^{N_{0h}} |\omega_i|\tilde{f}(Q_i)\phi(Q_i), \forall\phi\in V_{0h} \end{cases} \tag{8.43}$$

Let $\{w_i\}$ be a vector basis of V_{0h}, then (8.43) is equivalent to

$$\begin{cases} \psi \in V_{gh}, \\ \frac{\alpha}{4}|\omega_i|\psi(Q_i) + \beta\int_\Omega \mathbf{M}\nabla\psi\cdot\nabla w_i dx = \frac{\gamma}{4}|\omega_i|\tilde{f}(Q_i), \forall i = 1,2,...,N_{0h}. \end{cases} \tag{8.44}$$

Consider that ψ can be written as

$$\psi = \sum_{j=1}^{N_{0h}} \psi(Q_i)w_j + \sum_{j=N_{0h}+1}^{N_h} g(Q_j)w_j, \tag{8.45}$$

denote $\psi(Q_i)$ by ψ_i, the vector $\{\psi_i\}_{i=0}^{N_0}$ is the solution of

$$\begin{cases} \frac{\alpha}{4}|\omega_i|\psi_i + \beta\sum_{j=1}^{N_{0h}}\left(\int_{\omega_i\cap\omega_j}\mathbf{M}\nabla w_j\cdot\nabla w_i dx\right)\psi_j = \\ \frac{\gamma}{4}|\omega_i|\tilde{f}(Q_i) - \beta\sum_{j=N_{0h}+1}^{N_h}\left(\int_{\omega_i\cap\omega_j}\mathbf{M}\nabla w_j\cdot\nabla w_i dx\right)g(Q_j), \\ i = 1,\ldots,N_{0h}. \end{cases} \tag{8.46}$$

Since h is small, $\omega_i\cap\omega_j$ is a small area and the matrix associated to linear system (8.46) is sparse. Further more assume $\mathbf{M}$ is symmetric positive semi definite matrix defined on Ω, then the associated matrix is symmetric positive definite and the existence and uniqueness of solution can be proved.

Now let's discuss how to compute $\int_{\omega_i\cap\omega_j}\mathbf{M}\nabla w_j\cdot\nabla w_i dx$. Since w_i is piecewise linear, $\nabla\omega_i$ is piece wise constant and $\omega_i\cap\omega_j$ consists only a small number of tetrahedrons, the computation should be simple. Consider a tetrahedron T with

vertices $A_1A_2A_3A_4$. We want to approximate

$$\int_T \mathbf{M}\nabla\phi \cdot \nabla\theta dx, \tag{8.47}$$

with ϕ, θ being real-valued linear functions on T and $\mathbf{M}$ being symmetric real matrix-valued linear function on T. First we approximate the integral using Trapezoidal rule:

$$\int_T \mathbf{M}\nabla\phi \cdot \nabla\theta dx = \frac{|T|}{4}\left(\sum_{i=1}^{4}\mathbf{M}(A_i)\right)\nabla\phi \cdot \nabla\theta. \tag{8.48}$$

Define the basis function $\{w_i\}_{i=1}^4$ as

$$\begin{cases} w_i \in P_1, \\ w_i(A_i) = 1, w_i(A_j) = 0 \text{ if } j \neq i. \end{cases} \tag{8.49}$$

We have $\phi = \sum_{i=1}^4 \phi(A_i)w_i$ and $\theta = \sum_{i=1}^4 \theta(A_i)w_i$, and so

$$\nabla\phi = \sum_{i=1}^{4}\phi(A_i)\nabla w_i \text{ and } \nabla\theta = \sum_{i=1}^{4}\theta(A_i)\nabla w_i. \tag{8.50}$$

Let x_i, y_i, z_i be the three coordinates of A_i. Define

$$W = \det\begin{pmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{pmatrix},$$

and

$$a = -\det\begin{pmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{pmatrix},\ b = \det\begin{pmatrix} 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{pmatrix},\ c = -\det\begin{pmatrix} 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{pmatrix}.$$

We have

$$\nabla w_1 = \frac{1}{W} \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

$\nabla w_2, \nabla w_3, \nabla w_4$ can be derived using similar formula. Then integral (8.48) can be computed and linear system (8.46) can be solved.

8.4.7 Enforcing the local positive semi definiteness of p (eigenvalue projection)

In this section, we talk about the projection $P_+(\cdot)$ to enforce the semi-positivity of $\mathbf{p}$. Assume $\mathbf{p}(Q_i)$ has eigenvalue decomposition:

$$\mathbf{p}(Q_i) = \mathbf{S}_i \mathbf{R}_i \mathbf{S}_i^{-1}$$

where $\mathbf{R}_i = \begin{pmatrix} r_i^1 & 0 & 0 \\ 0 & r_i^2 & 0 \\ 0 & 0 & r_i^3 \end{pmatrix}$ with r_i^1, r_i^2, r_i^3 being $\mathbf{p}_i$'s eigenvalues. Let

$$\mathbf{R}_i^+ = \begin{pmatrix} r_i^{1+} & 0 & 0 \\ 0 & r_i^{2+} & 0 \\ 0 & 0 & r_i^{3+} \end{pmatrix}$$

with $r_i^{k+} = \max(r_i^k, 0)$ for $k = 1, 2, 3$. Then $P_+(\cdot)$ is defined as

$$P_+(\mathbf{p}^{n+1}(Q_i)) = \mathbf{S}_i \mathbf{R}_i^+ \mathbf{S}_i^{-1}, \forall i = 1, 2, ..., N.$$

(8.39) can also be written as

$$\begin{cases} \mathbf{p}^{n+1/2}(Q_i) = e^{-\gamma\Delta t}\mathbf{p}^n(Q_i) + (1-e^{-\gamma\Delta t})\mathbf{D}_h^2(u^{n+1})(Q_i) \\ \mathbf{p}^{n+1}(Q_i) = \mathbf{S}_i\mathbf{R}_i^+\mathbf{S}_i^{-1}, \\ \forall i = 1, 2, ..., N. \end{cases} \tag{8.51}$$

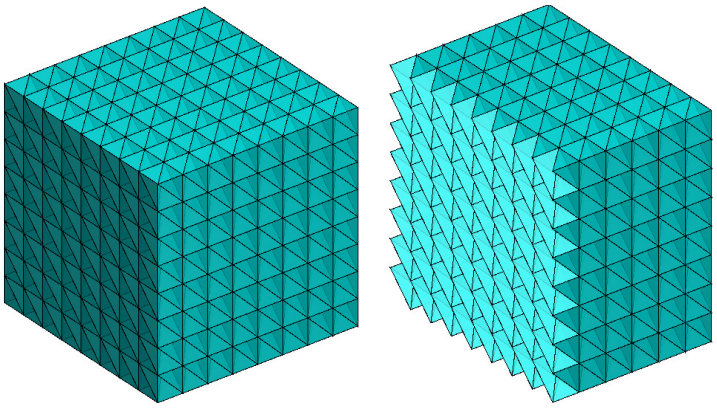
Remark 8.4.3. *Since the solution to the Monge-Ampère equation is strictly convex,* $\mathbf{p}^n$ *will is positive definite if it is close enough to the Hessian matrix of the solution. There are some* N *such that* $P_+(\cdot)$ *will have no effect on* $\mathbf{p}^n$ *for* $n > N$*.*

8.4.8 A two-stage strategy

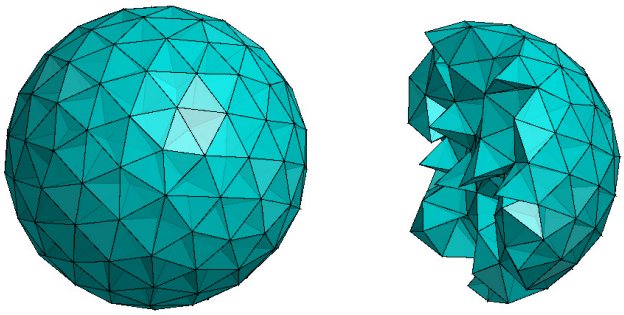
In [48] when solving 2D Monge-Ampère problem with classical solution, a two-stage strategy is used to accelerate the algorithm. In 3D problems, the two-stage strategy still works but the improvement depends on problems. We state the two-stage strategy here. Consider the following initial value problem:

$$\begin{cases} \begin{cases} -\frac{\partial(\nabla\cdot[(\varepsilon\mathbf{I}+\mathbf{M})\nabla u])}{\partial\tau} - \nabla\cdot\left[\left(\varepsilon\mathbf{I}+\mathrm{cof}(\mathbf{p})\right)\nabla u\right] + 2f = 0, (\text{for } t>0), \\ u = g \text{ on } \partial\Omega\times(0,+\infty), \end{cases} \\ \frac{\partial\mathbf{p}}{\partial\tau} + \gamma\left(\mathbf{p}-\mathbf{D}^2u\right) = \mathbf{0} \text{ in } \Omega\times(0,+\infty), \\ u(0) = u_1,\ \mathbf{p}(0) = \mathbf{p}_1, \end{cases} \tag{8.52}$$

where τ is an artificial time. $\mathbf{M}$ is a constant symmetric positive definite matrix which can be taken as the approximation of $\mathbf{D}^2u^*$ with u^* being the solution to (8.1). (8.52) is similar to (8.11), it can be solved using steps discussed before with minor modifications. Solving (8.52) needs good initial condition, otherwise it does not converge. We first solve (8.11) for some small number iterations to get good initial conditions $\{u_1, \mathbf{p}_1\}$. We use $\mathbf{M} = \mathbf{D}^2u_1$ to approximate $\mathbf{D}^2u^*$.



(a)



(b)

Figure 8.1: Finite element in a cube and a sphere. (a) figure of a cube and its cross section. (b) figure of a sphere and its cross section.

Then we solve (8.52). The two-stage strategy is summarized as follows:

Stage 1

Compute $\{u_0, \mathbf{p}_0\}$ according to (8.12) and (8.13).

Solve (8.11) for a small number of iterations.

Denote the result by $\{u', \mathbf{p}'\}$.

Stage 2

Set $\{u_1, \mathbf{p}_1\} = \{u', \mathbf{p}'\}, \mathbf{M} = \mathbf{D}^2 u'$.

Solve (8.52) up to the steady state.

8.5 Numerical examples

In this section, we test our algorithm on different domains: a cube and a sphere. The graph of our domain and their cross section are shown in Figure 8.1. In our experiments, we choose $\kappa = \beta = 0.5$ where κ is the coefficient used when computing γ, β is the coefficient used when initializing u^0. $\epsilon = \epsilon_1 = h^2$ is used. Without specification, stopping criteria $\frac{\|\mathbf{D}^2 u^n - \mathbf{p}^n\|}{\|\mathbf{p}^n\|} < 10^{-4}$ and time step $dt = 2h^2$ is used, where

$$\|\mathbf{S}\|^2 = \frac{1}{3}\sum_{k=1}^{N_h} |\omega_k| \|\mathbf{S}(Q_k)\|^2, \, \forall \mathbf{S} \in \mathbf{Q}_h \, .$$

Several examples are solved by our original algorithm on both domain with different mesh size.

The two-stage strategy is tested for the first two examples. When using the two-stage strategy, for the first stage we use time step $dt = 2h^2$ with stopping criteria $\frac{\|\mathbf{D}^2 u^n - \mathbf{p}^n\|}{\|\mathbf{p}^n\|} < 10^{-1}$. For the second stage, since $\|\mathbf{D}^2 u^n - \mathbf{p}^n\|$ and $\frac{\|\mathbf{D}^2 u^n - \mathbf{p}^n\|}{\|\mathbf{p}^n\|}$ are always very small, we use time step $d\tau = 1/2$ with stopping criteria $\|u^{n+1} - u^n\| < 10^{-6}$. For uniformity, when comparing the performance of the two algorithms, we use stopping criteria $\|u^{n+1} - u^n\| < 10^{-6}$ and time step $dt = 2h^2$ for the original algorithm.

(a)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	64	4.74×10^{-5}	1.99×10^{-4}		4.49×10^{-4}	
1/16	32	4.87×10^{-7}	4.63×10^{-5}	2.10	1.06×10^{-4}	2.08
1/32	153	1.04×10^{-8}	1.20×10^{-5}	1.95	2.71×10^{-5}	1.97

(b)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	14	2.89×10^{-7}	8.71×10^{-4}		5.00×10^{-3}	
1/16	30	1.22×10^{-8}	1.91×10^{-4}	2.19	1.35×10^{-3}	1.89
1/32	134	9.45×10^{-9}	5.08×10^{-5}	1.91	2.97×10^{-4}	2.18

Table 8.1: (Example 1) Number of iterations and errors on a cube and sphere with different mesh size, $dt = 2h^2$. (a) is on a cube with different mesh size. (b) is on a sphere with different mesh size. In (b) for $h = 1/8$, $dt = h^2$ is used.

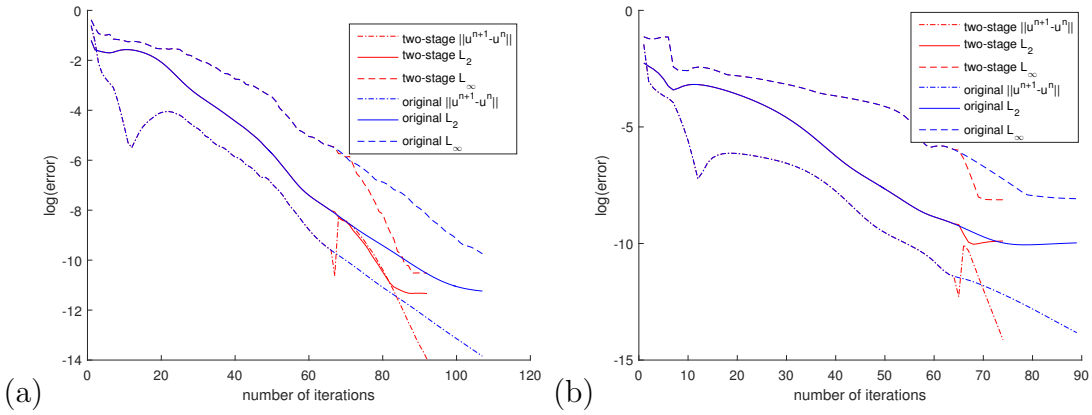


Figure 8.2: (Example 1) Comparison of the error history of the original algorithm and the two-stage strategy on different domain with $h = 1/32$. (a) is on a cube. (b) is on a sphere.

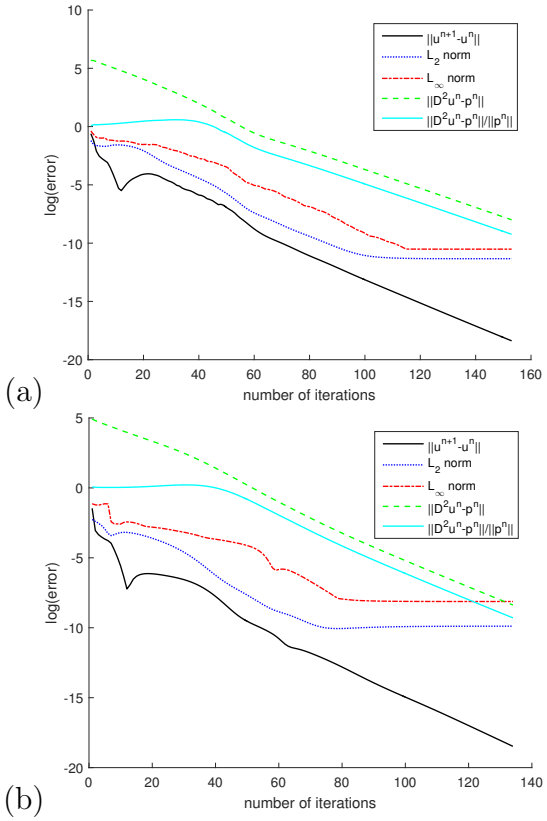


Figure 8.3: (Example 1) Error behavior with h=1/32. (a) is on a cube. (b) is on a sphere.

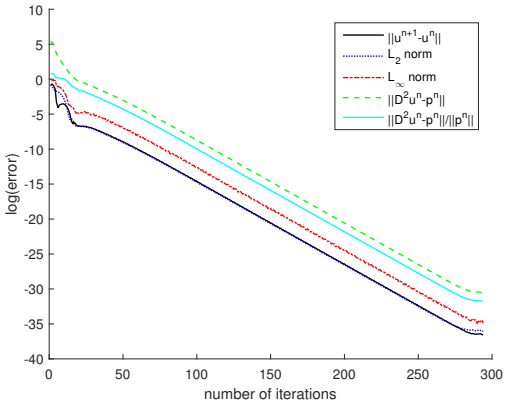


Figure 8.4: (Example 1) Error behavior with $h = 1/32, \epsilon = \epsilon_1 = 0$.

8.5.1 Example 1

For the first example, we use

$$g = (x_1 - 1/2)^2 + (x_2 - 1/2)^2 + (x_3 - 1/2)^2, \tag{8.53}$$

$$f = 8. \tag{8.54}$$

The error for different h on both domain are shown in Table 8.1. The error behavior with $h = 1/32$ for different ϵ is shown in Figure 8.3. For $h = 1/8$, $dt = 2h^2$ is too large. On a cube, if we use $dt = h^2$, only 13 iterations is needed to satisfy the stopping criteria with the same accuracy. On a sphere, our algorithm does not converge if we use $dt = 2h^2$. In Table 8.1b for $h = 1/8$, $dt = h^2$ is used. Generally, on both domain second order accuracy is observed. The comparison of error history of the original algorithm and the two-stage strategy with $h = 1/32$ on different domain is shown in Figure 8.2. On the cube, the improvement of the two-stage strategy is obvious. We can see a sudden decrease of the L_2- and L_∞-error. On the sphere, since the L_2- and L_∞-error are already very close to the best accuracy when the algorithm switches to the second stage, the improvement is not so obvious as on the cube. On both domain, in the second stage the result converges to the steady state much faster than the original algorithm.

For this example, since the zero Neumann boundary condition is exact, both L_2- and L_∞ error converges to machine precision if we choose $\epsilon = \epsilon_1 = 0$. The error history is shown in Figure 8.4.

(a)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	17	1.12×10^{-5}	2.37×10^{-3}		7.27×10^{-3}	
1/16	67	1.65×10^{-6}	7.99×10^{-4}	1.57	2.22×10^{-3}	1.71
1/32	360	3.31×10^{-9}	2.12×10^{-4}	1.91	5.71×10^{-4}	1.96

(b)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	14	4.78×10^{-6}	1.27×10^{-3}		5.43×10^{-3}	
1/16	57	2.69×10^{-7}	4.18×10^{-4}	1.60	1.70×10^{-3}	1.68
1/32	318	3.10×10^{-9}	1.10×10^{-4}	1.93	4.19×10^{-4}	2.02

Table 8.2: (Example 2) Number of iterations and errors on a cube and sphere with different mesh size. (a) is on a cube with different mesh size. (b) is on a sphere with different mesh size.

8.5.2 Example 2

For the second example, define

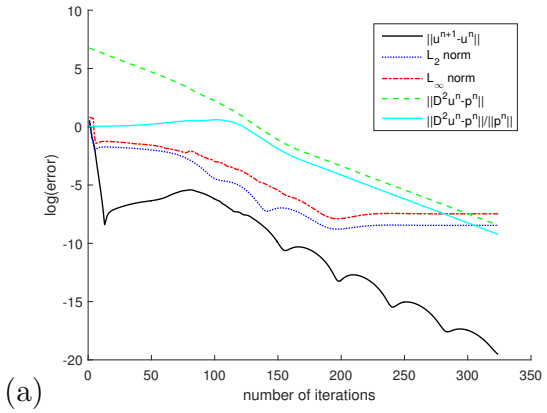
$$r^2 = (x_1 - 1/2)^2 + (x_2 - 1/2)^2 + (x_3 - 1/2)^2, \tag{8.55}$$

we use

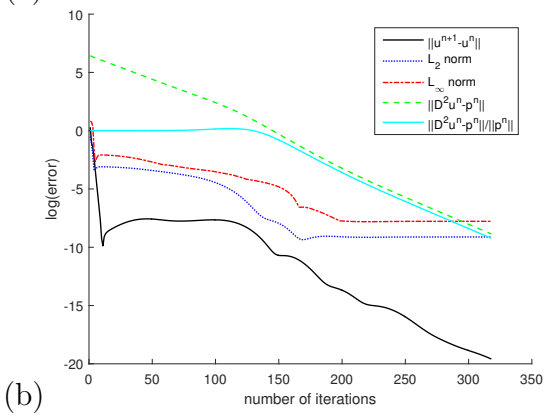
$$g = e^{r^2/2}, \tag{8.56}$$

$$f = (1 + r^2)e^{\frac{3}{2}r^2}. \tag{8.57}$$

The error for different h on both domain is shown in Table 8.2. The error history for $h = 1/32$ on both domain is shown in Figure 8.5. On average, the convergence rate is about 1.7 on a cube and 1.3 on a sphere.

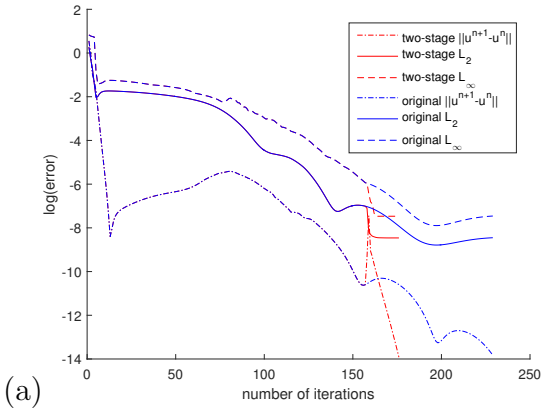


(a)

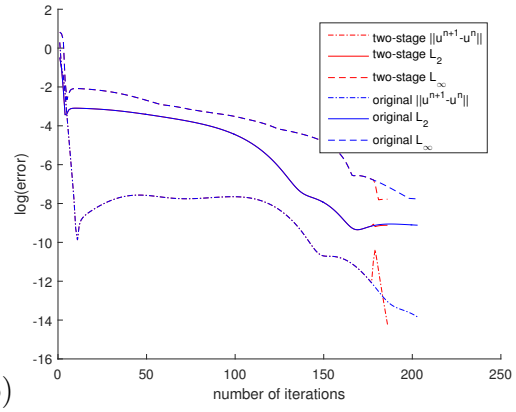


(b)

Figure 8.5: (Example 2) Error behavior with h=1/32. (a) is on a cube. (b) is on a sphere.



(a)



(b)

Figure 8.6: (Example 2) Comparison of the two algorithms: error behavior with h=1/32 (a) on a cube, (b) on a sphere.

For this example, the comparison of the error history between the original algorithm and the two-stage strategy is shown in Figure 8.6. In this example, on both domain there is no big improvements by the two-stage strategy.

(a)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	31	2.67×10^{-7}	3.13×10^{-3}		1.39×10^{-2}	
1/16	52	5.44×10^{-7}	1.07×10^{-3}	1.55	7.29×10^{-3}	0.93
1/32	182	9.64×10^{-7}	2.91×10^{-4}	1.88	2.99×10^{-3}	1.29

(b)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	15	4.14×10^{-7}	2.81×10^{-3}		1.56×10^{-2}	
1/16	41	8.49×10^{-7}	8.79×10^{-3}	1.68	6.60×10^{-3}	1.24
1/32	149	9.47×10^{-7}	2.57×10^{-4}	1.77	2.54×10^{-3}	1.38

(c)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	13	8.67×10^{-7}	3.87×10^{-3}		2.46×10^{-2}	
1/16	41	8.03×10^{-7}	1.09×10^{-3}	1.83	1.25×10^{-2}	0.98
1/32	156	9.87×10^{-7}	2.96×10^{-4}	1.88	5.36×10^{-3}	1.22

Table 8.3: (Example 3) Number of iterations and errors on cube and sphere with different mesh size. (a) is on a cube with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point. (b) is on a sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point. (c) is on a sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ not being a grid point.

8.5.3 Example 3

For the third example, we test the function such that there is a singular point in f:

$$g = \left(\frac{3}{5}\right)\left(\frac{3}{2}\right)^{\frac{1}{3}}\left(r^{\frac{5}{3}}-1\right), \tag{8.58}$$

$$f = \frac{1}{r} \tag{8.59}$$

where similar to Example 2, $r^2 = (x_1-1/2)^2+(x_2-1/2)^2+(x_3-1/2)^2$. In this example, $f\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ is infinity. In our experiment, if $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ is a grid point, we

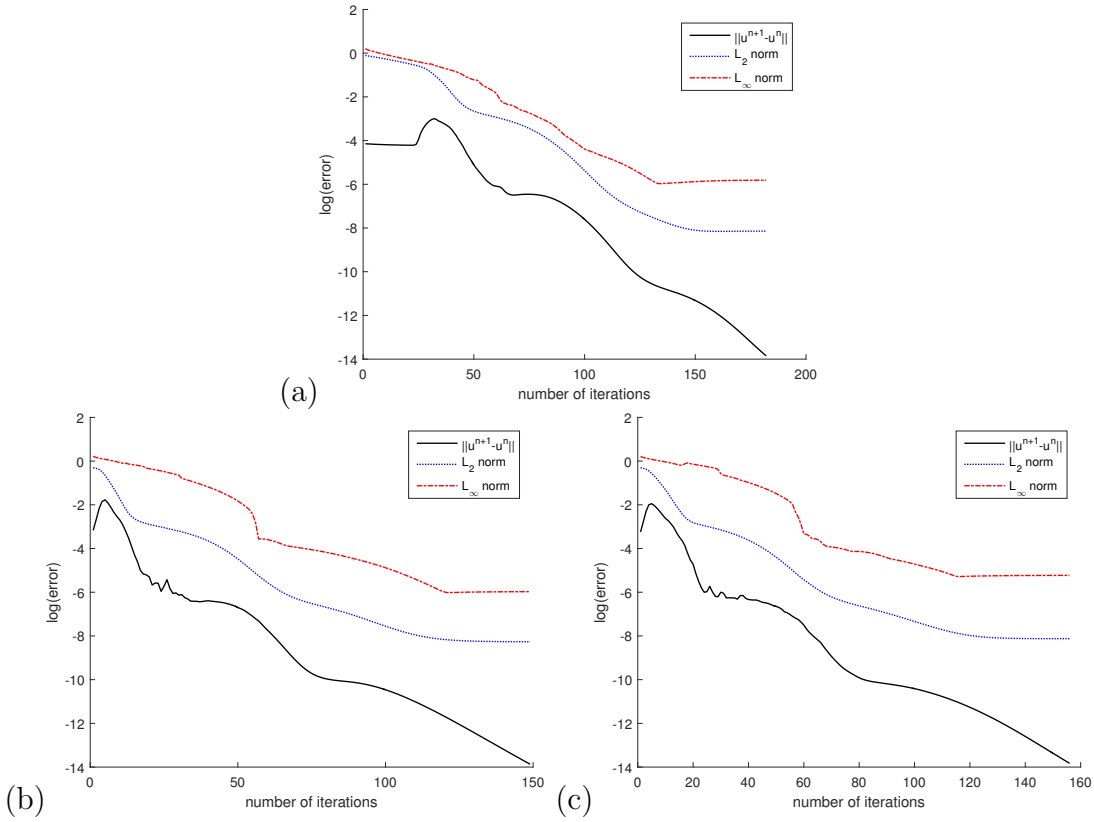


Figure 8.7: (Example 3) Error behavior with h=1/32. (a) is on a cube with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ being a grid point. (b) is on a sphere with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ being a grid point. (c) is on a sphere with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ not being a grid point.

use

$$f(\frac{1}{2},\frac{1}{2},\frac{1}{2}) = \frac{1}{h} \tag{8.60}$$

to approximate it. If $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ is not a grid point, we do not use any approximation. Because of the existence of the singular point in f, $\|\mathbf{D}^2u^n - \mathbf{p}^n\|$ and $\frac{\|\mathbf{D}^2u^n-\mathbf{p}^n\|}{\|\mathbf{p}^n\|}$ cannot go to very small. We use stopping criteria $\|u^{n+1}-u^n\| < 10^{-6}$. Our experiments are conducted on three domains: cube with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point, sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point and sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ not being a grid point. The number of iterations and errors with different mesh size is shown in Table 8.3. The error history with $h = 1/32$ on different mesh is shown in Figure 8.7. The convergence rate is more than 1.5 for L_2-error and more than 1.0 for L_∞-error. On the sphere, the mesh with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point provides smaller error than mesh with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ not being a grid point.

(a)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	65	9.79×10^{-7}	2.58×10^{-2}		1.90×10^{-1}	
1/16	361	9.95×10^{-7}	9.92×10^{-3}	1.38	1.17×10^{-1}	0.70
1/32	3156	9.98×10^{-7}	4.93×10^{-3}	1.01	6.91×10^{-2}	0.76

(b)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	29	9.58×10^{-7}	1.95×10^{-2}		1.18×10^{-1}	
1/16	116	9.77×10^{-7}	7.05×10^{-3}	1.47	9.73×10^{-2}	0.28
1/32	585	9.99×10^{-7}	3.78×10^{-3}	0.90	7.02×10^{-2}	0.47

(c)

h	n	$\|u^{n+1}-u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/8	28	7.68×10^{-7}	2.00×10^{-2}		1.24×10^{-1}	
1/16	109	9.66×10^{-7}	6.34×10^{-3}	1.66	8.82×10^{-2}	0.49
1/32	576	9.98×10^{-7}	1.95×10^{-3}	1.70	5.59×10^{-2}	0.66

Table 8.4: (Example 4) Number of iterations and errors on cube and sphere with different mesh size. (a) is on a cube with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point. (b) is on a sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point. (c) is on a sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ not being a grid point.

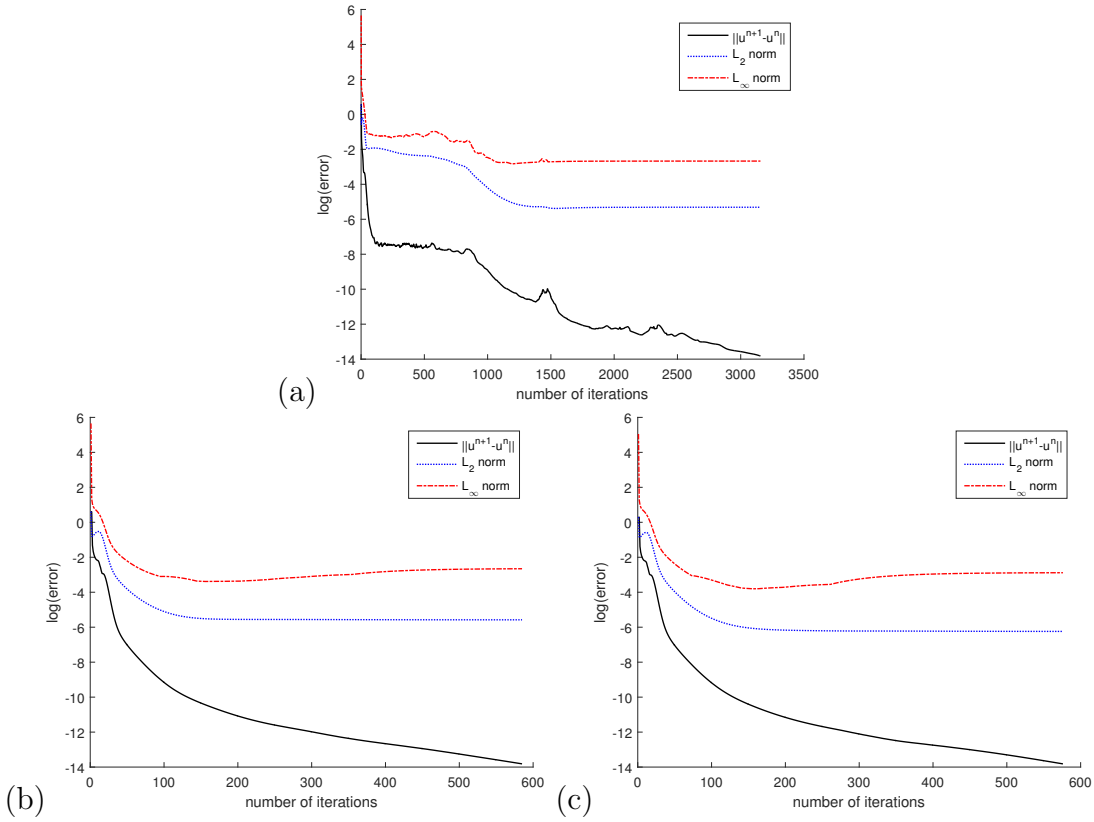


Figure 8.8: (Example 4) Error behavior with h=1/32. (a) is on a cube with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ being a grid point. (b) is on a sphere with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ being a grid point. (c) is on a sphere with $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$ not being a grid point.

8.5.4 Example 4

In the last example, we solve the problem with f being a delta function:

$$u = 3^{1/3}(r - \frac{1}{2}), \tag{8.61}$$

$$f = 4\pi\delta \approx \frac{3\epsilon^2}{(r^2+\epsilon^2)^{5/2}}, \tag{8.62}$$

where

$$r^2 = (x_1 - 1/2)^2 + (x_2 - 1/2)^2 + (x_3 - 1/2)^2.$$

To approximate f, we use the fundamental solution of Laplacian equation in 3D:

$$-\Delta\left(1/r\right) = 4\pi\delta_{\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)} \tag{8.63}$$

Introduce a small parameter ϵ_2, we have

$$\begin{aligned} f = 4\pi\delta &\approx -\Delta\left(1/\sqrt{\epsilon_2^2 + r^2}\right) \\ &= \frac{3\epsilon^2}{(r^2+\epsilon^2)^{5/2}}. \end{aligned} \tag{8.64}$$

Because of the non-smoothness of f, $\|\mathbf{D}^2 u^n - \mathbf{p}^n\|$ and $\frac{\|\mathbf{D}^2 u^n - \mathbf{p}^n\|}{\|\mathbf{p}^n\|}$ cannot go to very small. A smaller time step is required to make our algorithm converges. In our experiment, we choose $dt = h^2$ with stopping criteria $\|u^{n+1} - u^n\| < 10^{-6}$. Similar to example 3, we test our algorithm on three different meshes: cube with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point, sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ being a grid point and sphere with $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$ not being a grid point. The number of iterations and errors is shown in Table 8.4. With $h = 1/32$, the error history is shown in Figure 8.8. Our algorithm can solve this problem. The convergence rate for L_2-norm is above 1. For L_∞ norm, the convergence rate is below 1 and depends on the mesh.

8.5.5 Example 5

In this example, we test our algorithm on a cylinder domain $\Omega = \{(x_1, x_2, x_3) | x_1^2 + x_2^2 \leq 1, 0 \leq x_3 \leq H\}$. The figure of the domain and its cross section with $H = 1.5, h = 1/8$ is shown in Figure 8.9.

On this domain, first we solve

$$\begin{cases} \det \mathbf{D}^2 u = 1 \text{ in } \Omega, \\ u = 0 \text{ on } \partial\Omega. \end{cases} \tag{8.65}$$

We choose $h = 1/8$ and $H = 1, 1.5, 2$. We set the stopping criterion as $\|u^{n+1} - u^n\|_2 < 10^{-7}$. The comparison of the cross section along $\{x_1 = 0, x_2 = 0\}$ and $\{x_2 = 0, x_3 = H/2\}$ is shown in Figure 8.10. In all figures, these cross sections are convex. As H increases, the minimum value along these two lines become smaller.

Then we solve the exponential function on this cylinder mesh:

$$\begin{cases} \det \mathbf{D}^2 u = (1 + r^2) e^{\frac{3}{2} r^2} \text{ in } \Omega, \\ u = e^{r^2/2} \text{ on } \partial\Omega, \end{cases} \tag{8.66}$$

In this example, $r^2 = (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 + (x_3 - \frac{1}{2})^2$ and $r^2 = x_1^2 + x_2^2 + (x_3 - \frac{1}{2})^2$ are used. The exact solution is given as $u^* = e^{r^2/2}$. The L_2 and L_∞ errors and accuracy orders are shown in Table 8.5. For both cases, if the mesh is not too coarse, the accuracy order of the L_2 and L_∞ order is around 1.5.

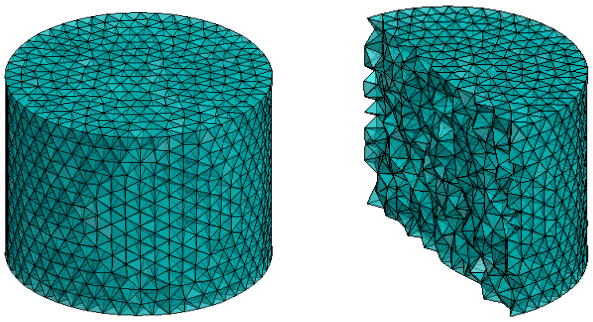


Figure 8.9: The cylinder domain and its cross section.

8.5.6 Example 6

The this example, we use a strictly convex domain with two singularities:

$$\Omega = \{\{x_1, x_2, x_3\} | \sqrt{x_1^2 + x_2^2 + x_3^2} < x_3(1 - x_3), 0 < x_3 < 1\}.$$

This domain looks like American football which is shown in Figure 8.11. On this domain we solve (8.65) with $h = 1/32$. The cross sections between the solution with the plane $x_3 = 0.5$ and $x_2 = 0$ are shown in Figure 8.12. Our solution is smooth and convex.

(a)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/4	31	6.24×10^{-8}	7.08×10^{-2}		9.56×10^{-2}	
1/8	46	9.66×10^{-8}	3.49×10^{-2}	1.02	5.58×10^{-2}	0.78
1/12	34	7.03×10^{-8}	1.89×10^{-2}	1.51	4.18×10^{-2}	0.71
1/16	57	4.29×10^{-8}	1.09×10^{-2}	1.91	2.95×10^{-2}	1.21
1/24	209	9.64×10^{-8}	5.06×10^{-3}	1.64	1.60×10^{-2}	1.51

(b)

h	n	$\|u^{n+1} - u^n\|$	L_2 norm	ratio	L_∞ norm	ratio
1/4	32	9.75×10^{-8}	3.22×10^{-2}		4.78×10^{-2}	
1/8	40	8.94×10^{-8}	1.61×10^{-2}	1.00	2.33×10^{-2}	1.04
1/12	38	6.54×10^{-8}	8.25×10^{-3}	1.65	1.10×10^{-2}	1.85
1/16	261	9.67×10^{-8}	4.80×10^{-3}	1.88	6.36×10^{-3}	1.90
1/24	115	9.88×10^{-8}	2.26×10^{-3}	1.86	2.97×10^{-3}	1.88

Table 8.5: (Example 5) L_2 and L_∞ errors of solving (8.66) with $H = 1$ and (a) $r^2 = (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 + (x_3 - \frac{1}{2})^2$ and (b) $r^2 = x_1^2 + x_2^2 + (x_3 - \frac{1}{2})^2$.

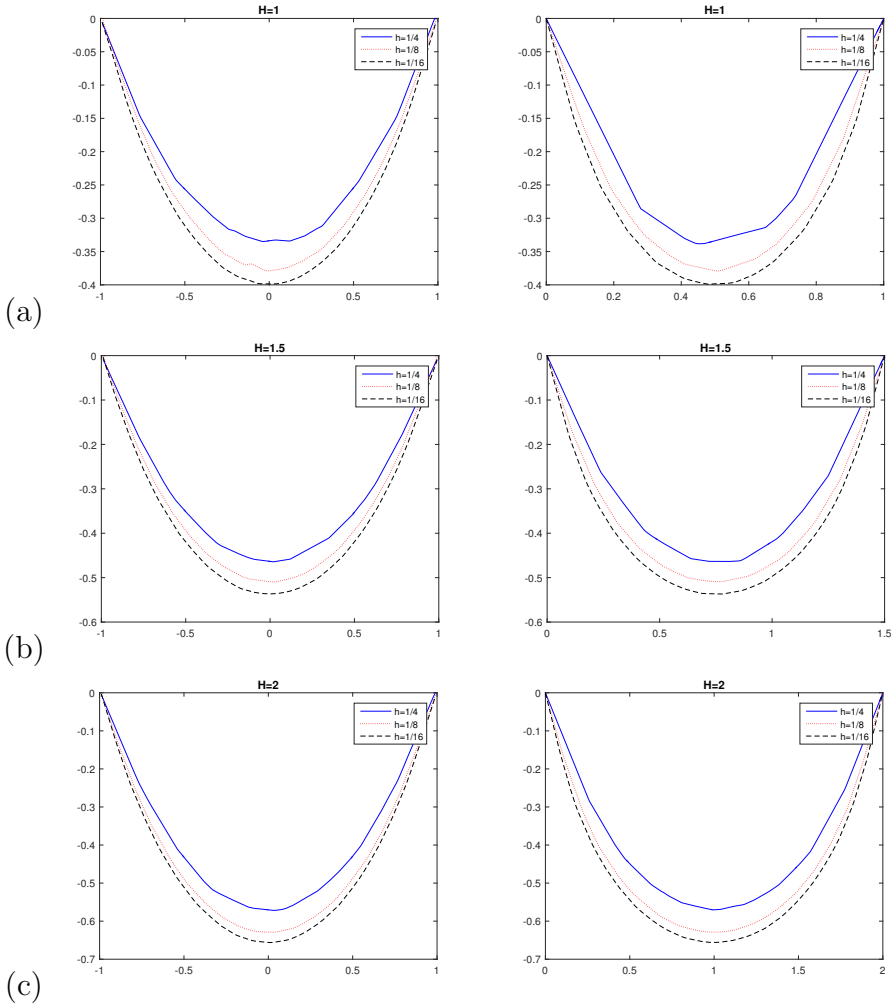


Figure 8.10: (Example 5) Cross sections of solutions along $\{x_1 = 0, x_2 = 0\}$ (left) and $\{x_2 = 0, x_3 = H/2\}$ (right) with (a) H=1, (b) H=1.5, (c) H=2.

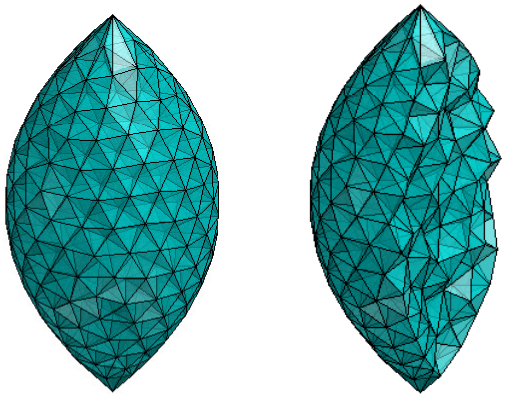


Figure 8.11: Mesh on the American football domain.

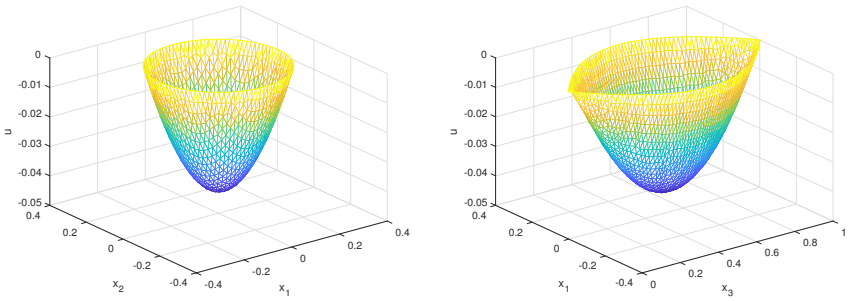


Figure 8.12: (Example 6) Cross sections between the result of (8.65) and the plane $x_3 = 0.5$ and $x_2 = 0$ on the American football mesh.

8.6 Conclusion

In this chapter, we have extended the operator-splitting method to solve the Monge-Ampère equation in three dimension. Our algorithm works well for different types of problem: problem with classical solution, problem with a singular point on the right hand side and when the right hand side contains the Dirac Delta function. Our algorithm can solve Monge-Ampère equation on various types of domains and meshes.

Chapter 9

Conclusion

This thesis discussed the application of two numerical methods on geometric problems: the level set method for dimension reduction problem and the operator splitting method for the Monge-Ampère equation.

In the first part, Chapter 2, a variational formulation for dimension reduction on Riemannian manifolds is proposed. Our algorithm is based on the level set method in a fully implicit formulation. So we do not need any a prior knowledge of the structure of the constructed curve. In the case that we want to construct an open curve, a modified algorithm is provided. We have tested the algorithm on various types of numerical examples with different manifolds. If the data has two patterns, we only need to initialize a curve large enough to enclose all data set, then it can auto-split into two patterns. Even with given measurements with noise or outliers, the proposed algorithm gives good and robust reconstructions.

In the second part, Chapter 3 to Chapter 8, a series of operator-splitting algorithms are proposed to solve Monge-Ampère type problems. The Monge-Ampère equation originates from the so called Minkowski problem from differential geometry. The algorithm in Chapter 3 is the basis of other chapters. In Chapter 3, a relatively easy to implement finite element and operator-splitting based

method for the numerical solution of the classical two dimensional Dirichlet Monge-Ampère equation is proposed. This method is based on an equivalent divergence form of the Monge-Ampère equation. The solution is obtained by solving a time dependent PDE system until steady state. The problem is discretized by the operator-splitting method in time and by the mixed finite element method in space. The related method has been working well for various types of triangulations (structured and unstructured) and can handle curved boundary quite easily. We introduced also a Newton-like two-stage variant of our methodology, which accelerates significantly the convergence if the problem under consideration has a smooth convex solution. In Chapter 4, the algorithm in Chapter 3 is modified to solve the Minkowski problem. Our method works well for various examples. Then our algorithm is modified to solve a series of variant of the classical Monge-Ampère equation: the obstacle Monge-Ampère problem, the degenerate Monge-Ampère problem, the Neumann Monge-Ampère problem and the three dimensional Monge-Ampère problem in Chapter 5, Chapter 6, Chapter 7 and Chapter 8 respectively. For all problems, our algorithms can provide smooth convex solutions. Among most of them, the convergence rate is computed and varies form 1 to 2.

Bibliography

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *14th ACM symposium on Computational Geometry*, 1998.

[2] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76, 1998.

[3] G. Awanou. Standard finite elements for the numerical resolution of the elliptic Monge-Ampère equation: classical solutions. *IMA Journal of Numerical Analysis*, 35(3):1150–1166, 2014.

[4] I. J. Bakel'Man. The dirichlet problem for the elliptic n-dimensional monge–ampere equations and related problems in the theory of quasilinear equations. In *Proceedings of Seminar on Monge–Ampere Equations and Related Topics, Firenze*, pages 1–78, 1980.

[5] I.J. Bakelman. *Convex Analysis and Nonlinear Geometric Elliptic Equations*. Springer-Verlag, Berlin, 1994.

[6] J.D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[7] J.D. Benamou, B. D. Froese, and A. M. Oberman. Two numerical methods for the elliptic Monge–Ampère equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(4):737–758, 2010.

[8] M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174:759–780, 2001.

[9] K. Böhmer. On finite element methods for fully nonlinear elliptic equations of second order. *SIAM Journal on Numerical Analysis*, 46(3):1212–1249, 2008.

[10] Y. Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991.

[11] S. Brenner, T. Gudi, M. Neilan, and L. Sung. C^0 penalty methods for the fully nonlinear Monge-Ampère equation. *Mathematics of Computation*, 80(276):1979–1995, 2011.

[12] S. C. Brenner and M. Neilan. Finite element approximations of the three dimensional Monge-Ampère equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(5):979–1001, 2012.

[13] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher. Fast global minimization of the active Contour/Snake model. *Journal of Mathematical Imaging and Vision*, 28:151–167, 2007.

[14] P. Burchard, L.-T. Cheng, B. Merriman, and S. Osher. Motion of curves in three spatial dimensions using a level set approach. *J. Comput. Phys.*, 170(2):720–741, 2001.

[15] A. Caboussat, R. Glowinski, and D. C. Sorensen. A least-squares method for the numerical solution of the Dirichlet problem for the elliptic Monge–Ampère equation in dimension two. *ESAIM: Control, Optimisation and Calculus of Variations*, 19(3):780–810, 2013.

[16] L. A. Caffarelli and M. Milman. *Monge–Ampère Equation: Applications to Geometry and Optimization: NSF-CBMS Conference on the Monge–Ampère Equation, Applications to Geometry and Optimization, July 9-13, 1997, Florida Atlantic University*, volume 226. American Mathematical Soc., Providence, RI, 1999.

[17] L.A. Caffarelli and X. Cabre. *Fully Nonlinear Elliptic Equations*. Amer. Math. Soc., Providence, RI, 1995.

[18] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[19] T.F. Chan, S. Esedoḡlu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM J. Appl. Math.*, 66:1632–1648, 2006.

[20] W. Chen, C.S. Chou, and C.Y. Kao. Lax-Friedrichs fast sweeping methods for steady state problems for hyperbolic conservation laws. *J. Comput. Phys.*, 234:452–471, 2013.

[21] L. T. Cheng. Construction of shapes arising from the minkowski problem using a level set approach. *Journal of scientific computing*, 19(1-3):123–138, 2003.

[22] L.-T. Cheng. Efficient level set methods for constructing wavefronts in three spatial dimensions. *J. Comput. Phys.*, 226:2250–2270, 2007.

[23] L.-T. Cheng, P. Burchard, B. Merriman, and S. Osher. Motion of curves constrained on surfaces using a level-set approach. *J. Comput. Phys.*, 175:604–644, 2002.

[24] S. Y. Cheng and S. T. Yau. On the regularity of the solution of the n-dimensional minkowski problem. *Communications on Pure and Applied Mathematics*, 29(5):495–516, 1976.

[25] P. Daskalopoulos and K. Lee. Fully degenerate monge-ampére equations. *Journal of Differential Equations*, 253(6):1665–1691, 2012.

[26] E. J. Dean and R. Glowinski. Numerical solution of the two-dimensional elliptic Monge–Ampère equation with Dirichlet boundary conditions: an augmented Lagrangian approach. *Comptes Rendus Mathematique*, 336(9):779–784, 2003.

[27] E. J. Dean and R. Glowinski. Numerical solution of the two-dimensional elliptic Monge–Ampère equation with Dirichlet boundary conditions: a least-squares approach. *Comptes Rendus Mathematique*, 339(12):887–892, 2004.

[28] E. J. Dean and R. Glowinski. An augmented Lagrangian approach to the numerical solution of the Dirichlet problem for the elliptic Monge–Ampère equation in two dimensions. *Electronic Transactions on Numerical Analysis*, 22:71–96, 2006.

[29] E. J. Dean and R. Glowinski. Numerical methods for fully nonlinear elliptic equations of the Monge–Ampère type. *Computer Methods in Applied Mechanics and Engineering*, 195(13):1344–1386, 2006.

[30] E. J. Dean and R. Glowinski. On the numerical solution of the elliptic Monge–Ampère equation in dimension two: A least-squares approach. In *Partial Differential Equations*, pages 43–63. Springer, 2008.

[31] E. J. Dean, R. Glowinski, and T.W. Pan. Operator-splitting methods and applications to the direct numerical simulation of particulate flow and to the solution of the elliptic Monge–Ampère equation. In *Control and Boundary Analysis*, pages 1–27. CRC Press, 2005.

[32] H. Edelsbrunner. Shapre reconstruction with Delaunay complex. In *Proc. of LATIN'98: Theoretical Informatics*, volume 1380. Springer-Verlag, 1998.

[33] X. Feng, R. Glowinski, and M. Neilan. Recent developments in numerical methods for fully nonlinear second order partial differential equations. *SIAM Review*, 55(2):205–267, 2013.

[34] X. Feng and M. Neilan. Mixed finite element methods for the fully nonlinear Monge–Ampère equation based on the vanishing moment method. *SIAM Journal on Numerical Analysis*, 47(2):1226–1250, 2009.

[35] X. Feng and M. Neilan. Vanishing moment method and moment solutions for fully nonlinear second order partial differential equations. *Journal of Scientific Computing*, 38(1):74–98, 2009.

[36] P.T. Fletcher and S. Joshi. Riemannian Geometry for the Statistical Analysis of Diffusion Tensor Data. *Signal Processing*, 87:250–262, 2007.

[37] P.T. Fletcher, C. Lu, S.M. Pizer, and S. Joshi. Principal Geodesic Analysis for the Study of Nonlinear Statistics of Shape. *IEEE Trans. Med. Imaging*, 23:995–1005, 2004.

[38] B. D. Froese and A. M. Oberman. Convergent finite difference solvers for viscosity solutions of the elliptic Monge–Ampère equation in dimensions two and higher. *SIAM Journal on Numerical Analysis*, 49(4):1692–1714, 2011.

[39] B. D. Froese and A. M. Oberman. Fast finite difference solvers for singular solutions of the elliptic Monge–Ampère equation. *Journal of Computational Physics*, 230(3):818–834, 2011.

[40] B.D. Froese. A numerical method for the elliptic Monge-Ampère equation with transport boundary conditions. *SIAM J. Sci. Comput.*, 34(3):A1432–A1459, 2012.

[41] D. Gilbarg and N.S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Springer-Verlag Berlin, 1983.

[42] R. Glowinski. *Numerical Nethods for Nonlinear Variational Problems*. Springer, New York, NY, 1984(2^{nd} printing: 2008).

[43] R. Glowinski. Numerical methods for fully nonlinear elliptic equations. In *6th International Congress on Industrial and Applied Mathermatics, ICIAM*, volume 7, pages 155–192, 2009.

[44] R. Glowinski. *Variational Methods for the Numerical Solution of Nonlinear Elliptic Problems*, volume 86. SIAM, Philadelphia, PA, 2015.

[45] R. Glowinski, S. Leung, and J. Qian. A penalization-regularization-operator splitting method for eikonal based traveltime tomography. *SIAM J. Imaging Sciences*, 8(2):1263–1292, 2015.

[46] R. Glowinski, S. Leung, and J. Qian. Operator-splitting based fast sweeping methods for isotropic wave propagation in a moving fluid. *SIAM J. Sci. Comp.*, 38(2):A1195–A1223, 2016.

[47] R. Glowinski, S. Osher, and W. Yin (Eds.). *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016.

[48] R. Glwoinski, H. Liu, S. Leung, and J. Qian. A finite element/operator-splitting method for the numerical solution of the two dimensional elliptic Monge-Ampère equation. *submitted*.

[49] A. Goh and R. Vidal. Clustering and dimensionality reduction on Riemannian manifolds. *CVPR 2008*, 2008.

[50] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[51] C.R. Goodall. Procrustes Methods in the Statistical Analysis of Shape. *Journal of the Royal Statistical Society, Series B*, 53:285–339, 1991.

[52] C. E. Gutiérrez. *The Monge-Ampère Equation*. Birkhaüser, Basel,, 2001.

[53] T. Hastie and W. Stuetzle. Principal curves. *J. Amer. Stat. Assoc.*, 84(406):502–516, 1989.

[54] S. Hauberg. Principal curves on Riemannian manifolds. *IEEE Trans. on Pattern Anal. Mach. Intell.*, Accepted, 2015.

[55] S. Huckemann and H. Ziezold. Principal Component Analysis for Riemannian Manifolds, With an Application to Triangular Shape Spaces. *Advances in Applied Probability*, 38:299–319, 2006.

[56] G. S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[57] S. Jung, I. L. Dryden, and J. S. Marron. Analysis of principal nested spheres. *Biometrika*, 99:551–568, 2012.

[58] C.Y. Kao, S.J. Osher, and J. Qian. Lax-Friedrichs sweeping schemes for static Hamilton-Jacobi equations. *J. Comp. Phys.*, 196:367–391, 2004.

[59] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[60] K. Kenobi, I. L. Dryden, and H. Le. Shape curves and geodesic modelling. *Biometrika*, 97:567–584, 2010.

[61] M. Knott and C. S. Smith. On the optimal mapping of distributions. *Journal of Optimization Theory and Applications*, 43(1):39–49, 1984.

[62] A. Kume, I. L. Dryden, and H. Le. Shape-space smoothing splines for planar landmark data. *Biometrika*, 94:513–528, 2007.

[63] L. Lamberg. *On the Minkowski problem and the lightcurve operator*. dissertation, Academia Scientiarum Fennica, Series A. I., 1993.

[64] L. Lamberg and M. Kaasalainen. Numerical solution of the minkowski problem. *Journal of computational and applied mathematics*, 137(2):213–227, 2001.

[65] S.-M. Lee, A.L. Abbott, and P.A. Araman. Dimensionality reduction and clustering on statistical manifolds. *CVPR 2007*, 2007.

[66] S. Leung and S. Osher. An alternating direction explicit (ADE) scheme for time-dependent evolution equations. *Preprint UCLA June*, 9:2005, 2005.

[67] S. Leung and S. Osher. Fast global minimization of the active contour model with tv-inpainting and two-phase denoising. *Proceeding of the 3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 149–160, 2005.

[68] S. Leung and J. Qian. An adjoint state method for 3d transmission traveltime tomography using first arrival. *Commun. Math. Sci.*, 4:249–266, 2006.

[69] H. Lewy. A priori limitations for solutions of monge-ampere equations. *Transactions of the American Mathematical Society*, 37(3):417–434, 1935.

[70] H. Lewy. On differential geometry in the large, i (minkowski's problem). *Transactions of the American Mathematical Society*, 43(2):258–270, 1938.

[71] W.B. Li and S. Leung. A fast local level set adjoint state method for first arrival transmission traveltime tomography with discontinuous slowness. *Geophs. J. Int.*, 195(1):582–596, 2013.

[72] W.B. Li, S. Leung, and J. Qian. A level-set adjoint-state method for crosswell transmission-reflection traveltime tomography. *Geophs. J. Int.*, 199(1):348–367, 2014.

[73] J. Liang, F. Park, and H. Zhao. Robust and efficient implicit surface reconstruction for point clouds based on convexified image segmentation. *J. Sci. Comput.*, 54:577–602, 2013.

[74] T. Lin, H. Zha, and S.U. Lee. Riemannian manifold learning for nonlinear dimensionality reduction. *ECCV 2006*, 2006.

[75] P. L. Lions. Sur les equations de monge-ampere. *Archive for rational mechanics and analysis*, 89(2):93–122, 1985.

[76] J. J. Little. An iterative method for reconstructing convex polyhedra from external guassian images. In *AAAI*, pages 247–250, 1983.

[77] J. J. Little. *Recovering shape and determining attitude from extended Gaussian images*. PhD thesis, University of British Columbia, 1985.

[78] X. D. Liu, S. J. Osher, and T. Chan. Weighted Essentially NonOscillatory schemes. *J. Comput. Phys.*, 115:200–212, 1994.

[79] G. Loeper and F. Rapetti. Numerical solution of the Monge-Ampère equation by a Newton algorithm. *C. R. Acad. Sci. Paris, Ser. I*, 340:319–324, 2005.

[80] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

[81] F. Memoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *J. Comput. Phys.*, 173:730–764, 2001.

[82] C. Min. Local level set methods in high dimension and codimension. *Journal of Computational Physics*, 200(1):368–382, 2004.

[83] W. Ming and J. Xu. The morley element for fourth order elliptic equations in any dimensions. *Numerische Mathematik*, 103(1):155–169, 2006.

[84] H. Minkowski. Allgemeine lehrsätze über die konvexen polyeder. In *Ausgewählte Arbeiten zur Zahlentheorie und zur Geometrie*, pages 121–139. Springer, 1989.

[85] H. Minkowski. Volumen und oberfläche. In *Ausgewählte Arbeiten zur Zahlentheorie und zur Geometrie*, pages 146–192. Springer, 1989.

[86] J.M. Mirebeau. Discretization of the 3d monge- ampère operator, between wide stencils and power diagrams. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(5):1511–1523, 2015.

[87] B. Mohammadi. Optimal transport, shape optimization and global minimization. *Comptes Rendus Mathematique*, 344(9):591–596, 2007.

[88] M. Neilan. A nonconforming morley finite element method for the fully nonlinear monge-ampère equation. *Numerische Mathematik*, 115(3):371–394, 2010.

[89] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. 27(2):372–376, 1983.

[90] S. Ninomiya and N. Victoir. Weak approximation of stochastic differential equations and application to derivative pricing. *Applied Mathematical Finance*, 15(2):107–121, 2008.

[91] L. Nirenberg. The weyl and minkowski problems in differential geometry in the large. *Communications on Pure and Applied Mathematics*, 6(3):337–394, 1953.

[92] A. M. Oberman. Wide stencil finite difference schemes for the elliptic monge-ampere equation and functions of the eigenvalues of the hessian. *Discrete Contin. Dyn. Syst. Ser. B*, 10(1):221–238, 2008.

[93] V.I. Oliker and L.D. Prussner. On the numerical solution of the equation $\frac{\partial^2 z}{\partial x^2}\frac{\partial^2 z}{\partial y^2} - \left(\frac{\partial^2 z}{\partial x \partial y}\right)^2 = f$ and its discretizations, i. *Numerische Mathematik*, 54(3):271–293, 1989.

[94] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 57. 2004.

[95] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[96] S. J. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[97] V.M. Panaretoes, T. Pham, and Z. Yao. Principal flows. *J. Amer. Stat. Assoc.*, 109(505):424–436, 2014.

[98] D. Peng, B. Merriman, S. Osher, H. K. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.

[99] P.O. Persson and G. Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, 2004.

[100] A. V. Pogorelov. Regularity of a convex surface with given gaussian curvature. *Matematicheskii Sbornik*, 73(1):88–103, 1952.

[101] A. V. Pogorelov. The minkowski multidimensional problem. scripta series in mathematics. J. Wiley, 1978.

[102] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[103] O. Savin. The obstacle problem for Monge–Ampère equation. *Calculus of Variations and Partial Differential Equations*, 22(3):303–320, 2005.

[104] H. Schaeffer and T. Y. Hou. An accelerated method for nonlinear elliptic pde. *Journal of Scientific Computing*, 69(2):556–580, 2016.

[105] J. A. Sethian. *Level set methods*. Cambridge Univ. Press, second edition, 1999.

[106] C. W. Shu and S. J. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[107] P. Smereka. Spiral crystal growth. *Physica D*, 138:282–301, 2000.

[108] D. C. Sorensen and R. Glowinski. A quadratically constrained minimization problem arising from pde of monge–ampère type. *Numerical Algorithms*, 53(1):53–66, 2010.

[109] G. Strang and G. J. Fix. *An Analysis of The Finite Element Method*, volume 212. Prentice-hall Englewood Cliffs, NJ, 1973.

[110] J. Su, I. L. Dryden, E. Klassen, H. Le, and A. Srivastava. Fitting smoothing splines to time-indexed, noisy points on nonlinear manifolds. *Image and Vision Computing*, 30:428–442, 2012.

[111] W. Su, S. Boyd, and E. Candes. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. pages 2510–2518, 2014.

[112] M. Sulman, J. F. Williams, and R. D. Russell. Optimal mass transport for higher dimensional adaptive grid generation. *Journal of Computational Physics*, 230(9):3302–3330, 2011.

[113] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via anisotropic diffusion of normals. *Proceedings of IEEE Visualization 2002*, pages 125–132, 2002.

[114] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. *ACM Transactions on Graphics*, 22, 2003.

[115] N. S. Trudinger and J. I. E. Urbas. The dirichlet problem for the equation of prescribed gauss curvature. *Bulletin of the Australian Mathematical Society*, 28(2):217–231, 1983.

[116] N. S. Trudinger and X. J. Wang. The monge-ampere equation and its geometric applications. *Handbook of geometric analysis*, 1:467–524, 2008.

[117] J. I. E. Urbas. The equation of prescribed gauss curvature without boundary conditions. *Journal of Differential Geometry*, 20(2):311–327, 1984.

[118] J. I. E. Urbas. Global hölder estimates for equations of monge-ampere type. *Inventiones mathematicae*, 91(1):1–29, 1988.

[119] J. I. E. Urbas. Boundary regularity for solutions of the equation of prescribed gauss curvature. In *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*, volume 8, pages 499–522. Elsevier, 1991.

[120] L.J.P. Van der Maaten, E.O. Postma, and H.J. Van den Herik. Dimensionality reduction: a comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[121] T. Wong and S. Leung. A fast sweeping method for eikonal equations on implicit surfaces. *Accepted by J. Sci. Comput.*, 2015.

[122] Z. Yao and T. Pham. Principal sub-manifolds. *Manuscript*, 2016.

[123] H. K. Zhao. Fast sweeping method for eikonal equations. *Math. Comp.*, 74:603–627, 2005.

[124] H.-K. Zhao, T. Chan, B. Merriman, and S. J. Osher. A variational level set approach for multiphase motion. *J. Comput. Phys.*, 127:179–195, 1996.

[125] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *1st IEEE Workshop on Variational and Level Set Methods, 8th ICCV*, pages 194–202, 2001.

[126] H.K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80:295–319, 2000.