Adaptive Grid Generation based on Monge-Kantorovich Optimization in Two and Three Dimensions

by

Weixi Gu

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Justin W.L. Wan

Waterloo, Ontario, Canada, 2017

Abstract

Adaptive grid generation has a wide range of applications in the areas of fluid mechanics, aerodynamics and heat transfer. It gives improvements in accuracy and efficiency of the numerical computation of partial differential equations. In adaptive grid generation, equidistribution has traditionally been a fundamental principle according to the literatures.

In this thesis, we propose a numerical method for generating two and three dimensional equidistribution grids based on Monge-Kantorovich optimization, and we implement algorithms based on our method. The procedure of our method involves solving a Monge-Ampère equation using Hamilton-Jacobi-Bellman formulation. The algorithms are tested with different examples. According to the numerical experiments, the algorithms generate grids according to the equidistribution principle; the shapes of the generated grids match the geometric features of the densities prescribed in the examples.

Acknowledgements

I would like to thank my supervisor Prof. Justin W.L. Wan for his valuable advice and firm support, without whom this work would not be possible. I would also like to thank my second reader Prof. Christopher Batty for his helpful comments and insightful feedback.

I thank Yangang Chen for kindly discussing his papers with me and being a very supportive friend.

Last but not the least, I would like to thank my family for their unconditional love and support.

Dedication

To my beloved husband Joseph.

Contents

List of Tables

List of Figures

List of Notations

Ω	a bounded convex domain, $\Omega \subset \mathbb{R}^n$
$\partial\Omega$	boundary of Ω
$\mathbf{n}$	the unit vector normal to $\partial\Omega$
ρ, ρ'	bounded density functions
ϕ	Coordinate transformation, $\phi \in \mathbb{R}^{n\times 1}$
θ, φ, ψ	angles
u	convex scalar potential field, $u \in C(\Omega)$
U	discrete approximation of u
∇u	gradient of u
D^2u	Hessian matrix of u
$\det[A]$	determinant of a matrix A
$\mathrm{Tr}[A]$	trace of a matrix A
$A \succ 0$	A is symmetric and positive definite
$A \succeq 0$	A is symmetric and positive semi-definite
$A \prec 0$	A is symmetric and negative definite
$A \preceq 0$	A is symmetric and negative semi-definite
$\mathrm{diag}(b_i)_{n\times n}$	an $n \times n$ diagonal matrix $\begin{pmatrix} b_1 & & & \\ & b_2 & & \\ & & \ddots & \\ & & & b_n \end{pmatrix}$

Chapter 1

Introduction

Many problems of practical interest in the natural science, engineering and even finance can be modeled and analyzed using partial differential equations (PDE). PDEs are a powerful family of equations that describe how objects or quantities change over time under other variables such as gravity, force and viscosity. Most PDEs are difficult to solve analytically. This means that for many problems, there exists no natural “closed-form” expression which is easily computed and understood in terms of the input quantities. In other words, the solutions that we care about are implicitly described.

Typically, we approximate solutions to PDEs that cannot be solved analytically using a popular technique called discretization. Discretization involves sampling the problem at a fixed number of points and simulating the equations that describe the behavior. If enough points are chosen, then simulating the equations will provide meaningful answers. Such domains of sample points are often called grids in the literature. In a single dimension, a grid corresponds to points on an arc; in higher dimensions, a grid corresponds to points in some coordinate system.

A grid with its points evenly distributed is called a uniform grid. However, in many applications, non-uniform grids in which points are not evenly distributed are more useful. For instance, in modeling of hurricane and other storms, the physical processes (such as the change of atmospheric pressure and the movement of convection currents) involved in each region may vary widely. Active regions typically require densely spaced grid points to capture features of interest, while less active regions can be coarsely simulated using widely spaced grid points (Weller et al [29], Mandli and Dawson [22]). To meet the computational requirements of such problems, the use of adaptive grid generation is required.

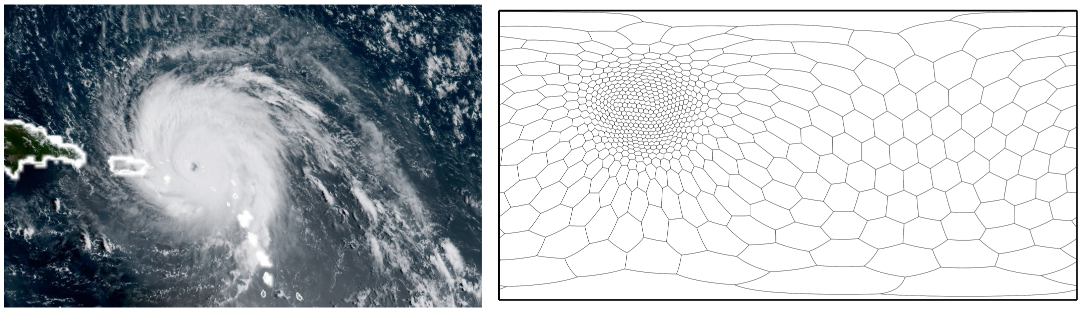


Figure 1.1: Left: hurricane storm surge (Image source: Berger [2]). Right: adaptive grid for storm surge model. (Image source: Budd [3])

An important notion in adaptive grids is that of equidistribution; that is a given quantity being distributed uniformly along cells in the grid. Equidistribution has traditionally been a fundamental principle in adaptive grid generation (Huang et at [15], Thompson [26]). The concept of equidistribution principle is first introduced by de Boor [10] for solving boundary value problems for ordinary differential equations, involves choosing grid points so that some measure of the numerical error of approximating a function using piecewise polynomial functions (splines) is equalized over each subinterval. Huang et al [15] show that this is a superior guiding principle for adaptive grid generation and leads to improvements in accuracy and efficiency of the numerical computation of PDEs.

In one dimension (Thompson [26], Eiseman [12], Liseikin [21]), the grid constrained to a given equidistribution principle exists uniquely, since each cell of the grid contains only one unknown that can be uniquely determined by the specified Jacobian of the grid transformation. In higher dimensions, there can be many possible grids that satisfy a given equidistribution principle; additional constraints are typically required for selecting a grid that is optimal according to some user criteria among all the possible grids. Such notions of optimality include a grid transformed from a uniform grid using the least amount of physical energy.

A number of different methods have been developed in the past for generating equidistributed grid in two or higher dimensions. Huang and Sloan [16] developed an adaptive grid method in two dimensions by considering equidistribution along a one-dimensional grid in the multi-dimensional domain. The approach is attractive as it decomposes the task of multidimensional grid generation into a series of one-dimensional grid equidistribution. However, the grids obtained by this method are only locally equidistributed and may even be folded (Huang and Sloan [16]). Another example is the deformation method proposed by Liao and Anderson [19] based on the work of Moser [23, 9]. The method involves computing the new positions of the points of an existing grid according to a system of ordinary differential equations (ODE). The drawback of this method is that there exists great latitude in choosing the ODE system, and for a fixed ODE system the resulting grid does not appear optimal in any sense (Delzanno et al [11]).

Recently proposed numerical methods based on Monge-Kantorovich (MK) optimization for optimal grid generation include the work of Delzanno et al [11], Sulman et al [25] and Weller et at [28]. The common idea of these methods is to minimize a grid quality measure derived from the MK transportation problem and constrained locally by the equidistribution principle, which involves solving a Monge-Ampère (MA) equation resulting from the MK optimization. While these methods have many common characteristics, the difference between the methods, on the other hand, appears in variations of the resulting MA equations and the techniques employed for solving the MA equations. For example, the method proposed by Delzanno et al [11] involves solving an elliptic MA equation numerically with a Jacobian-Free Newton-Krylov method. Sulman et al [25] compute the grid transformation by integrating a parabolic MA equation to steady state at every step. Weller et al [28] compute the grid transformation on a sphere by computing a semi-implicit solution of a parabolic MA equation.

In this thesis, we propose a numerical method for two and three dimensional grid equidistribution based on Monge-Kantorovich optimization, and we implement algorithms based on our method. The procedure of our method involves solving an elliptic MA equation using Hamilton-Jacobi-Bellman (HJB) formulation. The advantage of our approach is that the computation of the HJB formulation of the MA equation is numerically amenable, since the differential operator of the HJB equation under fixed controls is linear, which will be demonstrated in details in Chapter 3. We will show that, based on the numerical experiments, our algorithms generate grids for prescribed densities according to the equidistribution principle; the shapes of the grids match the geometric features of the prescribed densities.

The remainder of this thesis is organized as follows. In Chapter 2, we provide the necessary background knowledge about grid generation problem and basic concepts of the method that we use to solve the problem. In Chapter 3, we present the algorithms for solving the two-dimensional and three-dimensional grid generation problems. In Chapter 4, we discuss the results of our numerical experiments. Finally, we give a conclusion of our work in Chapter 5.

Chapter 2

Background

Our objective is to generate adaptive grids in two and three dimensions according to the equidistribution principle. The numerical method proposed in this thesis for solving the problem is based on MK optimization. In this chapter, we provide background knowledge on the concepts of our method. We start by explaining the mathematical formulation of the grid generation problem and the connection between the MK optimal transport model and the MA equation that we attempt to solve. Then we introduce the HJB formulation of the MA equation and the related concepts.

2.1 Monge-Kantorovich Optimization

The grid generation problem considers generating a grid that evenly distribute a given quantity along its cells. The mathematical formulation of the problem is as follows. Let ρ' be a prescribed positive, non-uniform density function on a physical domain $\Omega_P \subset \mathbb{R}^n$, where n is the dimension of the space. Let $\mathbf{x}$ be a uniform grid in a computational domain $\Omega_C \subset \mathbb{R}^n$ and $d^n\mathbf{x}$ and ρ be a constant density function on Ω_C such that ρ is equidistributed over $\mathbf{x}$, i.e,

$$\rho(\mathbf{x})d^n\mathbf{x} = \text{ constant},$$

where $\rho(\mathbf{x})d^n\mathbf{x}$ is the mass in each cell $d^n\mathbf{x}$ of the grid $\mathbf{x}$ (we note that $d^n\mathbf{x} \equiv dxdy$ in two dimensions and $d^n\mathbf{x} \equiv dxdydz$ in three dimensions). See Figure 2.1.

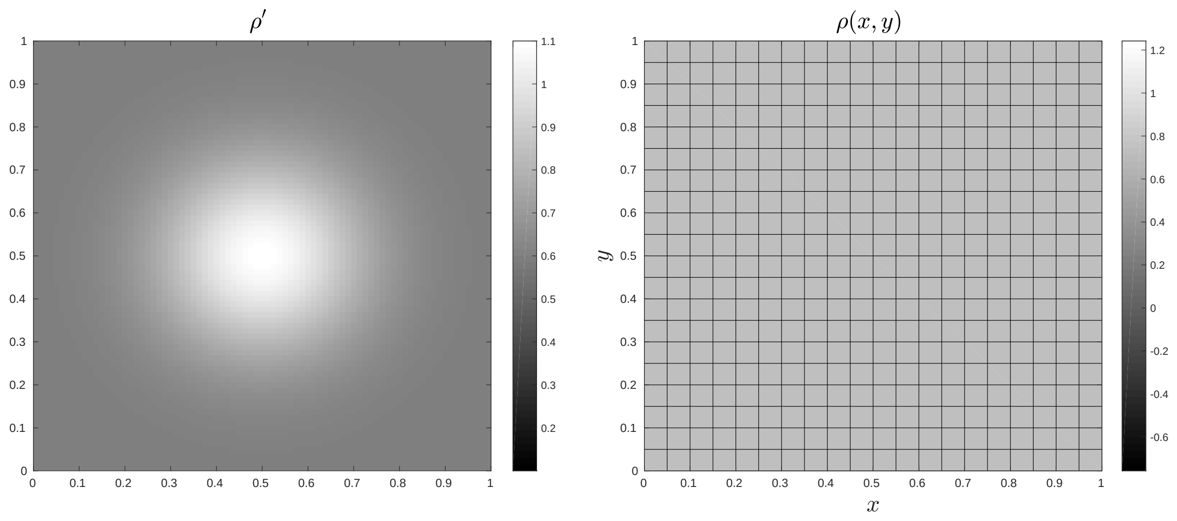


Figure 2.1: Left: the image of a non-uniform density function $\rho' > 0$ on a physical domain $\Omega_P = [0,1] \times [0,1]$. Right: the image of a constant density $\rho > 0$ equidistributed by a uniform grid $\mathbf{x} \in \Omega_C = [0,1] \times [0,1]$.

The goal is to find a coordinate transformation $\phi : \Omega_C \to \Omega_P$ of $\mathbf{x}$ such that the new grid $\mathbf{x}' = \phi(\mathbf{x})$ equidistributes the density ρ' on Ω_P:

$$\rho'(\mathbf{x}')d^n\mathbf{x}' = \rho(\mathbf{x})d^n\mathbf{x} = \text{ constant.} \tag{2.1}$$

Accordingly, we have

$$\int_{\mathbf{x}' \in \Omega_P} \rho'(\mathbf{x}')d^n\mathbf{x}' = \int_{\mathbf{x} \in \Omega_C} \rho(\mathbf{x})d^n\mathbf{x} = \text{ constant.} \tag{2.2}$$

Differentiating both sides of Equation (2.2), we get $\rho'(\mathbf{x}')\det[\mathcal{J}(\mathbf{x}')] = \rho(\mathbf{x})$, or equivalently,

$$\det[\mathcal{J}(\mathbf{x}')]\frac{\rho'(\mathbf{x}')}{\rho(\mathbf{x})} = 1, \tag{2.3}$$

where $\mathcal{J}(\mathbf{x}')$ is the $n \times n$ Jacobian matrix $\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}$ and $\det[\mathcal{J}(\mathbf{x}')]$ is the Jacobian determinant. Equation (2.3) is called the equidistribution principle; it ensures the generated grid $\mathbf{x}'$ is equidistributed according to the prescribed densities ρ' and ρ. The scalar function $\frac{\rho'(\mathbf{x}')}{\rho(\mathbf{x})}$ is also known as a monitor function in the literature.

Figures 2.2 and 2.3 are sketchs of the grid $\mathbf{x}'$ generated according to ρ and ρ'.

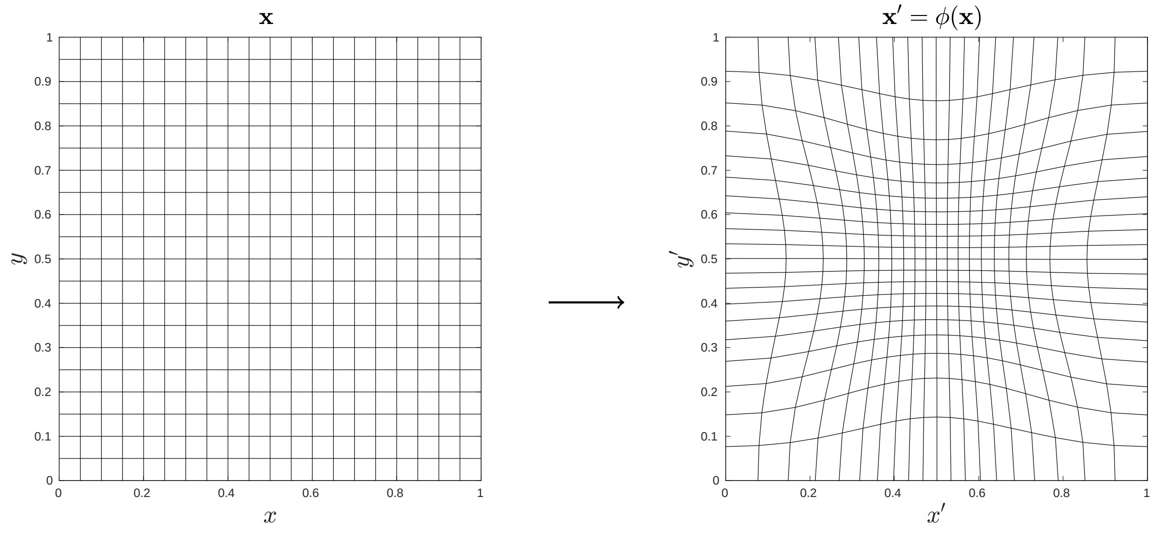


Figure 2.2: Sketch of a coordinate transformation $\phi: \ \Omega_C \to \Omega_P$ that transforms a uniform grid $\mathbf{x}$ (left) into a non-uniform grid $\mathbf{x}'$ (right), such that $\mathbf{x}'$ equidistributes the density ρ' ($\Omega_C = [0,1] \times [0,1]$, $\Omega_P = [0,1] \times [0,1]$).

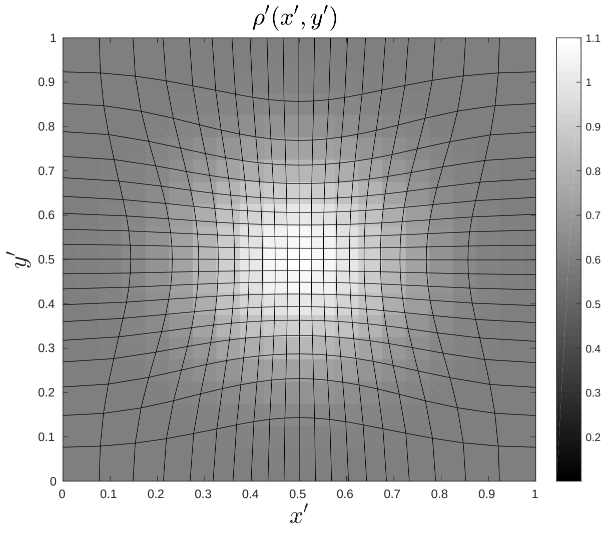


Figure 2.3: A sketch of the deformed grid $\mathbf{x}'(= \phi(\mathbf{x}) = (x', y'))$ that equidistributes ρ' on $\Omega_P = [0,1] \times [0,1]$; The mass $\rho'(x', y')dx'dy'$ in each cell $dx'dy'$ of the grid $\mathbf{x}'$ is uniform.

We note that the above grid generation problem is not well-posed for two or higher dimensions. More specifically, there exist infinitely many transformations that solve the problem. Figure 2.4 shows an example of the non-uniqueness of equidistribution grid in two-dimensional.

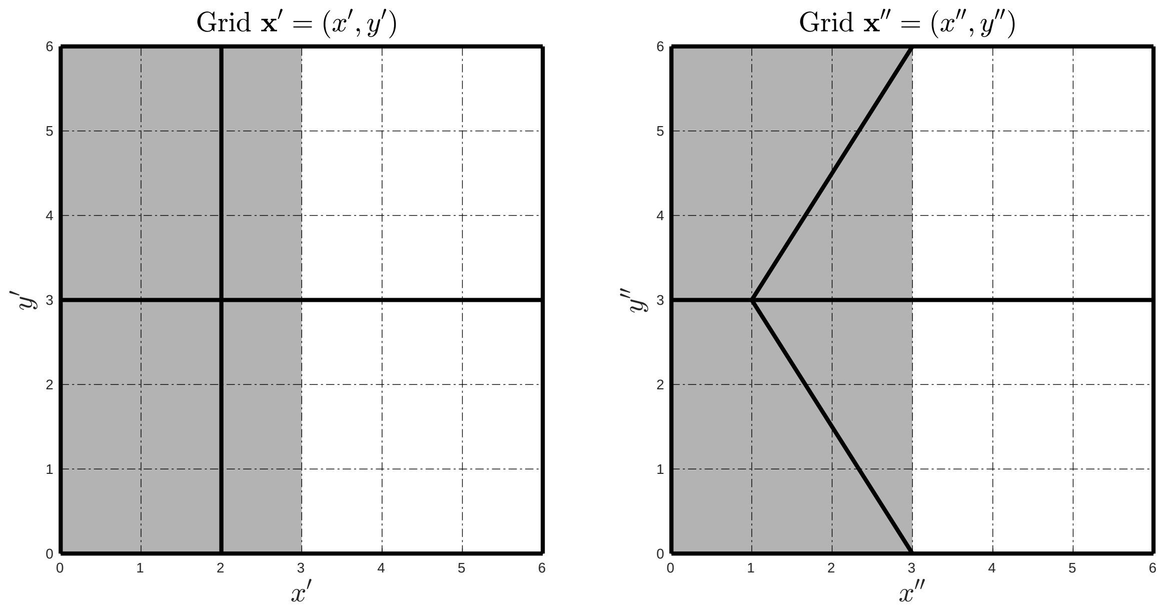

Grid $\mathbf{x}' = (x', y')$
Grid $\mathbf{x}'' = (x'', y'')$
x'
y'
x''
y''
0
1
2
3
4
5
6

Figure 2.4: Example of two different 2×2 equidistribution grid (solid lines) generated for the sample prescribed density ρ' (gray area: $\rho' = 3$; white area: $\rho' = 1$). The grid $\mathbf{x}'$ on the left hand side is closer to a uniform grid than the grid $\mathbf{x}''$ on the right hand side.

To make the problem well-posed, we aim at finding an optimal coordinate transformation $\phi^*(\mathbf{x})$ in a sense that it minimizes some measure of energy cost for transforming a uniform grid $\mathbf{x}$ into a deformed grid $\mathbf{x}'$ according to the equidistribution principle. For this purpose, we use the L_2-norm to measure the grid displacement between $\mathbf{x}$ and $\mathbf{x}'$. The optimal transformation $\phi^*(\mathbf{x})$ is then defined as

$$\phi^*(\mathbf{x}) \equiv \arg\min_{\phi(\mathbf{x})} \int_{\mathbf{x} \in \Omega_C} \|\mathbf{x} - \phi(\mathbf{x})\|_{L_2}^2 d^n\mathbf{x}. \tag{2.4}$$

That is, the optimal transformation $\phi^*(\mathbf{x})$ minimizes the total squared grid displacement between $\mathbf{x}$ and $\mathbf{x}'$. The optimal transformation exists uniquely amoung all the transformations that are constrained to the equidistribution principle (2.3). The minimization problem (2.4) constrained to (2.3) is called Monge-Kantorovich (MK) optimization. As we will see later, the optimization (2.4) not only helps to reduce mesh skewness but it also helps to prevent mesh tangling.

In this thesis, we specialize our discussion without loss of generality to the case that $\Omega_C = \Omega_P = \Omega$.

2.2 Monge-Ampère Equation

Knott and Smith [17] show that the optimal transformation $\phi^*(\mathbf{x})$ that minimize (2.4) is the gradient $\nabla u(\mathbf{x})$ of a convex function $u(\mathbf{x}) \in C(\Omega)$. Thus, to find the solution $\phi^*(\mathbf{x})$ to the grid generation problem, it suffices to solve

$$\det[D^2 u(\mathbf{x})] \frac{\rho'(\nabla u(\mathbf{x}))}{\rho(\mathbf{x})} = 1, \;\; u \text{ is convex} \tag{2.5}$$

for $u(\mathbf{x})$, where $D^2 u(\mathbf{x}) \in \mathbb{R}^{n \times n}$ is the Hessian matrix of $u(\mathbf{x})$ arising from (2.3). Let $f(u(\mathbf{x})) \equiv \frac{\rho(\mathbf{x})}{\rho'(\nabla u(\mathbf{x}))}$. Then (2.5) can be rewritten as

$$\det[D^2 u(\mathbf{x})] = f(u(\mathbf{x})), \;\; u \text{ is convex.} \tag{2.6}$$

Equation (2.6) is called Monge-Ampère (MA) equation.

We note that the grid generation method based on the MA equation (2.6) is robust to grid tangling (Budd et al [4], Weller et al [28]) due to the convex constraint of (2.6). Since $u(\mathbf{x})$ is convex, $\det[D^2 u(\mathbf{x})]$ is positive, or equivalently, the Jacobian determinant $\det[\mathcal{J}(\mathbf{x}')]$ is positive. It follows that the $\mathcal{J}(\mathbf{x}')$ is invertible (by the inverse function theorem) and so the generated grid $\mathbf{x}'$ will not tangle.

We now specify the boundary conditions of (2.6) in order to complete the grid generation model. Following Delzanno et al [11], we require the gradient $\nabla u(\mathbf{x})$, which corresponds to the optimal transformation $\phi^*(\mathbf{x})$, to satisfy

$$(\nabla u(\mathbf{x}) - \mathbf{x}) \cdot \mathbf{n} = 0 \text{ on } \partial\Omega, \tag{2.7}$$

where $\partial\Omega$ denotes the boundary of Ω, $\nabla u(\mathbf{x}) - \mathbf{x}$ is the displacement of a point on $\partial\Omega$, and $\mathbf{n}$ is the unit vector normal to $\partial\Omega$. That is, the displacement of a point is tangential to the boundary under the transformation ϕ. Equation (2.7) is called Neumann boundary condition.

2.3 Viscosity Solution

We note that the MA equation (2.6) is a fully nonlinear PDE, i.e., (2.6) is nonlinear in the highest-order derivatives of the prospective solution $u(\mathbf{x})$, since the left hand side of (2.6) consists of products of the second-order derivatives of $u(\mathbf{x})$. Accordingly, (2.6) may have multiple weak solutions, i.e., solutions that satisfy the equation almost everywhere. We are concerned with a particular type of weak solution, known as the viscosity solution.

The viscosity solution (Crandall et al [7], Crandall and Lions [8]) of a fully nonlinear PDE is a type of weak solution. The notion of viscosity solution is first introduced by Crandall and Lions [8] as a generalization of the classical concept of a solution to a PDE and it allows merely continuous functions to be solutions of fully nonlinear equations of second order (Crandall et al [7]). In our grid generation problem, we are interested in the viscosity solution of (2.6), since the viscosity solution of the MA equation is globally convex (Froese and Oberman [13]) while other weak solutions may not be convex. In addition, the existence and uniqueness of the viscosity solution of the MA equation (2.6) are guaranteed due to the constraint that u is convex and that Ω is bounded and convex (Crandall et al [7] and Gutiérrez,[14]), which makes the problem well-posed.

To give precise definition of the viscosity solution of (2.6), we rewrite the MA equation (2.6) as

$$\mathcal{F}(\mathbf{x}, D^2u(\mathbf{x})) \equiv -\det[D^2u(\mathbf{x})] + f(u(\mathbf{x})) = 0. \tag{2.8}$$

Definition 2.1. *A convex function $u \in C(\Omega)$ is a viscosity sub-solution (or super-solution) of the MA equation (2.8), if for all the test functions $v \in C^2(\Omega)$ and all $\mathbf{x} \in \Omega$, such that $u - v$ has a local minimum (or maximum) at $\mathbf{x}$, we have*

$$\mathcal{F}_*(\mathbf{x}, D^2v(\mathbf{x})) \leq 0 \ (\textit{or } \mathcal{F}^*(\mathbf{x}, D^2v(\mathbf{x})) \geq 0),$$

where $\mathcal{F}_*$ (or $\mathcal{F}^*$) is the lower (or upper) semi-continuous envelope of $\mathcal{F} : C \to \mathbb{R}$ on a closed set C, defined as:

$$\mathcal{F}_*(x) = \lim_{y\to x,} \inf_{y\in C} \mathcal{F}(y) \quad \left(\textit{or } \mathcal{F}^*(x) = \lim_{y\to x,} \sup_{y\in C} \mathcal{F}(y)\right).$$

Furthermore, u is a viscosity solution if it is both a viscosity sub-solution and super-solution.

According to Barles and Souganidis [1], it is desirable to employ a monotone scheme in the discretization of a fully nonlinear second order PDE in order to converge to the viscosity solution of the PDE. The definition of a monotone scheme (Barles and Souganidis [1]) is included as follows.

Definition 2.2. *A scheme in the form* $u_i^{n+1} = \sum_{k=-k_L}^{k_R} b_k u_{i+k}^n$ *(where* k_L *and* K_R *are two non-negative integers,* b_k *are the coefficients of the scheme, and* u_{i+k}^n *are data value at time level* n*) is said to be monotone, if all coefficients* b_k *are non-negative.*

2.4 Hamilton-Jacobi-Bellman Formulation

Our goal is to compute the viscosity solution of the MA equation (2.6). While we could attempt to solve (2.6) directly, we notice that it is difficult to design a numerical scheme for discretizing (2.6) due to the nonlinear operator $\det[D^2u]$ in it. Instead of solving the equation (2.6) directly, we convert (2.6) into an equivalent Hamilton-Jacobi-Bellman (HJB) equation (Chen and Wan [5, 6]). As we will see later, the differential operator of the equivalent HJB equation under fixed control parameters is linear, which makes the HJB equation amenable to numerical techniques. Hence, we apply the HJB formulation in the numerical computation of the MA equation (2.6).

The equivalence of MA equation and HJB equation is first established by Krylov [18] and Lions [20] for arbitrary dimensions. We include the equivalence as follows.

We adopt the notation that $A \succeq 0$ (or $A \preceq 0$) means that A is positive semi-definite (or negative semi-definite). We note that A is symmetric if $A \succeq 0$ (or $A \preceq 0$).

Lemma 2.1. *Let* $B \in \mathbb{R}^{n\times n}$ *be a symmetric matrix,* $n \geq 2$*, and let* $c \geq 0$*. Define the set*

$$S_1^+ \equiv \left\{A \in \mathbb{R}^{n\times n} : A \succeq 0,\ Tr[A] = 1\right\}. \tag{2.9}$$

Then

$$\max_{A\in S_1^+}\{Tr(AB) + c\sqrt[n]{\det[A]}\} = 0 \tag{2.10}$$

if and only if

$$B \preceq 0,\ n\sqrt[n]{\det(-B)} = c. \tag{2.11}$$

Proof. We refer readers to Lemma 2.6 of Smears [24] □

By applying Lemma 2.1, we reformulate the MA equation (2.6) into the following HJB equation (2.12):

Theorem 2.1. *Let $\Omega \subset \mathbb{R}^n$ be a convex set. Let $u \in C^2(\Omega)$ be a convex function, and $f \in C(\Omega)$ be a non-negative function. Then u solves the MA equation (2.6) if and only if it solves the following HJB equation*

$$\max_{A(\mathbf{x}) \in S_1^+} \left\{ -Tr\left[A(\mathbf{x}) D^2 u(\mathbf{x})\right] + n \sqrt[n]{\det[A(\mathbf{x})] f(u(\mathbf{x}))} \right\} = 0 \tag{2.12}$$

where S_1^+ is the set defined in (2.9) and $A(\mathbf{x}) \in S_1^+$ is the control at point $\mathbf{x} \in \Omega$.

Proof. Let $c = n\sqrt[n]{f}$, $B = -D^2 u$. Then Equation (2.10) becomes (2.12), and (2.11) gives $\det[D^2 u] = f$, which is the MA equation (2.6). By Lemma 2.1, u solves (2.6) if and only if it solves (2.12). $\square$

Chapter 3

Methodology

In the previous chapter, we provided background knowledge of our method for grid generation, including the HJB formulation (2.12) of the MA equation (2.6) that arises from the grid generation model. In this chapter, we present our contribution of solving the HJB equation (2.12) in two and three dimensions numerically using finite difference methods and constructing algorithms for grid generation. We start by introducing the parametrization of the HJB equation (2.12) in two and three dimensions. Then we describe the monotone schemes employed for discretizing the parametrized HJB equations and the resulting discrete systems. At last, we present the algorithms based on our numerical method for grid generation.

For simplicity, we specialize our discussion to the case that $\rho = 1$, $\Omega = [0,1] \times [0,1]$ for two-dimensional case and $\Omega = [0,1] \times [0,1] \times [0,1]$ for three-dimensional case.

3.1 Parametrization

Our objective is to solve the HJB equation (2.12) for $u(\mathbf{x})$. For this purpose, we need an explicit form of the matrix $A(\mathbf{x})$ in (2.12). Since $A(\mathbf{x}) \in S_1^+$ is positive semi-definite, it can be diagonalized by an orthogonal matrix. More specifically, $A(\mathbf{x})$ can be parametrized by a set of controls, which is shown as follows. We note that the following parametrization covers all the matrices contained in the set S_1^+, i.e., the parametrization is fully general.

3.1.1 2D Case

In two dimensions, following the work of Chen and Wan [5, 6], we parametrize $A(\mathbf{x})$ using a pair of controls $(a(\mathbf{x}), \theta(\mathbf{x}))$ at point $\mathbf{x}$ as

$$A(\mathbf{x}) = P(\theta(\mathbf{x}))^T D(a(\mathbf{x})) P(\theta(\mathbf{x})), \tag{3.1}$$

where

$$\begin{aligned} &P(\theta(\mathbf{x})) = \begin{pmatrix} \cos\theta(\mathbf{x}) & \sin\theta(\mathbf{x}) \\ -\sin\theta(\mathbf{x}) & \cos\theta(\mathbf{x}) \end{pmatrix}, \; D(a(\mathbf{x})) = \begin{pmatrix} a(\mathbf{x}) & 0 \\ 0 & 1 - a(\mathbf{x}) \end{pmatrix} \text{ and} \\ &a(\mathbf{x}) \in [0, 1], \;\; \theta(\mathbf{x}) \in [-\pi, \pi). \end{aligned} \tag{3.2}$$

We note that the matrix $P(\theta(\mathbf{x}))$ in (3.2) is an orthogonal matrix. Let $\Gamma = [0, 1] \times [-\pi, \pi]$ denote the set of admissible controls from (3.2) and let $\mathcal{C}(\mathbf{x})$ denote the pair of controls at point $\mathbf{x}$, i.e., $\mathcal{C}(\mathbf{x}) \equiv (a(\mathbf{x}), \theta(\mathbf{x}))$. By (3.1) and (3.2) , the HJB equation (2.12) in two dimensions (i.e., $n = 2$) can be re-formulated as the following equation, which we attempt to solve:

$$\begin{aligned} \max_{(a(\mathbf{x}),\theta(\mathbf{x}))\in\Gamma} &\{-\alpha_{11}(\mathcal{C}(\mathbf{x}))u_{xx}(\mathbf{x}) - \alpha_{22}(\mathcal{C}(\mathbf{x}))u_{yy}(\mathbf{x}) \\ &-2\alpha_{12}(\mathcal{C}(\mathbf{x}))u_{xy}(\mathbf{x}) + 2\sqrt{a(\mathbf{x})(1 - a(\mathbf{x}))f(u(\mathbf{x}))}\Big\} = 0, \end{aligned} \tag{3.3}$$

where $\alpha_{11}(\mathcal{C}(\mathbf{x})),\ \alpha_{12}(\mathcal{C}(\mathbf{x})),\ \alpha_{22}(\mathcal{C}(\mathbf{x}))$ are coefficients of the second order derivatives $u_{xx}(\mathbf{x}),\ u_{xy}(\mathbf{x}),\ u_{yy}(\mathbf{x})$ of $u(\mathbf{x})$, defined as:

$$\begin{aligned} \alpha_{11}(\mathcal{C}(\mathbf{x})) &= \frac{1}{2}[1 - (1 - 2a(\mathbf{x}))\cos 2\theta(\mathbf{x})], \\ \alpha_{22}(\mathcal{C}(\mathbf{x})) &= \frac{1}{2}[1 + (1 - 2a(\mathbf{x}))\cos 2\theta(\mathbf{x})], \\ \alpha_{12}(\mathcal{C}(\mathbf{x})) &= \frac{1}{2}(1 - 2a(\mathbf{x}))\sin 2\theta(\mathbf{x}). \end{aligned}$$

The differential operator of the HJB equation (3.3) is

$$\mathcal{D}_{\mathcal{C},u} \equiv -\alpha_{11}(\mathcal{C})u_{xx} - \alpha_{22}(\mathcal{C})u_{yy} - 2\alpha_{12}(\mathcal{C})u_{xy} + 2\sqrt{a(1-a)f},$$

which is linear under fixed controls $\mathcal{C}$. Thus, comparing with the MA equation (2.6) that is non-linear, the HJB equation (3.3) is more amenable to numerical techniques.

3.1.2 3D Case

We generalize the technique of parametrizing $A(\mathbf{x})$ in two dimensions for the three dimensional case. We parametrize $A(\mathbf{x})$ in three dimensions using a set of controls $(a_1(\mathbf{x}), a_2(\mathbf{x}), \theta(\mathbf{x}), \varphi(\mathbf{x}), \psi(\mathbf{x}))$ at point $\mathbf{x}$ as follows:

$$A(\mathbf{x}) = P(\theta(\mathbf{x}), \varphi(\mathbf{x}), \psi(\mathbf{x}))^T D(a_1(\mathbf{x}), a_2(\mathbf{x})) P(\theta(\mathbf{x}), \varphi(\mathbf{x}), \psi(\mathbf{x})), \tag{3.4}$$

where

$$\begin{aligned} P(\theta,\varphi,\psi) &= \begin{pmatrix} \cos\psi\cos\varphi & -\sin\psi\cos\theta + \cos\psi\sin\varphi\sin\theta & \sin\psi\sin\theta + \cos\psi\sin\varphi\cos\theta \\ \sin\psi\cos\varphi & \cos\psi\cos\theta + \sin\psi\sin\varphi\sin\theta & -\cos\psi\sin\theta + \sin\psi\sin\varphi\cos\theta \\ -\sin\varphi & \cos\varphi\sin\theta & \cos\varphi\cos\theta \end{pmatrix} \\ D(a_1,a_2) &= \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & 1-a_1-a_2 \end{pmatrix} \text{ and} \\ & a_1 \in [0,1],\ a_2 \in [0, 1-a_1],\ \theta \in [-\pi,\pi),\ \varphi \in [-\pi,\pi),\ \psi \in [-\pi,\pi). \end{aligned} \tag{3.5}$$

We note that the matrix $P(\theta(\mathbf{x}), \varphi(\mathbf{x}), \psi(\mathbf{x}))$ in (3.5) is an orthogonal matrix. Let $\mathcal{C}(\mathbf{x}) \equiv (a_1(\mathbf{x}), a_2(\mathbf{x}), \theta(\mathbf{x}), \phi(\mathbf{x}), \psi(\mathbf{x}))$ denote the set of controls at $\mathbf{x}$ and let Γ denote the five-dimensional set of admissible controls from (3.5). By (3.4) and (3.5), the HJB equation (2.12) in three dimensions (i.e., $n = 3$) can be rewritten as the following equation, which we attempt to solve:

$$\begin{aligned} \max_{\mathcal{C}(\mathbf{x})\in\Gamma} \{ & -\alpha_{11}(\mathcal{C}(\mathbf{x}))u_{xx}(\mathbf{x}) - \alpha_{22}(\mathcal{C}(\mathbf{x}))u_{yy}(\mathbf{x}) - \alpha_{33}(\mathcal{C}(\mathbf{x}))u_{zz}(\mathbf{x}) \\ & -2\alpha_{12}(\mathcal{C}(\mathbf{x}))u_{xy}(\mathbf{x}) - 2\alpha_{13}(\mathcal{C}(\mathbf{x}))u_{xz}(\mathbf{x}) - 2\alpha_{23}(\mathcal{C}(\mathbf{x}))u_{yz}(\mathbf{x}) \\ & + 3\sqrt[3]{a_1(\mathbf{x})a_2(\mathbf{x})(1-a_1(\mathbf{x})-a_2(\mathbf{x}))f(u(\mathbf{x}))} \qquad \} = 0, \end{aligned} \tag{3.6}$$

where $u_{xx}(\mathbf{x})$, $u_{yy}(\mathbf{x})$, $u_{zz}(\mathbf{x})$, $u_{xy}(\mathbf{x})$, $u_{xz}(\mathbf{x})$, $u_{yz}(\mathbf{x})$ are the second order derivatives of $u(\mathbf{x})$, and

$$\begin{aligned} \alpha_{11}(\mathcal{C}(\mathbf{x})) &= (\cos^2\psi\cos^2\varphi)a_1 + (\sin^2\psi\cos^2\varphi)a_2 + \sin^2\varphi(1-a_1-a_2), \\ \alpha_{22}(\mathcal{C}(\mathbf{x})) &= (-sin\psi\cos\theta + \cos\psi\sin\varphi\sin\theta)^2 a_1 \\ &\quad + (\cos\psi\cos\theta + \sin\psi\sin\varphi\sin\theta)^2 a_2 \\ &\quad + \cos^2\varphi\sin^2\theta(1-a_1-a_2), \\ \alpha_{33}(\mathcal{C}(\mathbf{x})) &= (\sin\psi\sin\theta + \cos\psi\sin\varphi\cos\theta)^2 a_1 \\ &\quad + (-\cos\psi\sin\theta + \sin\psi\sin\varphi\cos\theta)^2 a_2 \\ &\quad + \cos^2\varphi\cos^2\theta(1-a_1-a_2), \end{aligned}$$

$$\begin{aligned}\alpha_{12}(\mathcal{C}(\mathbf{x})) = &\cos\psi\cos\varphi(-\sin\psi\cos\theta+\cos\psi\sin\varphi\sin\theta)a_1\\ &+\sin\psi\cos\theta(\cos\psi\cos\theta+\sin\psi\sin\varphi\sin\theta)a_2\\ &-\sin\varphi\cos\varphi\sin\theta(1-a_1-a_2),\\ \alpha_{13}(\mathcal{C}(\mathbf{x})) = &\cos\psi\cos\varphi(\sin\psi\sin\theta+\cos\psi\sin\varphi\cos\theta)a_1\\ &+\sin\psi\cos\varphi(-\cos\psi\sin\theta+\sin\psi\sin\varphi\cos\theta)a_2\\ &-\sin\varphi\cos\varphi\cos\theta(1-a_1-a_2),\\ \alpha_{23}(\mathcal{C}(\mathbf{x})) = &(-\sin\psi\cos\theta+\cos\psi\sin\varphi\sin\theta)(\sin\psi\sin\theta+\cos\psi\sin\varphi\cos\theta)a_1\\ &+(\cos\psi\cos\theta+\sin\psi\sin\varphi\sin\theta)(-\cos\psi\sin\theta+\sin\psi\sin\varphi\cos\theta)a_2\\ &+\cos^2\varphi\sin\theta\cos\theta(1-a_1-a_2).\end{aligned}$$

We note that the differential operator of the HJB equation (3.6) is

$$\begin{aligned}\mathcal{D}_{\mathcal{C},u} \equiv &-\alpha_{11}(\mathcal{C})u_{xx}-\alpha_{22}(\mathcal{C})u_{yy}-\alpha_{33}(\mathcal{C})u_{zz}-2\alpha_{12}(\mathcal{C})u_{xy}\\ &-2\alpha_{13}(\mathcal{C})u_{xz}-2\alpha_{23}(\mathcal{C})u_{yz}+3\sqrt[3]{a_1a_2(1-a_1-a_2)f},\end{aligned}$$

which is linear under fixed controls $\mathcal{C}$.

3.2 Finite Difference Discretization

We now discretize the re-formulated HJB equations (3.3) and (3.6) using finite difference methods. We intend to use monotone schemes in order to converge to the viscosity solutions, as we mentioned earlier in Section 2.3.

For brevity, we use the following notation:

$$\begin{aligned}&U_{i,j}\ (\text{or } U_{i,j,k}) \equiv \text{ discrete approximation of } u(\mathbf{x}_{i,j})\ (\text{or } u(\mathbf{x}_{i,j,k})),\\ &f_{i,j}\ (\text{or } f_{i,j,k}) \equiv f(u(\mathbf{x}_{i,j}))\ (\text{or } f(u(\mathbf{x}_{i,j,k}))),\\ &a_{i,j} \equiv a(\mathbf{x}_{i,j}),\\ &\mathcal{C}_{i,j} \equiv \text{ the controls } (a(\mathbf{x}_{i,j}),\theta(\mathbf{x}_{i,j})),\\ &\mathcal{C}_{i,j,k} \equiv \text{ the controls } (a_1(\mathbf{x}_{i,j,k}),a_2(\mathbf{x}_{i,j,k}),\theta(\mathbf{x}_{i,j,k}),\varphi(\mathbf{x}_{i,j,k}),\psi(\mathbf{x}_{i,j,k})),\\ &(\alpha_{\text{ind}})_{i,j} \equiv \alpha_{\text{ind}}(\mathcal{C}_{i,j}),\ \text{ind} \in \{11,12,22\}.\\ &(\alpha_{\text{ind}})_{i,j,k} \equiv \alpha_{\text{ind}}(\mathcal{C}_{i,j,k}),\ \text{ind} \in \{11,22,33,12,13,23\}.\end{aligned} \tag{3.7}$$

3.2.1 2D Case

In two dimensions, we consider an $N \times N$ square grid

$$\{\mathbf{x}_{i,j} \equiv (x_i, y_j) | \ \mathbf{x}_{i,j} \in \Omega \text{ for } 1 \leq i, j \leq N; \ \mathbf{x}_{i,j} \in \partial\Omega \text{ for } i, j = 0 \text{ or } N+1\}$$

with grid size $h = \frac{1}{N}$. We approximate $u_{xx}(\mathbf{x}_{i,j})$ and $u_{yy}(\mathbf{x}_{i,j})$ in (3.3) using central finite differences:

$$u_{xx}(\mathbf{x}_{i,j}) \approx (U_{xx})_{i,j} \equiv \frac{1}{h^2}(U_{i+1,j} - 2U_{i,j} + U_{i-1,j}), \tag{3.8}$$

$$u_{yy}(\mathbf{x}_{i,j}) \approx (U_{yy})_{i,j} \equiv \frac{1}{h^2}(U_{i,j+1} - 2U_{i,j} + U_{i,j-1}). \tag{3.9}$$

For $u_{xy}(\mathbf{x}_{i,j})$ in (3.3), we approximate it using a standard 7-point stencil discretization, which leads to a monotone scheme in the following two cases (Chen and Wan [5, 6]):

(i) if

$$(\alpha_{11})_{i,j} \geq |(\alpha_{12})_{i,j}|, \ (\alpha_{22})_{i,j} \geq |(\alpha_{12})_{i,j}|, \text{ and } (\alpha_{12})_{i,j} \geq 0, \tag{3.10}$$

then $u_{xy}(\mathbf{x}_{i,j}) \approx (U_{xy})^{(1)}_{i,j}$, where

$$(U_{xy})^{(1)}_{i,j} \equiv \frac{2U_{i,j} + U_{i+1,j+1} + U_{i-1,j-1} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1}}{2h^2}. \tag{3.11}$$

(ii) if

$$(\alpha_{11})_{i,j} \geq |(\alpha_{12})_{i,j}|, \ (\alpha_{22})_{i,j} \geq |(\alpha_{12})_{i,j}|, \text{ and } (\alpha_{12})_{i,j} < 0, \tag{3.12}$$

then $u_{xy}(\mathbf{x}_{i,j}) \approx (U_{xy})^{(2)}_{i,j}$, where

$$(U_{xy})^{(2)}_{i,j} \equiv \frac{-2U_{i,j} - U_{i+1,j-1} - U_{i-1,j+1} + U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}}{2h^2}. \tag{3.13}$$

If either Condition (3.10) or Condition (3.12) is satisfied at $\mathbf{x}_{i,j}$, then the discretization of the HJB equation (3.3) at $\mathbf{x}_{i,j}$ can be written as

$$\max_{\mathcal{C}_{i,j} \in \Gamma} \left\{ -(\alpha_{11})_{i,j}(U_{xx})_{i,j} - 2(\alpha_{12})_{i,j}(U_{xy})^{(\text{idx})}_{i,j} - (\alpha_{22})_{i,j}(U_{yy})_{i,j} + 2\sqrt{a_{i,j}(1 - a_{i,j})f_{i,j}} \right\} = 0, \tag{3.14}$$

where

$$U_{xy}^{(\mathrm{idx})} = \begin{cases} U_{xy}^{(1)} \text{ in } (3.11) \ , \text{ if Condition } (3.10) \text{ is satisfied,} \\ U_{xy}^{(2)} \text{ in } (3.13) \ , \text{ if Condition } (3.12) \text{ is satisfied.} \end{cases}$$

We note that Conditions (3.10) and (3.12) are determined by the coefficients $(\alpha_{11})_{i,j}$, $(\alpha_{12})_{i,j}$ and $(\alpha_{22})_{i,j}$, which depend on the controls $\mathcal{C}_{i,j}$. The computation of $\mathcal{C}_{i,j}$ depends on $f_{i,j}$, which further depends on the values of the prescribed density functions ρ and ρ'. According to our numerical experienments, if ρ and ρ' are smooth functions, i.e, ρ and ρ' have continuous derivatives up to first order over their domains, then either Condition (3.10) or Condition (3.12) is satisfied at each point $\mathbf{x}_{i,j}$.

3.2.2 3D Case

In three dimensions, we consider an $N \times N \times N$ cubic grid

$$\{\mathbf{x}_{i,j,k} \equiv (x_i, y_j, z_k) \mid \mathbf{x}_{i,j,k} \in \Omega \text{ for } 1 \le i,j,k \le N; \ \mathbf{x}_{i,j,k} \in \partial\Omega \text{ for } i,j,k = 0 \text{ or } N+1\}$$

with grid size $h = \frac{1}{N}$. Similar to the two-dimensional case, we approximate $u_{xx}(\mathbf{x}_{i,j,k})$, $u_{yy}(\mathbf{x}_{i,j,k})$ and $u_{zz}(\mathbf{x}_{i,j,k})$ in (3.6) using central finite differences:

$$u_{xx}(\mathbf{x}_{i,j,k}) \approx (U_{xx})_{i,j,k} \equiv \frac{1}{h^2}(U_{i+1,j,k} - 2U_{i,j,k} + U_{i-1,j,k}) \tag{3.15}$$

$$u_{yy}(\mathbf{x}_{i,j,k}) \approx (U_{yy})_{i,j,k} \equiv \frac{1}{h^2}(U_{i,j+1,k} - 2U_{i,j,k} + U_{i,j-1,k}) \tag{3.16}$$

$$u_{zz}(\mathbf{x}_{i,j,k}) \approx (U_{zz})_{i,j,k} \equiv \frac{1}{h^2}(U_{i,j,k+1} - 2U_{i,j,k} + U_{i,j,k-1}). \tag{3.17}$$

For $u_{xy}(\mathbf{x}_{i,j,k})$, $u_{xz}(\mathbf{x}_{i,j,k})$ and $u_{yz}(\mathbf{x}_{i,j,k})$ in (3.6), we approximate them using standard 7-point stencil discretization. For $u_{xy}(\mathbf{x}_{i,j,k})$, we consider the two approximations $(U_{xy})_{i,j,k}^{(1)}$ and $(U_{xy})_{i,j,k}^{(2)}$ that are derived from the Taylor expansions of $U_{i+1,j+1,k}$, $U_{i-1,j-1,k}$, $U_{i+1,j-1,k}$ and $U_{i-1,j+1,k}$:

$$(U_{xy})_{i,j,k}^{(1)} = \frac{2U_{i,j,k} + U_{i+1,j+1,k} + U_{i-1,j-1,k} - U_{i+1,j,k} - U_{i-1,j,k} - U_{i,j+1,k} - U_{i,j-1,k}}{2h^2}, \tag{3.18}$$

$$(U_{xy})_{i,j,k}^{(2)} = \frac{-2U_{i,j,k} - U_{i+1,j-1,k} - U_{i-1,j+1,k} + U_{i+1,j,k} + U_{i-1,j,k} + U_{i,j+1,k} + U_{i,j-1,k}}{2h^2}. \tag{3.19}$$

Similarly, for $u_{xz}(\mathbf{x}_{i,j,k})$ and $u_{yz}(\mathbf{x}_{i,j,k})$, we consider the following approximations:

$$(U_{xz})^{(1)}_{i,j,k} = \frac{2U_{i,j,k} + U_{i+1,j,k+1} + U_{i-1,j,k-1} - U_{i+1,j,k} - U_{i-1,j,k} - U_{i,j,k+1} - U_{i,j,k-1}}{2h^2}, \tag{3.20}$$

$$(U_{xz})^{(2)}_{i,j,k} = \frac{-2U_{i,j,k} - U_{i+1,j,k-1} - U_{i-1,j,k+1} + U_{i+1,j,k} + U_{i-1,j,k} + U_{i,j,k+1} + U_{i,j,k-1}}{2h^2}, \tag{3.21}$$

$$(U_{yz})^{(1)}_{i,j,k} = \frac{2U_{i,j,k} + U_{i,j+1,k+1} + U_{i,j-1,k-1} - U_{i,j+1,k} - U_{i,j-1,k} - U_{i,j,k+1} - U_{i,j,k-1}}{2h^2}, \tag{3.22}$$

$$(U_{yz})^{(2)}_{i,j,k} = \frac{-2U_{i,j,k} - U_{i,j+1,k-1} - U_{i,j-1,k+1} + U_{i,j+1,k} + U_{i,j-1,k} + U_{i,j,k+1} + U_{i,j,k-1}}{2h^2}. \tag{3.23}$$

Theorem 3.1. *The discretization (3.15) – (3.23) of the second-order derivatives of $u(\mathbf{x}_{i,j,k})$ can lead to a monotone scheme for discretizing the HJB equation (3.6) in the following eight cases:*

(i) if

$$\begin{aligned}&(\alpha_{12})_{i,j,k} \geq 0,\ (\alpha_{13})_{i,j,k} \geq 0,\ (\alpha_{23})_{i,j,k} \geq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,\\ &(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\end{aligned} \tag{3.24}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}$.

(ii) if

$$\begin{aligned}&(\alpha_{12})_{i,j,k} \leq 0,\ (\alpha_{13})_{i,j,k} \geq 0,\ (\alpha_{23})_{i,j,k} \geq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,\\ &(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\end{aligned} \tag{3.25}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}$.

(iii) if

$$\begin{aligned}&(\alpha_{12})_{i,j,k} \geq 0,\ (\alpha_{13})_{i,j,k} \leq 0,\ (\alpha_{23})_{i,j,k} \geq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,\\ &(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\end{aligned} \tag{3.26}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}$.

(iv) if

$$\begin{aligned}&(\alpha_{12})_{i,j,k} \leq 0,\ (\alpha_{13})_{i,j,k} \leq 0,\ (\alpha_{23})_{i,j,k} \geq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,\\ &(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\end{aligned} \tag{3.27}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}$.

(v) if

$$(\alpha_{12})_{i,j,k} \geq 0,\ (\alpha_{13})_{i,j,k} \geq 0,\ (\alpha_{23})_{i,j,k} \leq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,$$
$$(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|, \tag{3.28}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$

(vi) if

$$(\alpha_{12})_{i,j,k} \leq 0,\ (\alpha_{13})_{i,j,k} \geq 0,\ (\alpha_{23})_{i,j,k} \leq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,$$
$$(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|, \tag{3.29}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$

(vii) if

$$(\alpha_{12})_{i,j,k} \geq 0,\ (\alpha_{13})_{i,j,k} \leq 0,\ (\alpha_{23})_{i,j,k} \leq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,$$
$$(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|, \tag{3.30}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$

(viii) if

$$(\alpha_{12})_{i,j,k} \leq 0,\ (\alpha_{13})_{i,j,k} \leq 0,\ (\alpha_{23})_{i,j,k} \leq 0,\ (\alpha_{11})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{13})_{i,j,k}|,$$
$$(\alpha_{22})_{i,j,k} \geq |(\alpha_{12})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|,\ (\alpha_{33})_{i,j,k} \geq |(\alpha_{13})_{i,j,k}| + |(\alpha_{23})_{i,j,k}|, \tag{3.31}$$

then $u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$

Proof. There are eight possible numerical schemes for discretizing the HJB equation (3.6) at point $\mathbf{x}_{i,j,k}$, based on the approximations (3.15) – (3.23), since each of the derivatives $u_{xy}(\mathbf{x}_{i,j,k})$, $u_{xz}(\mathbf{x}_{i,j,k})$ and $u_{yz}(\mathbf{x}_{i,j,k})$ has two forms of discretization. Let $C_{U_{i,j,k}}$ denote the coefficient of $U_{i,j,k}$ in the discretized HJB equation obtained by using one of the eight schemes. By Definition 2.2, the employed scheme is monotone if $C_{U_{i,j,k}} \geq 0$ and $C_{U_{i',j',k'}} \leq 0$ for all $(i',j',k') \neq (i,j,k)$, which leads to Conditions (3.24) – (3.31) as follows.

(i) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}+(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j+1,k}} = C_{U_{i-1,j-1,k}} = \frac{-(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k+1}} = C_{U_{i-1,j,k-1}} = \frac{-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k+1}} = C_{U_{i,j-1,k-1}} = \frac{-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \end{cases}$$

which is fulfilled if Condition (3.24) is satisfied.

(ii) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}+(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j-1,k}} = C_{U_{i-1,j+1,k}} = \frac{(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k+1}} = C_{U_{i-1,j,k-1}} = \frac{-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k+1}} = C_{U_{i,j-1,k-1}} = \frac{-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \end{cases}$$

which is fulfilled if Condition (3.25) is satisfied.

(iii) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}-(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j+1,k}} = C_{U_{i-1,j-1,k}} = \frac{-(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k-1}} = C_{U_{i-1,j,k+1}} = \frac{(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k+1}} = C_{U_{i,j-1,k-1}} = \frac{-(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.26) is satisfied.

(iv) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k}, \;\; u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k}, \;\; u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(1)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}-(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j-1,k}} = C_{U_{i-1,j+1,k}} = \frac{(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k-1}} = C_{U_{i-1,j,k+1}} = \frac{(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k+1}} = C_{U_{i,j-1,k-1}} = \frac{-(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.27) is satisfied.

(v) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k}, \;\; u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k}, \;\; u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}+(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j+1,k}} = C_{U_{i-1,j-1,k}} = \frac{-(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k+1}} = C_{U_{i-1,j,k-1}} = \frac{-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k-1}} = C_{U_{i,j-1,k+1}} = \frac{(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.28) is satisfied.

(vi) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(1)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}+(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j-1,k}} = C_{U_{i-1,j+1,k}} = \frac{(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i+1,j,k+1}} = C_{U_{i-1,j,k-1}} = \frac{-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k-1}} = C_{U_{i,j-1,k+1}} = \frac{(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.29) is satisfied.

(vii) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(1)}_{i,j,k},\ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k},\ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}-(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}+(\alpha_{12})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}-(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{-(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.30) is satisfied.

(viii) Suppose the discretization scheme is constructed using

$$u_{xy}(\mathbf{x}_{i,j,k}) \approx (U_{xy})^{(2)}_{i,j,k}, \ u_{xz}(\mathbf{x}_{i,j,k}) \approx (U_{xz})^{(2)}_{i,j,k}, \ u_{yz}(\mathbf{x}_{i,j,k}) \approx (U_{yz})^{(2)}_{i,j,k}.$$

Then the scheme is monotone if

$$\begin{cases} C_{U_{i,j,k}} = \frac{2((\alpha_{11})_{i,j,k}+(\alpha_{22})_{i,j,k}+(\alpha_{33})_{i,j,k}+(\alpha_{12})_{i,j,k}+(\alpha_{13})_{i,j,k}+(\alpha_{23})_{i,j,k})}{h^2} \geq 0, \\ C_{U_{i+1,j,k}} = C_{U_{i-1,j,k}} = \frac{-(\alpha_{11})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j+1,k}} = C_{U_{i,j-1,k}} = \frac{-(\alpha_{22})_{i,j,k}-(\alpha_{12})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k+1}} = C_{U_{i,j,k-1}} = \frac{-(\alpha_{33})_{i,j,k}-(\alpha_{13})_{i,j,k}-(\alpha_{23})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{(\alpha_{12})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{(\alpha_{13})_{i,j,k}}{h^2} \leq 0, \\ C_{U_{i,j,k}} = C_{U_{i,j,k}} = \frac{(\alpha_{23})_{i,j,k}}{h^2} \leq 0 \end{cases}$$

which is fulfilled if Condition (3.31) is satisfied.

□

According to our numerical experiments, if the prescribed density functions ρ and ρ' are smooth functions, then at each point $\mathbf{x}_{i,j,k}$, one of the conditions (3.24) – (3.31) is satisfied. If so, the discretization of the HJB equation (3.6) at $\mathbf{x}_{i,j,k}$ can be represented as

$$\begin{aligned} \max_{\mathcal{C}_{i,j,k}\in\Gamma} \Big\{ & -(\alpha_{11})_{i,j,k}(U_{xx})_{i,j,k} - (\alpha_{22})_{i,j,k}(U_{yy})_{i,j,k} - (\alpha_{33})_{i,j,k}(U_{zz})_{i,j,k} \\ & - 2(\alpha_{12})_{i,j,k}(U^{(\text{idx})}_{xy})_{i,j,k} - 2(\alpha_{13})_{i,j,k}(U^{(\text{idx})}_{xz})_{i,j,k} - 2(\alpha_{23})_{i,j,k}(U^{(\text{idx})}_{yz})_{i,j,k} \\ & + 3\sqrt[3]{a_1(\mathbf{x}_{i,j,k})a_2(\mathbf{x}_{i,j,k})(1-a_1(\mathbf{x}_{i,j,k})-a_2(\mathbf{x}_{i,j,k}))f_{i,j,k}} \Big\} = 0, \end{aligned} \tag{3.32}$$

where idx $\in \{1,2\}$ and $U^{(\text{idx})}_{xy}$, $U^{(\text{idx})}_{xz}$ and $U^{(\text{idx})}_{yz}$ are given by (3.18) – (3.23).

3.3 The Discrete Systems

In the previous section, we discretized the HJB equation (3.3) in two dimensions and the HJB equation (3.6) in three dimensions using finite difference methods.

The discretization of (3.3) (on an $N \times N$ grid) leads to a discrete system of N^2 equations; each of the discrete equations corresponds to a grid point $\mathbf{x}_{i,j}$ and is given by the discretization (3.14). The discretization of (3.3) (on an $N \times N \times N$ grid) generates a discrete system of N^3 equations; each of the discrete equations corresponds to a grid point $\mathbf{x}_{i,j,k}$ and is given by the discretization (3.32).

We show that the two discrete systems arising from (3.14) and (3.32) can be written into a general matrix form as follows.

- In two dimensions, we let

$$\mathbf{U} \equiv \{U_{i,j} | 1 \leq i, j \leq N\} \in \mathbb{R}^{N^2 \times 1}$$

be a vector of the unknowns, and let $a \in \mathbb{R}^{N^2 \times 1}$ and $\theta \in \mathbb{R}^{N^2 \times 1}$ be vectors of the corresponding controls. Then (3.14) can be written into the matrix form:

$$\max_{(a,\theta)\in\Gamma} \{\mathbf{A}(a,\theta)\mathbf{U} - \mathbf{F}(a,\theta;\mathbf{U})\} = 0, \tag{3.33}$$

where $\mathbf{A}(a,\theta) \in \mathbb{R}^{N^2 \times N^2}$ is the matrix that consists of the coefficients of the unknowns $\{U_{i,j}\}$ in (3.14), $\mathbf{F}(a,\theta;\mathbf{U}) \in \mathbb{R}^{N^2 \times 1}$ is the vector that consists of the term $2\sqrt{a(1-a)f}$ in (3.14) and the boundary term that arises from the Neumann boundary condition (2.7). We note that $\mathbf{A}(a,\theta)$ depend on the controls (a,θ), while $\mathbf{F}(a,\theta;\mathbf{U})$ depends on (a,θ) and $\mathbf{U}$.

- In three dimensions, we let

$$\mathbf{U} \equiv \{U_{i,j,k} | 1 \leq i, j, k \leq N\} \in \mathbb{R}^{N^3 \times 1}$$

be a vector of the unknowns

$$\begin{aligned}\mathbf{U} = (&U_{1,1,1}, U_{2,1,1}, \cdots, U_{N,1,1}, \quad U_{1,2,1}, U_{2,2,1}, \cdots, U_{N,2,1}, \quad \cdots \quad U_{1,N,1}, U_{2,N,1}, \cdots, U_{N,N,1},\\ &U_{1,1,2}, U_{2,1,2}, \cdots, U_{N,1,2}, \quad U_{1,2,2}, U_{2,2,2}, \cdots, U_{N,2,2}, \quad \cdots \quad U_{1,N,2}, U_{2,N,2}, \cdots, U_{N,N,2},\\ &\cdots\\ &U_{1,1,N}, U_{2,1,N}, \cdots, U_{N,1,N}, \; U_{1,2,N}, U_{2,2,N}, \cdots, U_{N,2,N}, \; U_{1,N,N}, U_{2,N,N}, \cdots, U_{N,N,N})^T\end{aligned}$$

and let $a_1 \in \mathbb{R}^{N^3\times 1}, a_2 \in \mathbb{R}^{N^3\times 1}, \varphi \in \mathbb{R}^{N^3\times 1}, \theta \in \mathbb{R}^{N^3\times 1}, \psi \in \mathbb{R}^{N^3\times 1}$ be vectors of the corresponding controls. Then (3.32) can be written into the form

$$\max_{a_1,a_2,\theta,\varphi,\psi\in\Gamma}\{\mathbf{A}(a_1,a_2,\theta,\varphi,\psi)\mathbf{U} - \mathbf{F}(a_1,a_2,\theta,\varphi,\psi;\mathbf{U})\} = 0, \tag{3.34}$$

where $\mathbf{A}(a_1,a_2,\theta,\varphi,\psi) \in \mathbb{R}^{N^3\times N^3}$ is a matrix that assembles the coefficients of the unknowns $\{U_{i,j,k}\}$ in (3.32), $\mathbf{F}(a_1,a_2,\theta,\varphi,\psi;\mathbf{U}) \in \mathbb{R}^{N^3\times 1}$ is a vector consists of the term $3\sqrt[3]{a_1a_2(1-a_1-a_2)f}$ in (3.32) and the boundary term arising from the Neumann boundary condition (2.7).

Without ambiguity, we will use

$$\max_{\mathcal{C}\in\Gamma}\{\mathbf{A}(\mathcal{C})\mathbf{U} - \mathbf{F}(\mathcal{C};\mathbf{U})\} = 0 \tag{3.35}$$

to represent both the discrete systems (3.33) and (3.34) in the follow sections. Here, $\mathcal{C}$ represents the set of controls, i.e., $\mathcal{C} = (a,\theta)$ in two dimensions and $\mathcal{C} = (a_1,a_2,\theta,\varphi,\psi)$ in three dimensions. The explicit form of (3.35) is determined based on the context.

3.4 The Algorithm

In this section, we construct algorithms for solving the discretized system (3.35) numerically.

For brevity, we use the following notation:

$$\begin{aligned}
&\mathcal{C}^{(m)} \equiv \text{the optimal controls } \mathcal{C} \text{ at m-th iteration}\\
&\mathbf{U}^{(m)} \equiv \text{the numerical solution } \mathbf{U} \text{ at m-th iteration},\\
&\mathbf{A}^{(m)} \equiv \text{the matrix } \mathbf{A}(\mathcal{C}^{(m)}),\\
&\mathbf{F}^{(m)} \equiv \text{the vector } \mathbf{F}(\mathcal{C}^{(m)};\mathbf{U}^{(m)}),\\
&\mathbf{R}^{(m)} \equiv \text{the residual } \mathbf{A}^{(m)}\mathbf{U}^{(m)} - \mathbf{F}^{(m)},\\
&\mathbf{x}_t \equiv \text{ a grid point } \mathbf{x}_{i,j} \text{ (or } \mathbf{x}_{i,j,k}) \text{ in } \Omega,\\
&\mathcal{C}_t \equiv \text{ the set of controls at the point } \mathbf{x}_t,\\
&\mathcal{L}(\mathbf{x}_t;\mathcal{C};\mathbf{U}) \equiv \text{ the equation contained in } \{\mathbf{A}(\mathcal{C})\mathbf{U} - \mathbf{F}(\mathcal{C};\mathbf{U})\} \text{ that}\\
&\text{corresponds to the grid point } \mathbf{x}_t.
\end{aligned} \tag{3.36}$$

Our first attempt is to use Policy Iteration (Chen and Wan [6]), which is a fixed point iteration algorithm. The Policy Iteration algorithm iteratively performs the following two steps:

(i) compute the optimal controls $\mathcal{C}^{(m)} = \{\mathcal{C}_t^{(m)}\}_{\mathbf{x}_t \in \Omega}$ under a given solution $\mathbf{U}^{(m)}$, where $\mathcal{C}_t^{(m)}$ is the set of optimal controls at point $\mathbf{x}_t \in \Omega$:

$$\mathcal{C}_t^{(m)} \equiv \underset{\mathcal{C}_t \in \Gamma}{\arg\max}\, \mathcal{L}(\mathbf{x}_t; \mathcal{C}; \mathbf{U}^{(m)}) \tag{3.37}$$

(ii) solve the following linear system for $\mathbf{U}^{(m+1)}$ under a given set of controls $\mathcal{C}^{(m)}$:

$$\mathbf{A}(\mathcal{C}^{(m)})\mathbf{U}^{(m+1)} = \mathbf{F}(\mathcal{C}^{(m)}; \mathbf{U}^{(m)}). \tag{3.38}$$

However, according to our numerical experiments, the resulting matrix $\mathbf{A}(\mathcal{C}^{(m)})$ in (3.38) is singular and the right-hand side term $\mathbf{F}(\mathcal{C}^{(m)}; \mathbf{U}^{(m)})$ of (3.38) is not in the range of $\mathbf{A}(\mathcal{C}^{(m)})$, in which case the system (3.38) is inconsistent.

To approximate the numerical solution to (3.38), we replace Step (ii) of the Policy Iteration by solving the following system for $\mathbf{U}^{(m+1)}$:

$$(\lambda I + \mathbf{A}(\mathcal{C}^{(m)}))\mathbf{U}^{(m+1)} = \mathbf{F}(\mathcal{C}^{(m)}; \mathbf{U}^{(m)}) + \lambda I \mathbf{U}^{(m)} \tag{3.39}$$

where I is an identity matrix and λ (> 0) is a constant regularizer specified by the user. The idea is that the solution $\mathbf{u}$ to the system

$$\frac{d\mathbf{u}}{dt} + \mathbf{A}\mathbf{u} = \mathbf{F} \tag{3.40}$$

approaches to the solution $\hat{\mathbf{u}}$ to the system

$$\mathbf{A}\hat{\mathbf{u}} = \mathbf{F} \tag{3.41}$$

as the temporal variable $t \to \infty$. Let

$$dt \equiv \frac{1}{\lambda} I, \; d\mathbf{u} \equiv \mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}, \; \mathbf{u} \equiv \mathbf{U}^{(m+1)} \text{ and } \hat{\mathbf{u}} \equiv \mathbf{U}^{(m+1)}.$$

Then (3.40) becomes (3.39), and (3.41) becomes (3.38). We can see that $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ decreases faster as $\frac{1}{\lambda} \to 0$ (i.e., $\lambda \to \infty$) since λ is correlated to t.

We note that a small λ corresponds to a large step size in the iteration of the algorithm and leads to a fast convergence to an approximation of the numerical solution $\mathbf{U}^{(m+1)}$ to the system (3.38). However, according to our numerical experiments, if λ is too small, then the algorithm may not converge due to floating point errors. This gives arise to concerns on the computational efficiency of the algorithm, since reducing the number of iterations leads to a decrease in the likelihood for the algorithm to converge.

To improve computational efficiency, we use an adaptive regularizer instead of a constant regularizer in the algorithm. The adaptive regularizer, denoted by λ, changes at each iteration of the algorithm according to the following criteria: at the m-iteration,

- if $\|\mathbf{R}^{(m)}\| < \|\mathbf{R}^{(m-1)}\|$ (where $\mathbf{R}^{(m)}$ is the residual defined in (3.36)), i.e., $\|\mathbf{R}\|$ keeps decreasing, then we reduce λ by $\lambda = \lambda * \lambda_{\text{damp}}$, where $\lambda_{\text{damp}} \in [0, 1]$ is the damping of λ, so that the residual $\|\mathbf{R}\|$ decreases faster;
- if $\|\mathbf{R}^{(m)}\| \geq \|\mathbf{R}^{(m-1)}\|$, i.e., $\|\mathbf{R}\|$ stops decreasing, which implies that the λ selected for computing the solution $\mathbf{U}^{(m)}$ is too large, then we go back to the $(m-1)$th iteration and iteratively perform the following two steps at the $(m-1)$th iteration:
  (a) increase λ by $\lambda = \lambda * \lambda_{\text{amp}}$, where $\lambda_{\text{amp}} > 1$ is the amplification of λ,
  (b) re-calculate the solution $\mathbf{U}^{(m)}$ using the new λ and compute the corresponding residual $\mathbf{R}^{(m)}$,

  until $\|\mathbf{R}^{(m)}\| < \|\mathbf{R}^{(m-1)}\|$ or $\|\mathbf{U}^{(m)} - \mathbf{U}^{(m-1)}\| \leq$ tolerance.

We note that, while we increase the regularizer λ to make the algorithm converges, we may encounter floating point errors if λ is too large. To reduce numerical errors, we require the maximum value $\lambda_{\max}$ of λ to satisfy $\lambda_{\max} \leq \frac{1}{\sqrt{\epsilon}}$, where ϵ is machine epsilon. Meanwhile, we specify a minimum value $\lambda_{\min} (> 0)$ of λ in order to make the algorithm less likely to diverge.

In addition, the procedure of performing the two steps (a) and (b) to adjust the value of the regularizer at each iteration can be repeated up to $\log_{\lambda_{\text{amp}}} \left(\frac{\lambda_{\max}}{\lambda} \right)$ times, where λ is the current choice of regularizer. The iterative behavior of this procedure can be reduced by choosing a smaller safety factor $\lambda_{\max}$ or a larger amplification λ_{amp}.

The complete procedure of the algorithm is described in Algorithm 1.

Algorithm 1 (2D & 3D)

Input: ρ, ρ', λ, λ_{damp}, λ_{amp}, $\lambda_{\max}$, $\lambda_{\min}$.

Output: Numerical solution $\mathbf{U}$ and the corresponding optimal controls $\mathcal{C}^*$.

```
 1: Set $\mathbf{U}^{(0)} = \begin{cases} \frac{1}{2}(x^2+y^2) & \text{in 2D case} \\ \frac{1}{2}(x^2+y^2+z^2) & \text{in 3D case} \end{cases}$, where $(x,y)$ or $(x,y,z)$ is the coordinate
    system of a uniform grid in two or three dimensions.
 2: for $m = 0, 1, \ldots$ until convergence do
 3:     Compute the optimal controls $\mathcal{C}^{(m)} = \{C_t^{(m)}\}_{\mathbf{x}_t \in \Omega}$ under the current solution $\mathbf{U}^{(m)}$,
        where $C_t^{(m)}$ is given by (3.37). The computation can be done using brute-force or
        Newton's method.
 4:     Compute the residual $\mathbf{R}^{(m)}$ given by (3.36)
 5:     if $\|\mathbf{R}^{(m)}\| \leq$ tolerance then return
 6:     Solve the system (3.39) for $\mathbf{U}^{(m+1)}$
 7:     if $\|\mathbf{R}^{(m)} - \mathbf{R}^{(m-1)}\| \leq$ tolerance or $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\| \leq$ tolerance then return
 8:     if $\|\mathbf{R}^{(m)}\| < \|\mathbf{R}^{(m-1)}\|$ then $\lambda = \lambda * \lambda_{\text{damp}}$ unless $\lambda = \lambda_{\min}$.
 9:     else set $\lambda = \lambda * \lambda_{\text{amp}}$ and go back to Step 6.
10: end for
11: return $\mathbf{U} = \mathbf{U}^{(m)}$, $\mathcal{C}^* = \mathcal{C}^{(m)}$.
```

Chapter 4

Numerical Results

In this chapter, we present the numerical results of the algorithms discussed in Chapter 3. We start by explaining the assessment criteria and measures for diagnosing the quality of the generated grids. The results of the numerical experiments are discussed by analyzing equidistribution error, convergence of the algorithms and presenting the generated grids. The algorithms are tested with several non-trivial examples for robustness analysis.

4.1 Assessment Criteria and Measures

Since our objective is to find an optimal grid cooridinate transformation that minimizes the grid displacement (2.4) under the equidistribution constraint (2.3), we assess our numerical method by checking the global displacement of the generated grids and the enforcement of the equidistribution.

4.1.1 Global Displacement

Let $\mathbf{x}$ denote the coordinate system of the uniform cells and $\mathbf{x}'$ be the coordinate system of the generated cells. We measure the global displacement $\mathbf{d}$ of the grid points by

$$\mathbf{d} = \sum_{i,j,k} (\|\mathbf{x}' - \mathbf{x}\|_{L_2}^2)_{i,j,k} \Delta x \Delta y \Delta z \tag{4.1}$$

which is a first-order approximation of the Monge-Kantorovich optimization (2.4). We report the geometric mean $\mathbf{d}^{1/n}$ (where n represents the dimension of the space) of the global displacement $\mathbf{d}$ as a meansure of the deformation of the generated grid.

4.1.2 Equidistribution Diagnostic

To assess the enforcement of equidistribution, we define the discretized volumes of the generated cells in terms of the determinant of the Jacobian matrix of $\mathbf{x}'$ with respect to $\mathbf{x}$. Since the generated cells are deformed, we need to compute the Jacobian determinant numerically. We use $\mathbf{J}_{\text{num}}$ to denote the numerically computed Jacobian determinant for brevity.

In the 2D case, we employ a four-point approximation from Delzanno et al [11] to compute $\mathbf{J}_{\text{num}}$. Let (x'_i, y'_i), $i = 1, 2, 3, 4$ denote the four vertices of a given cell in the new grid coordinate system (see Figure 4.1).

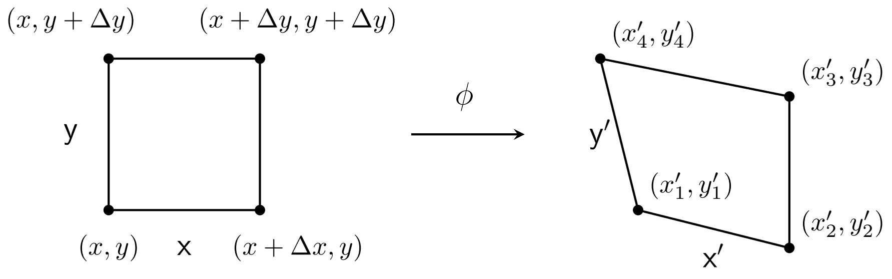


Figure 4.1: Sketch of the transformation of a two-dimensional cell.

The derivatives at each cell center are computed as

$$\begin{aligned}
\frac{\Delta x'}{\Delta x} &= \frac{x'_2 - x'_1 + x'_3 - x'_4}{2\Delta x},\\
\frac{\Delta x'}{\Delta y} &= \frac{x'_4 - x'_1 + x'_3 - x'_2}{2\Delta y},\\
\frac{\Delta y'}{\Delta x} &= \frac{y'_2 - y'_1 + y'_3 - y'_4}{2\Delta x},\\
\frac{\Delta y'}{\Delta y} &= \frac{y'_4 - y'_1 + y'_3 - y'_2}{2\Delta y}.
\end{aligned}$$

Then the numerical Jacobian determinant at the center of the cell is computed by

$$\mathbf{J}_{\text{num}} = \frac{\Delta x'}{\Delta x}\frac{\Delta y'}{\Delta y} - \frac{\Delta x'}{\Delta y}\frac{\Delta y'}{\Delta x}. \tag{4.2}$$

The discretized cell volume in two-dimensional space is $\mathbf{J}_{\text{num}}\Delta x \Delta y$. With this discretization, the area of the deformed cell corresponds to the natural geometrical area of the polygon connecting the four vertices of the cell (Delzanno et al [11]).

In 3D case, we derive an eight-point approximation from the above four-point approximation to compute $\mathbf{J}_{\text{num}}$. We denote the eight vertices of a given cell in the new grid coordinate system by (x_i', y_i', z_i'), $i = 1, 2, ..., 8$. See Figure 4.2.

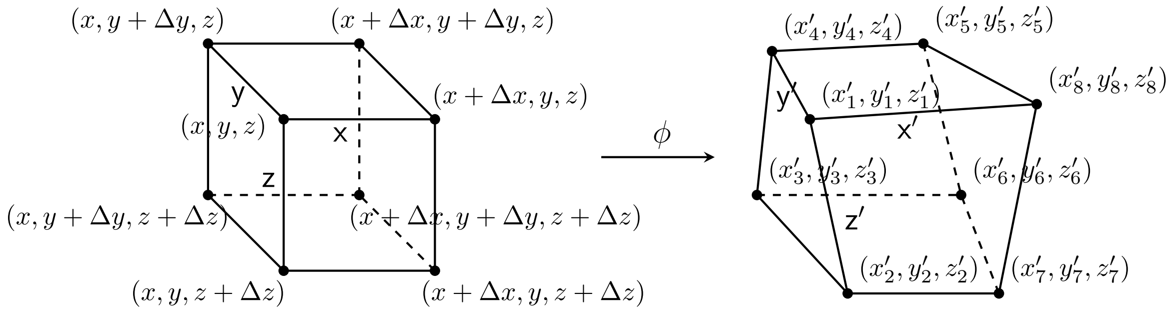


Figure 4.2: Sketch of the transformation of a three-dimensional cell.

We compute the derivatives at each cell center by

$$\begin{aligned}
\frac{\Delta x'}{\Delta x} &= \frac{(x_8' - x_1') + (x_5' - x_4') + (x_7' - x_2') + (x_6' - x_3')}{4\Delta x}, \\
\frac{\Delta y'}{\Delta x} &= \frac{(y_8' - y_1') + (y_5' - y_4') + (y_7' - y_2') + (y_6' - y_3')}{4\Delta x}, \\
\frac{\Delta z'}{\Delta x} &= \frac{(z_8' - z_1') + (z_5' - z_4') + (z_7' - z_2') + (z_6' - z_3')}{4\Delta x}, \\
\frac{\Delta x'}{\Delta y} &= \frac{(x_4' - x_1') + (x_5' - x_8') + (x_3' - x_2') + (x_6' - x_7')}{4\Delta y}, \\
\frac{\Delta y'}{\Delta y} &= \frac{(y_4' - y_1') + (y_5' - y_8') + (y_3' - y_2') + (y_6' - y_7')}{4\Delta y}, \\
\frac{\Delta z'}{\Delta y} &= \frac{(z_4' - z_1') + (z_5' - z_8') + (z_3' - z_2') + (z_6' - z_7')}{4\Delta y}, \\
\frac{\Delta x'}{\Delta z} &= \frac{(x_2' - x_1') + (x_3' - x_4') + (x_7' - x_8') + (x_6' - x_5')}{4\Delta z}, \\
\frac{\Delta y'}{\Delta z} &= \frac{(y_2' - y_1') + (y_3' - y_4') + (y_7' - y_8') + (y_6' - y_5')}{4\Delta z}, \\
\frac{\Delta z'}{\Delta z} &= \frac{(z_2' - z_1') + (z_3' - z_4') + (z_7' - z_8') + (z_6' - z_5')}{4\Delta z}
\end{aligned}$$

and the numerical Jacobian determinant at the center of the cell is computed as

$$\begin{aligned}\mathbf{J}_{\text{num}} = &\frac{\Delta x'}{\Delta x}\frac{\Delta y'}{\Delta y}\frac{\Delta z'}{\Delta z} - \frac{\Delta x'}{\Delta x}\frac{\Delta y'}{\Delta z}\frac{\Delta z'}{\Delta y} - \frac{\Delta x'}{\Delta y}\frac{\Delta y'}{\Delta x}\frac{\Delta z'}{\Delta z} + \frac{\Delta x'}{\Delta y}\frac{\Delta y'}{\Delta z}\frac{\Delta z'}{\Delta x} \\ &+ \frac{\Delta x'}{\Delta z}\frac{\Delta y'}{\Delta x}\frac{\Delta z'}{\Delta y} - \frac{\Delta x'}{\Delta z}\frac{\Delta y'}{\Delta y}\frac{\Delta z'}{\Delta x} \end{aligned} \tag{4.3}$$

and $\mathbf{J}_{\text{num}}\Delta x\Delta y\Delta z$ is the discretized cell volume in 3D.

According to the equidistribution principle (2.3), if the coordinate system $\mathbf{x}'$ of the generated cells strictly satisfies equidistribution constraint, then we have

$$\det[\mathcal{J}(\mathbf{x}')] = \frac{\rho(\mathbf{x})}{\rho'(\mathbf{x}')}, \tag{4.4}$$

where $\mathcal{J}(\mathbf{x}')$ is the Jacobian matrix $\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}$, ρ and ρ' are prescribed densities. We let $\mathbf{J}'$ denote the Jacobian determinant in (4.4) for brevity. In order to assess the enforcement of equidistribution, we compare the determinant of the numerical Jacobian of the generated coordinate system $\mathbf{x}'$ with the prescribed Jacobian determinant, i.e., we check if $\mathbf{J}' = \mathbf{J}_{\text{num}}$ at the cell centers. For this purpose, we define the equidistribution error of the generated cells as

$$\text{Error} = \left(\sum_{i,j,k} |\mathbf{J}_{\text{num}} - \mathbf{J}'|^2_{i,j,k}\Delta x\Delta y\Delta z\right)^{1/n}. \tag{4.5}$$

We note that the equidistribution error is an absolute error.

4.2 Experimental Results

In this section, we present the numerical results for grid generation using our algorithm.

We note that, since we are using adaptive regularization, each iteration may not have the same number of function evaluations, which can be seen later from the results of the following examples. At each iteration, the function can be re-evaluated up to $\log_{\lambda_{\text{amp}}}\left(\frac{\lambda_{\max}}{\lambda}\right)$ times in order to find a suitable regularizer, where λ is the current choice of regularizer, $\lambda_{\max}$ is the maximum value that λ can be assigned with, and λ_{amp} is the amplification of λ. In our examples, the worst case is to re-evaluate the function up to 5 times at each iteration. This worst case generally occurs on the final iteration.

Informally, the algorithm has three stages: initialization, stabilization and terminantion. Initialization is a cold-start that determines a suitable regularizer, stabilization is where $\lambda = \lambda_{\text{min}}$ or some other choice that does not frequently change and termination is where several function evaluations are made to verify convergence has occurred. The stage of initialization typically requires one or two function evaluations until a stable regularizer is chosen. Later iterations with $\lambda = \lambda_{\text{min}}$ require a single function evaluation. In the worst case, iterations between initialization and termination require two function evaluations. This occurs because $\lambda \neq \lambda_{\text{min}}$ and λ alternates between an acceptable value and an unacceptable value. Other heuristics such as delayed gratification (Transtrum and Sethna [27]), i.e. decrease the regularizer λ by a small fixed factor when the residual $\|\mathbf{R}\|$ keeps decreasing and increase λ by a large fixed factor when $\|\mathbf{R}\|$ stops decreasing, or choosing a smaller safety factor λ_{min} can be used to reduce this type of cycling behavior.

In each of the following examples, we assess the algorithm on grids of $N \times N$ points with $N = 10,\ 20,\ 40,\ 80$. We consider $\rho = 1$ and $\Omega = [0,1] \times [0,1]$ for all 2D examples and $\Omega = [0,1] \times [0,1] \times [0,1]$ for all 3D examples, as we mentioned earlier. All computations of the examples presented in this chapter are implemented in Matlab 2013b and tested on a computing machine with 132 GB of memory (CPU model: Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz).

2D case

Example 1. Consider the density function

$$\rho'(x', y') = \exp\left(-\frac{(x-0.5)^2 + (y-0.5)^2}{0.05}\right) + 0.1.$$

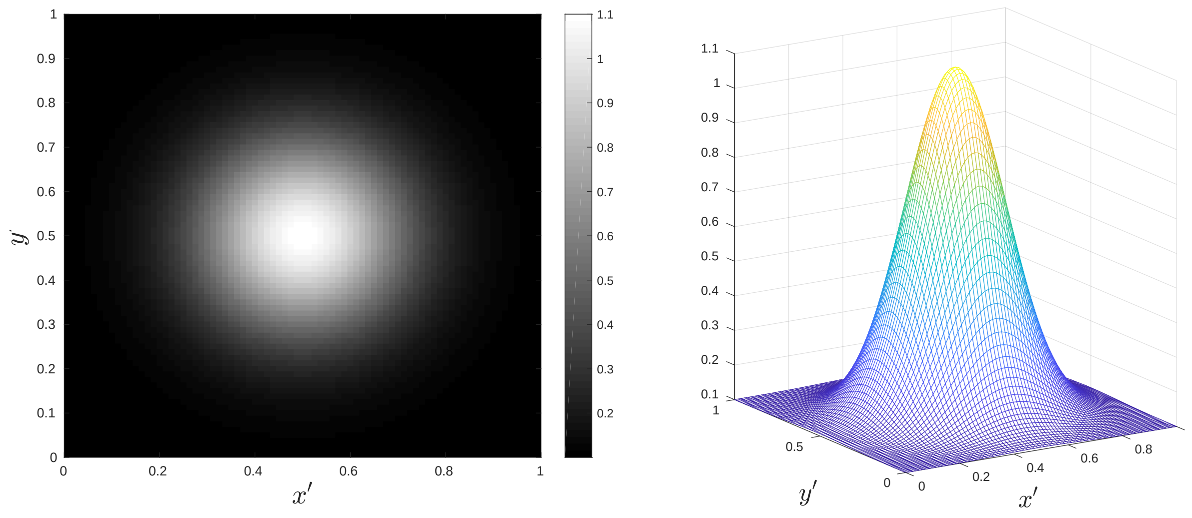


Figure 4.3: Example 1. Left: $\rho'(x', y')$. Right: A mesh plot of $\rho'(x', y')$ for illustrating the gradient of the densities.

This example corresponds to a diffusion centered at $(x', y') = (0.5, 0.5)$. Since a generated grid equidistributes the quantity given by $\rho'(x', y')$, that is, the volume in each cell of the generated grid is the same, the shape of the grid is expected to look like a contraction centered at $(x', y') = (0.5, 0.5)$.

The results of Example 1 are presented in Table 4.1. The equidistribution error (i.e. *Error* in Table 4.1) decreases as the grid becomes finer. Meanwhile, the geometric mean $\sqrt{\mathbf{d}}$ of the global displacement $\mathbf{d}$ defined in (4.1) increases as the number of cells in the generated grid increases. Intuitively, the displacement of a grid point (x', y') tends to be large when the rate of change in density at the point (x', y') is large. By looking at the mesh plot of $\rho'(x', y')$ in Figure 4.3, we notice that the norms of the gradient of the density at most of the grid points are large. This leads to the result that most of the cells in a generated grid are deformed when the grid is small. As the generated grid becomes finer, the total number of deformed cells increases, while the deformation of the cells becomes smaller, which causes the phenomenon that the global displacement keeps increasing, while the rate of the increment keeps decreasing as the number of cells in the generated grid increases.

In Table 4.1, the columns *Iterations* and *Function evaluations* represent the number of iterations and the number of function evaluations taken by the algorithm until it converges, respectively. We observe that in this example, the computational cost of a grid (including *CPU time*, *Iterations* and *Function evaluations*) increases as the grid becomes finer. This may be caused by the fact that the total number of deformed cells of the generated grid in this example increases as the grid becomes finer, as we mentioned above, and that the cell coordinates of the generated grid are transformed from the ones of a uniform grid, which is shown in Algorithm 1. As the number of grid points increases, more unknowns (i.e. coordinates of grid points) are involved in the discrete system (3.35) and need to be computed, which results in an increase in the computational cost.

In addition, we notice that in Table 4.1, the number of function evaluations is larger than the number of iterations, which indicates that there are extra function calls are made to adjust the value of a suitable regularizer λ in the process of the algorithm. The extra function calls are mainly made at the stages of initialization and termination, as we will see later from Figure 4.4 and Figure 4.5. According to our experiments, given a fixed prescribed density ρ', the choice of a suitable λ at each iteration of the algorithm stays almost the same when the grid size increases. Accordingly, the number of extra function calls (i.e. the difference between *Function evaluations* and *Iterations* in Table 4.1) made for adjusting λ throughout the whole process of the algorithm remains approximately the same as the grid becomes finer.

The behavior of the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the change of $\|\mathbf{R}^{(m)}\|$ (where $\mathbf{R}^{(m)} = \mathbf{A}^{(m)}\mathbf{U}^{(m)} - \mathbf{F}^{(m)}$ is the residual at m-th iteration) at each iteration of the algorithm are shown (in log scale) in Figure 4.4 and Figure 4.5. From the figures we can observe the three stages of the algorithm. The first two iterations are the stage of initialization, where the algorithm is in the process of determining a suitable regularizer λ. At this stage, $\|\mathbf{R}^{(m)}\|$ decreases slowly, since the current choice of λ is larger than the suitable value. Accordingly, the value of λ keeps decreasing, which leads to an increase in the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ (recall that a small λ corresponds to a large step size in the iteration, as we have shown in Section 3.4). Once a suitable regularizer is determined, the algorithm enters the stage of stabilization: $\|\mathbf{R}^{(m)}\|$ decreases faster than the previous stage and the iterates behave as expected. At the last stage, $\|\mathbf{R}^{(m)}\|$ stops decreasing and the value of a suitable regularizer keeps increasing until convergence has occurred. The increase of λ at this stage leads to a decrease in the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$, which can be seen from Figure 4.4.

Figure 4.6 shows two samples of the generated grids of different sizes. We can see that the shapes of both grids look like contractions centered at $(0.5, 0.5)$, as we described earlier. The computed results are consistent with the geometric features (such as the gradient of the density) of the prescribed density function ρ'.

Table 4.1: Example 1 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\max} = 10^6$, $\lambda_{\min} = 100$).

Number of cells	Error	$\sqrt{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
10×10	1.15×10^{-1}	0.0530	5.89	22	26
20×20	4.60×10^{-2}	0.0567	11.55	23	27
40×40	1.75×10^{-2}	0.0585	31.97	25	29
80×80	6.60×10^{-3}	0.0594	123.9	28	33

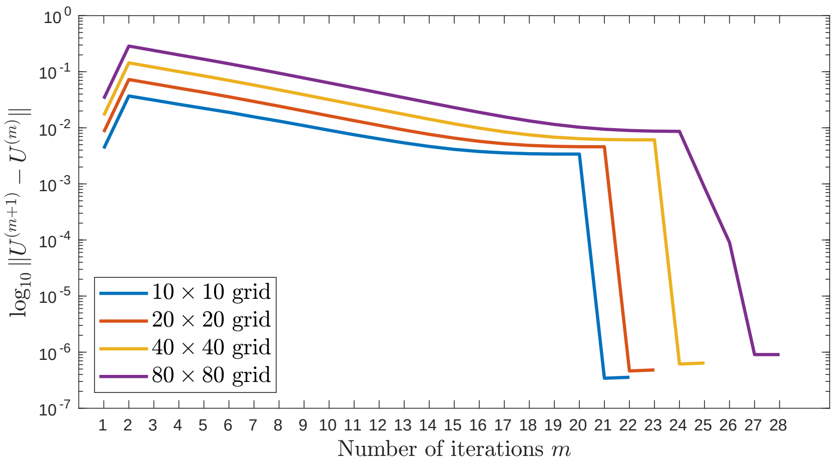


Figure 4.4: Example 1. Change of $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$.

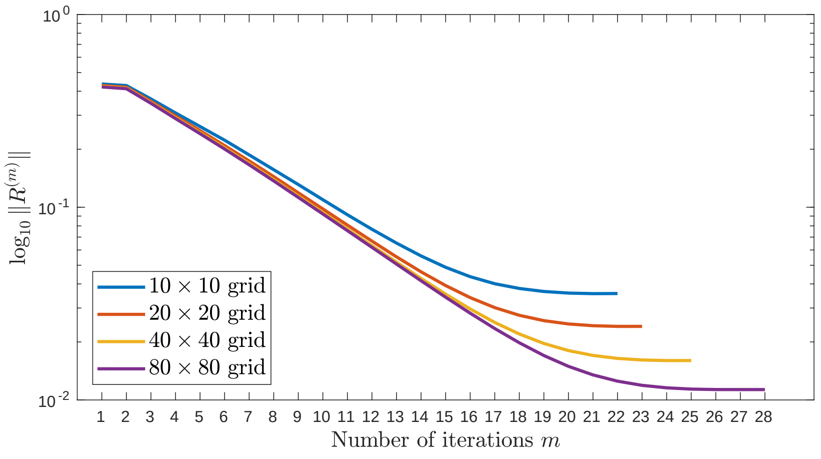


Figure 4.5: Example 1. Change of $\log_{10}\|\mathbf{R}^{(m)}\|$.

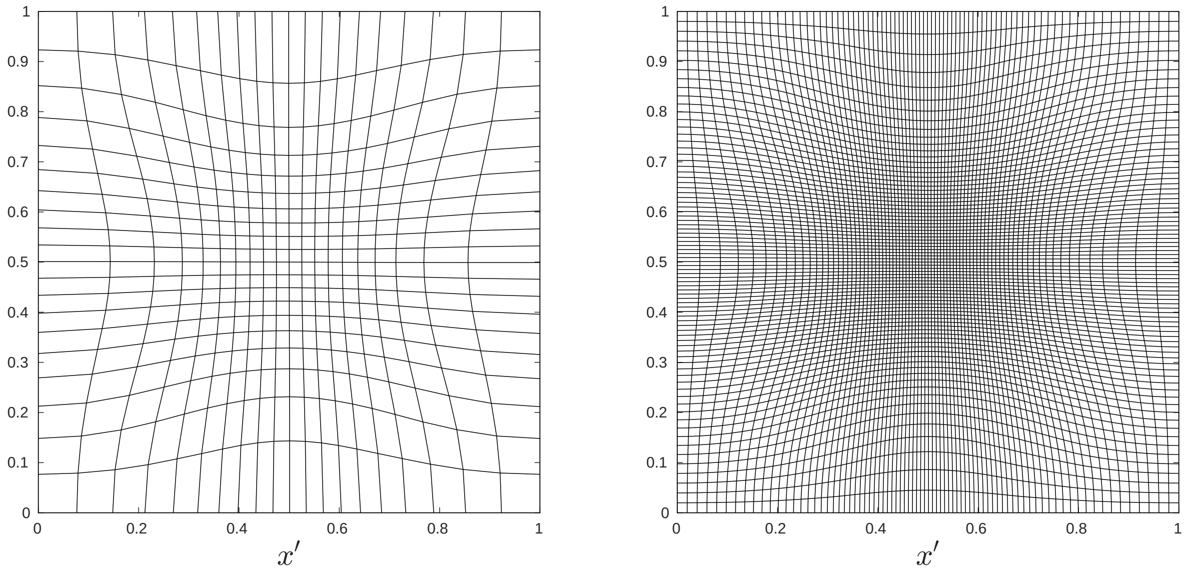


Figure 4.6: Example 1. Left: new grid of 20×20 cells; Right: new grid of 80×80 cells.

Example 2. Consider

$$\rho'(x', y') = \frac{0.01}{2 + \cos(8\pi\sqrt{(x' - 0.5)^2 + (y' - 0.5)^2})}$$

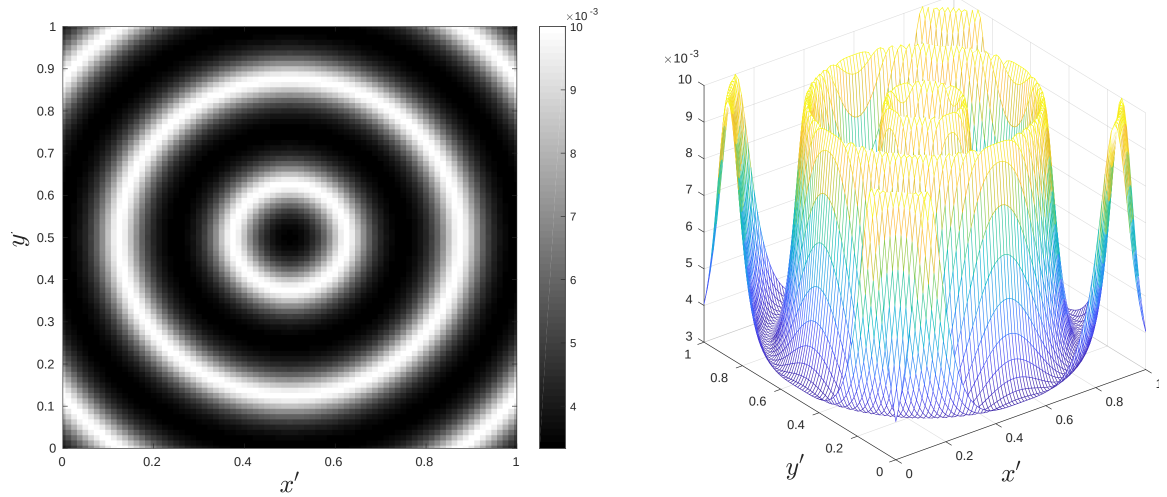


Figure 4.7: Example 2. Left: $\rho'(x', y')$. Right: A mesh plot of $\rho'(x', y')$ for illustrating the gradient of the densities.

Example 2 corresponds to a periodic diffusion centered at $(x', y') = (0.5, 0.5)$. Comparing with Example 1 which corresponds to a single diffusion, the gradient of the density ρ' in this example has more local diffusion patterns and the local convexity/concavity of the density function ρ' varies widely over the domain, as we can see from Figure 4.7. The computational difficulty is increased in this example, since we need competing dialations and contractions that arise when convexity/concavity changes.

The results of Example 2 are presented in Table 4.2. We note that, comparing with Example 1, the difference between the maximum and minimum values of $\rho'(x', y')$ in this example is smaller, therefore the equidistribution errors, the geometric mean $\sqrt{\mathbf{d}}$ of global displacement and the computational cost are all smaller. By comparing the numbers of iterations and the numbers of function evaluations, we observe that additional function evaluations are made to adjust the regularizer during the process of the computation.

Figures 4.8 and 4.9 show the change of the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the residual $\|\mathbf{R}^{(m)}\|$ of the discretized linear system that the algorithm solves for the numerical solution

U. We can see that, during the first two iterations of the algorithm, the value of $\|\mathbf{R}^{(m)}\|$ has been decreased slowly, since the current value of the regularizer λ is larger than a suitable value. The value of λ has thus been increased, which leads to an increase in the iterate $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ (see Figures 4.8). At the last a few iterations of the algorithms, the value of $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ keeps decreasing rapidly since λ is being reduced for verifying the occurrence of convergence. In general, for different grid size, the algorithm takes about the same amount of iterations to converge.

Figure 4.10 illustrates the generated grids of different sizes. We can see that the shapes of the grids look like a sequence of contractions and dilations, which is what we predicted; the generated grids are symmetric with respect to the point $(x', y') = (0.5, 0.5)$, which matches the symmetry properties of the prescribed density $\rho'(x', y')$.

Table 4.2: Example 2 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\text{max}} = 10^6$, $\lambda_{\text{min}} = 100$).

Number of cells	Error	$\sqrt{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
10×10	9.22×10^{-2}	0.0020	3.6	14	18
20×20	3.68×10^{-2}	0.0023	7.37	15	19
40×40	1.38×10^{-2}	0.0023	20.39	16	20
80×80	5.31×10^{-3}	0.0023	69.8	16	20

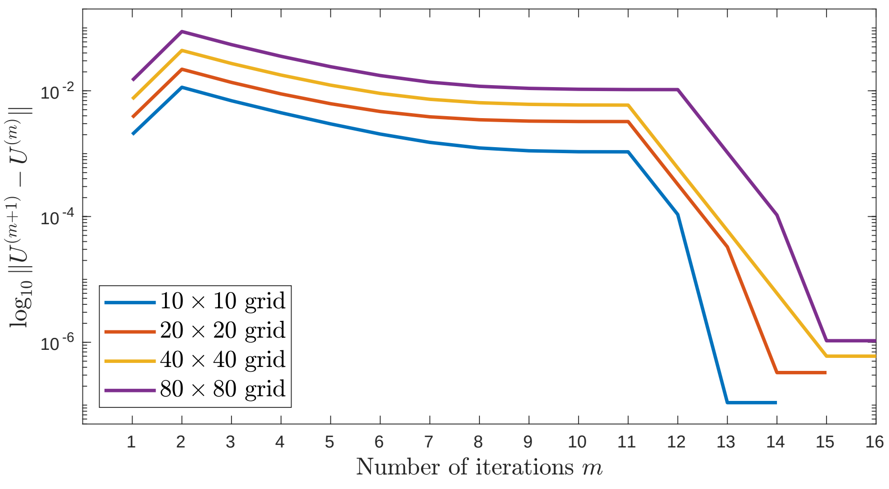


Figure 4.8: Example 2. Change of $\log_{10}\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$.

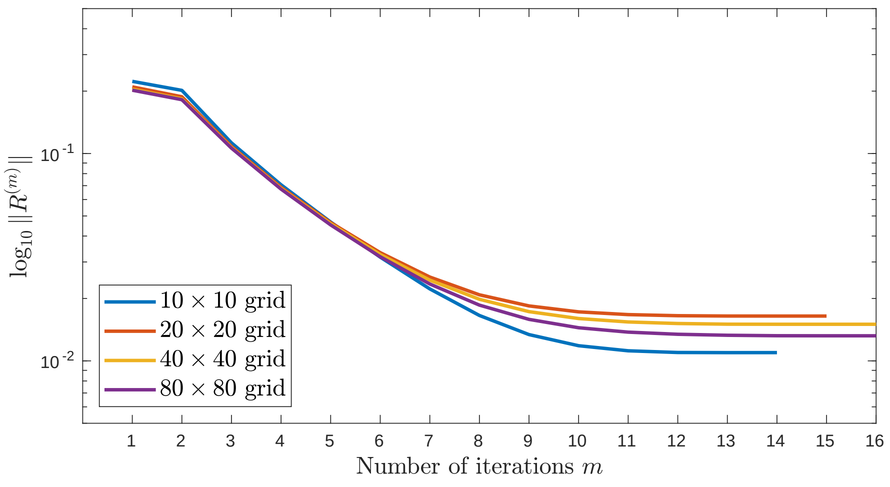
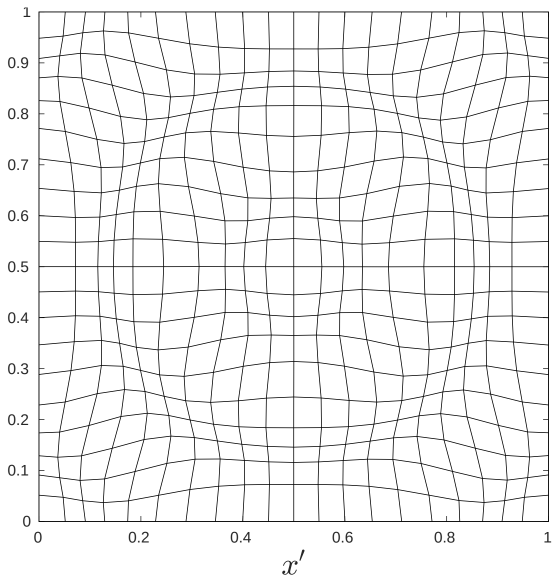


Figure 4.9: Example 2. Change of $\log_{10}\|\mathbf{R}^{(m)}\|$.



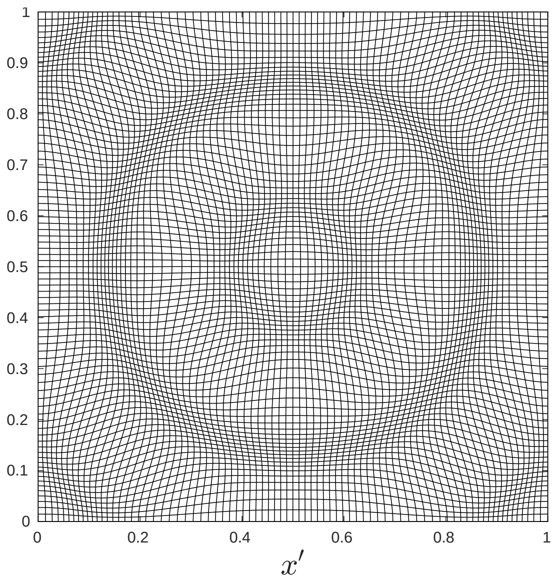


Figure 4.10: Example 2. Left: new grid of 20×20 cells; Right: new grid of 80×80 cells.

Example 3.

$$\rho'(x',y') = \left(\frac{2}{(5r\cos(\theta - 10r^2))^2 + 1} + 1\right),$$

where $r = \sqrt{(x'-0.7)^2 + (y'-0.5)^2}$, $\quad \theta = \tan^{-1}\left(\frac{y'-0.5}{x-0.7}\right)$.

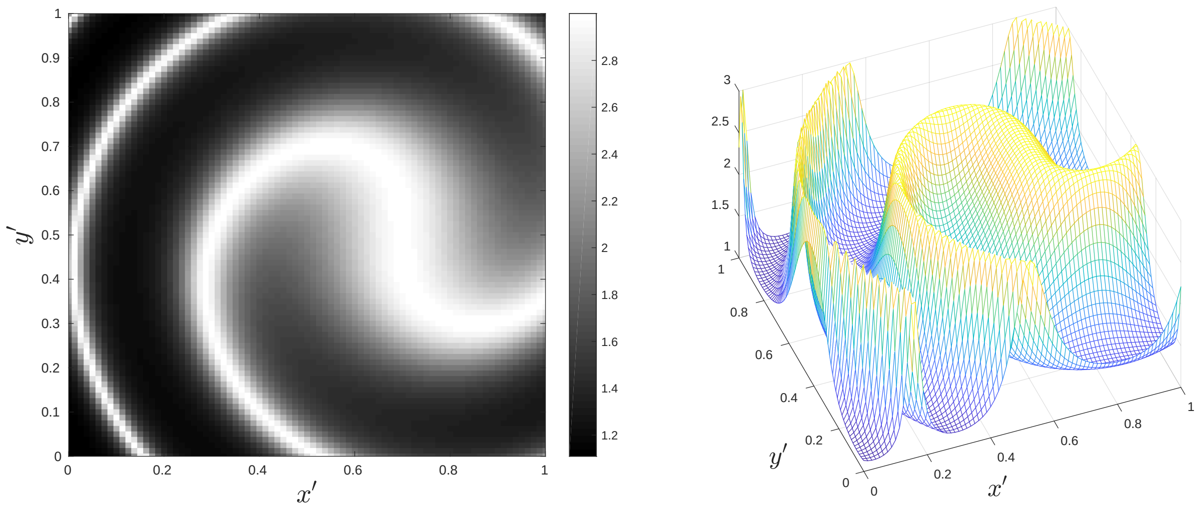


Figure 4.11: Example 3. Left: $\rho'(x',y')$. Right: A mesh plot of $\rho'(x',y')$ for illustrating the gradient of the densities.

Example 3 is a spiral centered at $(x',y') = (0.7, 0.5)$; the geometric features of the density $\rho'(x',y')$ is shown in Figure 4.11. Comparing with Example 2, the difference between the maximum and minimum values of ρ' in this example is much larger; the diffusion patterns of the gradient of ρ' in this example are also more irregular. The computational difficulty is increased in this example, since the high and irregular variance of the local convexity/concavity of the density function ρ' makes it harder to generate properly displaced grid points.

The results of this example are shown in Table 4.3. We notice that the geometric mean $\sqrt{\mathbf{d}}$ of the global displacement of the generated grid in this example decreases as the grid becomes finer. This phenomena is likely caused by the geometric features of the density $\rho(x',y')$. From Figure 4.11 we notice that there are many grid points, such as the center $(x',y') = (0.7, 0.5)$, at which the rate of change in $\rho'(x',y')$ is close to zero. The new grid points that are transformed from these grid points tend to be approximately uniform grid points. As the generated grid $\mathbf{x}'$ becomes finer, the portion of deformed cells in $\mathbf{x}'$ becomes smaller, which lead to a decrease in the global displacement $\mathbf{d}$ of $\mathbf{x}'$. We also note that, when the size of the generated grid is small, the global displacement of the generated grid tends to be large due to truncation error arising from the discretization of the HJB equation. From Table 4.3 we can also see that the number of iterations and the number of function evaluations decreases as the grid becomes finer, which matches the change of $\sqrt{\mathbf{d}}$.

Figures 4.12 and 4.13 show the changes of $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and $\|\mathbf{R}^{(m)}\|$ during the whole process of the algorithm. We can see that the algorithm went through the stages of initialization, stabilization and termination based on these changes. The stage of initialization in this example costs one more iteration than that in the previous examples, which indicates that the performance of the algorithm is more sensitive to the choice of the regularizer λ on this problem. Overall, the process of the algorithm in this example is similar to those in the previous examples. Figure 4.14 shows the new grids generated for $\rho'(x', y')$ in this example. We observe that the geometric features of $\rho'(\mathbf{x})$ in Figure 4.11 are reflected on the deformed grids in Figure 4.14.

Table 4.3: Example 3 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\max} = 10^6$, $\lambda_{\min} = 10$).

Number of cells	Error	$\sqrt{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
10×10	6.51×10^{-2}	0.0639	10.07	23	28
20×20	3.24×10^{-2}	0.0559	10.11	20	26
40×40	1.33×10^{-2}	0.0529	25.09	18	23
80×80	5.01×10^{-3}	0.0523	89.42	18	23

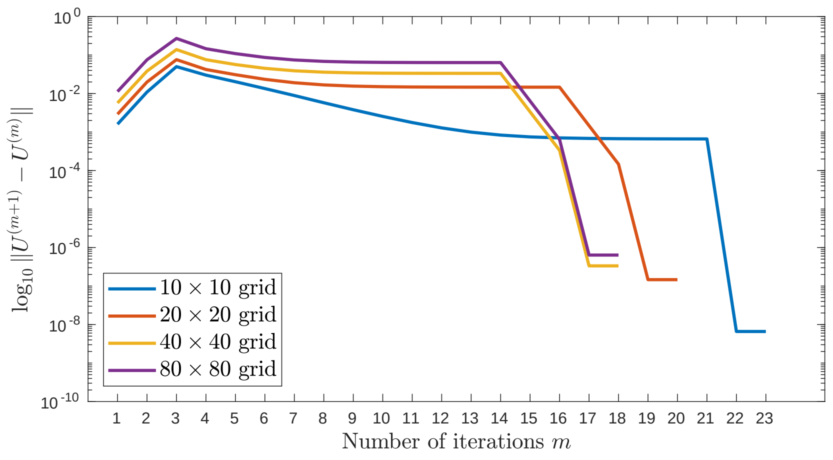


Figure 4.12: Example 3. Change of $\log_{10} \|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$.

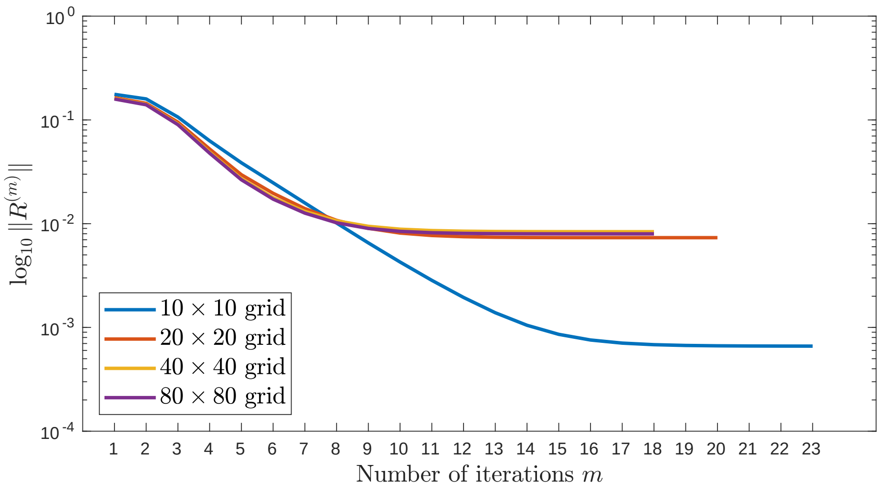


Figure 4.13: Example 3. Change of $\log_{10}\|\mathbf{R}^{(m)}\|$.

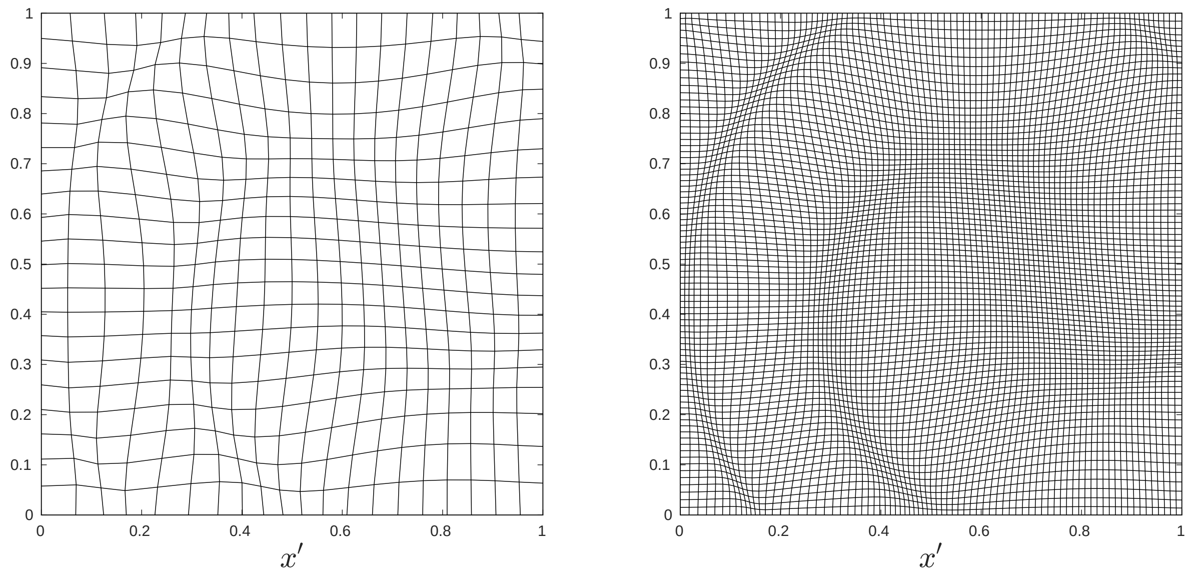


Figure 4.14: Example 3. Left: new grid of 20×20 cells; Right: new grid of 80×80 cells.

3D Case

Example 4. Consider the density function

$$\rho'(x', y', z') = \exp\left(-\frac{(x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2}{0.05}\right) + 0.1. \tag{4.6}$$

This example corresponds to a diffusion centered at $(x', y', z') = (0.5, 0.5, 0.5)$ and it can be considered as the 3D generalization of Example 1. We set the parameters λ, $\lambda_{\max}$, $\lambda_{\min}$, λ_{amp} and λ_{damp} in this example to be the same as the ones in Example 1 in order to compare the two examples.

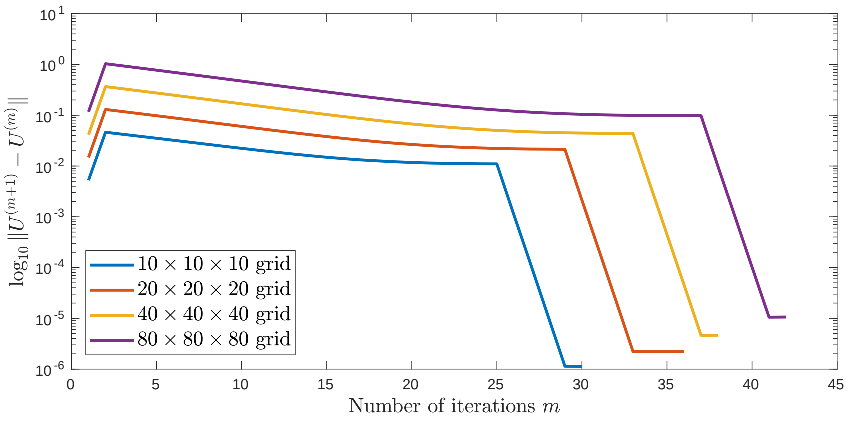
The results of Example 4 are shown in Table 4.4. We can see that the equidistribution error decreases as the grid becomes finer, which is similar to our observation on Example 1. When the number of cells in the grid increases, the total number of deformed cells increases, which leads to an increase in the geometric mean $\sqrt[3]{\mathbf{d}}$ of global displacement; meanwhile, the deformation of each cell becomes smaller, which causes the rate of the increment of $\sqrt[3]{\mathbf{d}}$ keeps decreasing. The computational cost also increases as the grid becomes finer. By comparing the number of iterations and the number of function evaluations in Table 4.4, we know that addtional functions calls are made in order to adjust the value of the regularizer λ during the computational process. In general, the computational time of a 3D grid is much longer than that of a 2D grid due to the exponential growth in the number of grid points.

The changes of the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the residual $\|\mathbf{R}^{(m)}\|$ in this example are presented in Figure 4.15. From the figure we can see that the three stages of the algorithm. At the first two iterations, the value of $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ increases due to the adjustment of the value of the regularizer; Meanwhile, the value of $\|\mathbf{R}^{(m)}\|$ decreases slowly. After the stage of initialization, the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ keeps decreasing and $\|\mathbf{R}^{(m)}\|$ is decreasing fast until convergence.

We present the generated grids $\mathbf{x}'$ by plotting 2D slices of $\mathbf{x}'$. In this example, we plot slices of $\mathbf{x}'$ only along the z-axis due to the symmetry properties held by $\rho'(x', y', z')$. The plots are shown in Figure 4.16. We can see that the deformation of the grid becomes less noticeable as z' changes from 0.5 to approximately 0.031, because the difference between the maximum value and the minimum value of $\rho'(x', y', z')$ for $z' = 0.031$ is smaller than that for $z' = 0.5$.

Table 4.4: Example 4 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\max} = 10^6$, $\lambda_{\min} = 100$).

Number of cells	Error	$\sqrt[3]{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
$10 \times 10 \times 10$	1.99×10^{-1}	0.0997	1.1808×10^2	30	34
$20 \times 20 \times 20$	1.13×10^{-1}	0.1065	1.0437×10^3	36	40
$40 \times 40 \times 40$	6.29×10^{-2}	0.1099	7.5584×10^3	38	42
$80 \times 80 \times 80$	3.80×10^{-2}	0.1117	2.7821×10^5	42	46



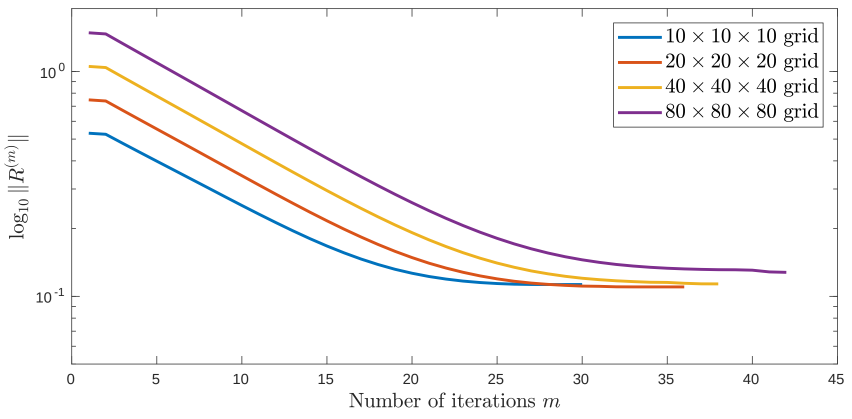


Figure 4.15: Example 4. Top: change of $\log_{10} \|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$; Bottom: change of $\log_{10} \|\mathbf{R}^{(m)}\|$.

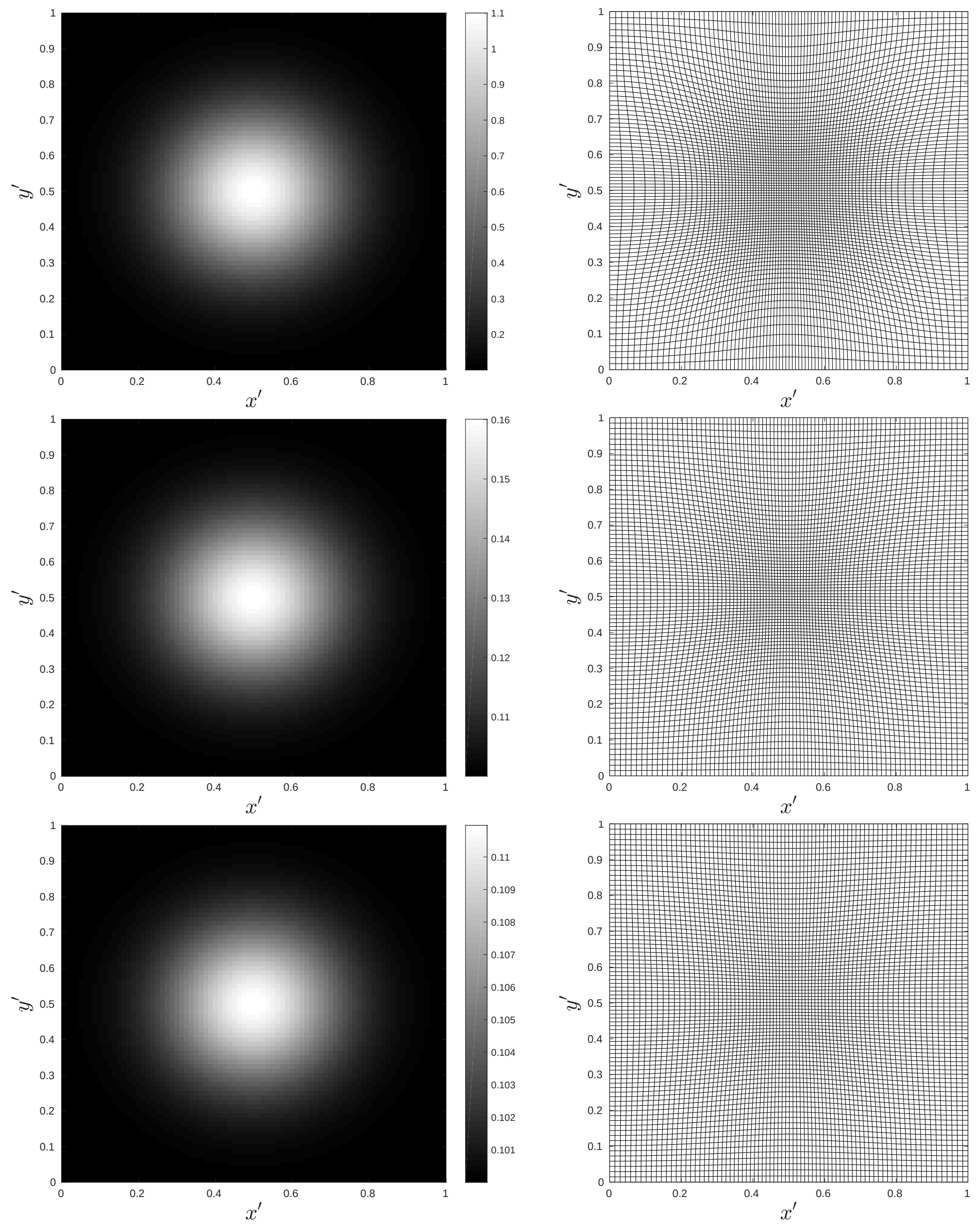


Figure 4.16: Example 4. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' \approx 0.5$, $z' \approx 0.125$, $z' \approx 0.031$.

Example 5. Consider

$$\rho'(x', y', z') = \frac{0.01}{2 + \cos(8\pi\sqrt{(x' - 0.5)^2 + (y' - 0.5)^2 + (z' - 0.5)^2})}.$$

This example corresponds to a periodic diffusion centered at $(x', y', z') = (0.5, 0.5, 0.5)$; the density function in this example is generalized from the one in Example 2. We set the parameters λ, λ_{max}, λ_{min}, λ_{amp} and λ_{damp} in this example to be the same as the ones in Example 2 in order to compare the two examples.

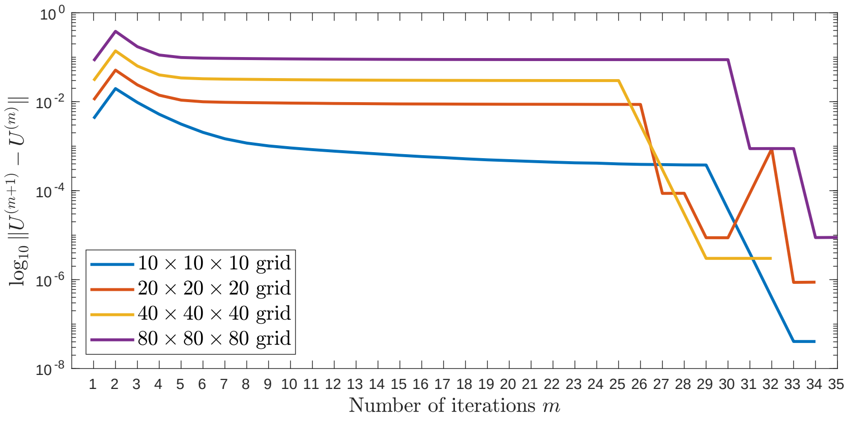
The results of Example 5 are shown in Table 4.5. We can see that the equidistribution error, the geometric mean $\sqrt[3]{\mathbf{d}}$ of the global displacement and the computational cost all increase as the dimension of the grid increases. For the three dimensional grids in this example, as the grid becomes finer, the equidistribution error decreases; meanwhile, the value of $\sqrt[3]{\mathbf{d}}$ does not increase much, since the difference between the maximum value ρ'_{max} and minimum value ρ'_{min} of ρ' is small. The difference between the number funtion evaluations and the number of iterations indicates that there are extra function calls made for adjusting λ at some iterations of the algorithm. We observe that, comparing with grids of other sizes, the number of iterations taken for generating a $40 \times 40 \times 40$ grid is smaller. This is likely to occur when the grid is very fine and the difference $|\rho'_{\text{max}} - \rho'_{\text{min}}|$ is small; in this case, the trunction errors in spatial discretization dominate the computation error.

Figure 4.17 shows the changes of the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the residual $\|\mathbf{R}^{(m)}\|$ in this example. We notice that after the first 10 iterations, the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the residual $\|\mathbf{R}^{(m)}\|$ appear decreasing more slowly: $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ decrease by approximately 10^{-4} and $\|\mathbf{R}^{(m)}\|$ decreases by approximately 10^{-3}. One can choose a different metric for convergence in order to terminate the algorithm sooner and get a similar result of which the equidistribution error is slightly larger. In addition, we observe that the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ that correspond to generating a 20×20 grid have briefly increased at the final stage of the algorithm. This happends because the residual $\|\mathbf{R}^{(m)}\|$ is still decreasing at those iterations, in which case the algorithm keeps reducing the value of the regularizer λ in order to get a fast convergence. When λ decreases, the step size of the iteration increases, which results in an increase in $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$.

The generated grids $\mathbf{x}'$ are shown in Figures 4.18 and 4.19. We plot slices of $\mathbf{x}'$ only along the z-axis due to the symmetry properties of $\rho'(x', y', z')$. From the figures we can see that the shape of the generated grid matches the geometric features of the density ρ'.

Table 4.5: Example 5 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\max} = 10^6$, $\lambda_{\min} = 100$).

Number of cells	Error	$\sqrt[3]{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
$10 \times 10 \times 10$	3.776×10^{-1}	0.0107	1.3559×10^2	34	38
$20 \times 20 \times 20$	2.476×10^{-1}	0.0123	9.5838×10^2	34	37
$40 \times 40 \times 40$	1.372×10^{-1}	0.0126	6.5522×10^3	32	36
$80 \times 80 \times 80$	7.560×10^{-2}	0.0126	2.6293×10^5	35	40



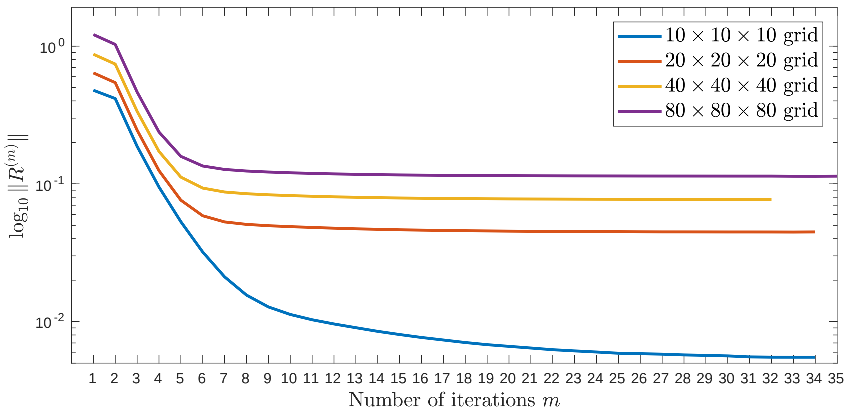


Figure 4.17: Example 5. Top: change of $\log_{10} \|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$; Bottom: change of $\log_{10} \|\mathbf{R}^{(m)}\|$.

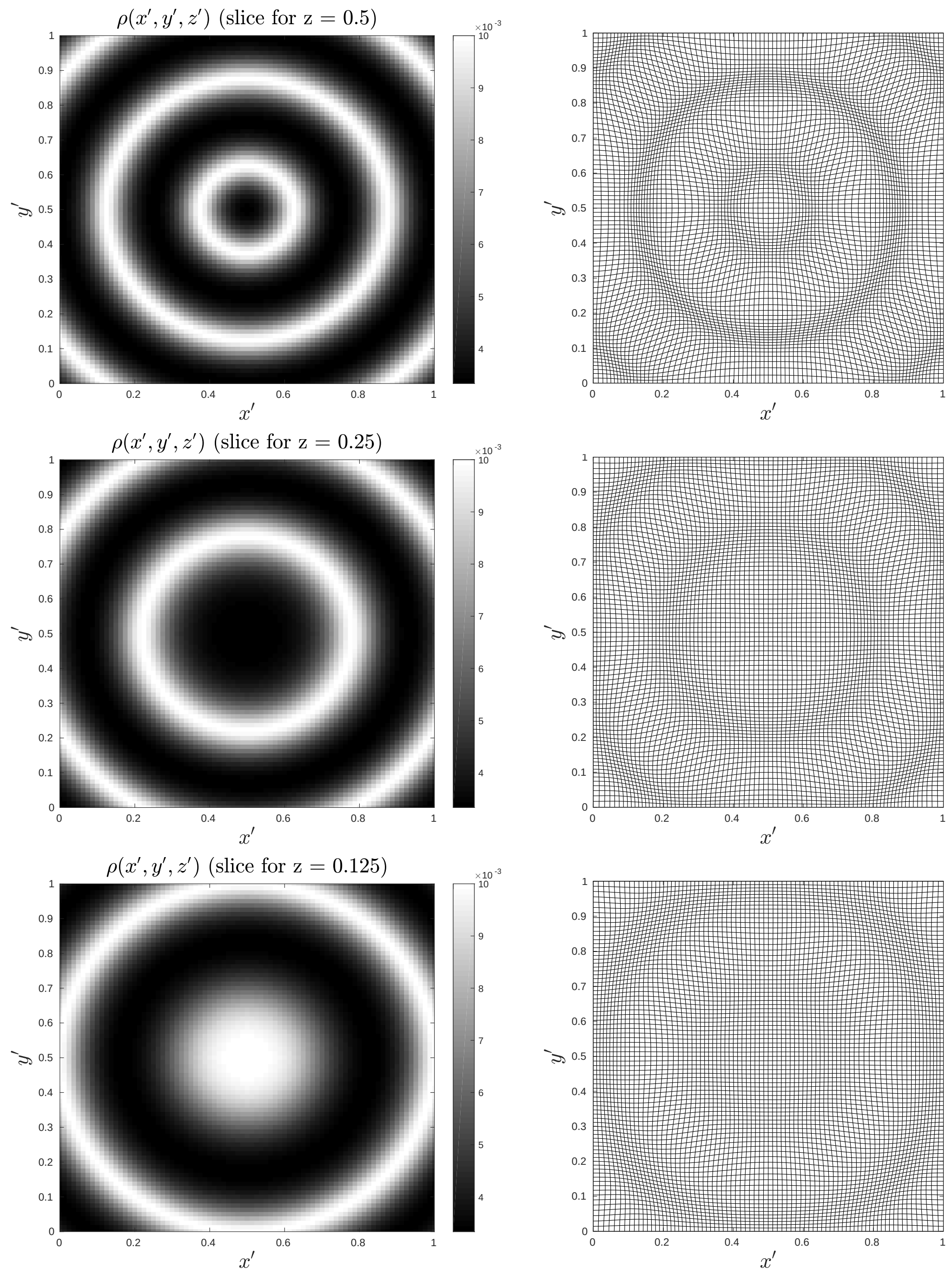


Figure 4.18: Example 5. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' \approx 0.5$, $z' \approx 0.25$, $z' \approx 0.125$.

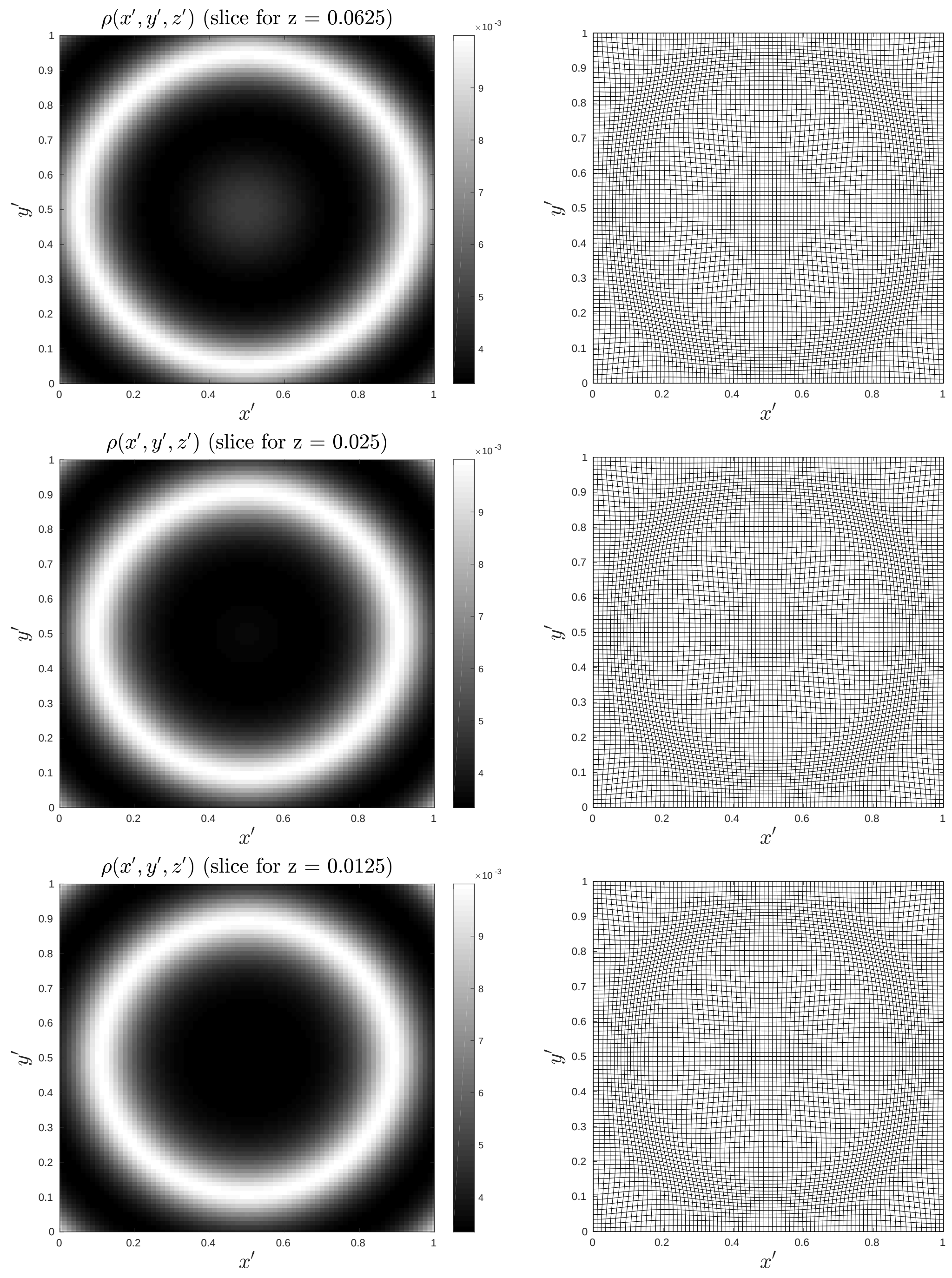


Figure 4.19: Example 5. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' \approx 0.0625$, $z' \approx 0.025$, $z' \approx 0.0125$.

Example 6. Consider the density function

$$\rho'(x',y',z') = \left(\frac{2}{(5r\cos(\theta - 10r^2))^2 + 1} + 1\right),$$

where $r = \sqrt{(x'-0.7)^2 + (y'-0.5)^2 + (z'-0.3)^2}$, $\quad \theta = \tan^{-1}\left(\frac{y'-0.5}{x-0.7}\right)$.

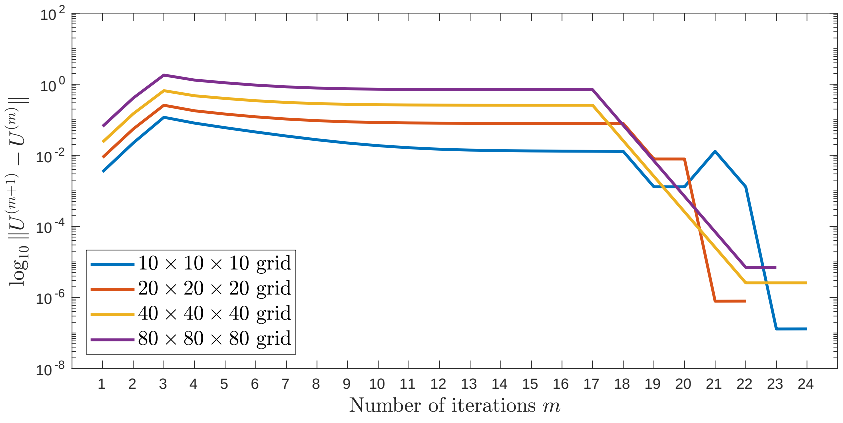
The $\rho'(x',y',z')$ in this example is similar to a 3D generalization of the density function in Example 3; the projection of $\rho'(x',y',z')$ on the $x'-y'$ plane corresponds to a spiral centered at $(x',y') = (0.7, 0.5)$. From Figures 4.21 to 4.23 we can see that this example is relatively more complicated than the previous ones, since the diffusion pattern of the density function $\rho'(x',y',z')$ in this example is asymmetric and non-uniformly distributed. We use this example to test the robustness of our algorithm.

The results of Example 6 are shown in Table 4.6. The equidistribution error decreases as the number of grid points increases. Similar to Example 3, the geometric mean $\sqrt[3]{\mathbf{d}}$ of the global displacement in this example decreases as the grid becomes finer. This phenomenon is likely caused by the geometric features of $\rho'(x',y',z')$ and truncation errors that arise from the spatial discretization of the HJB equation. From Table 4.6 we also observe that the computation of a small grid does not always take fewer iterations/function evaluations than that of a big grid does. The number of iterations/function evaluations taken by the computation depends on the portion of deformed cells among all the cells in the generated grid, since the generated grid is transformed from a uniform grid (as it is shown in Algorithm 1). If the portion of deformed cells is small, then the algorithm tends to take fewer iterations/function evaluations to converge. In addition, when the computed grid is fine, the truncation errors arise in spatial discretization may dominate the numerical error, in which case the number of iterations taken by the algorithm may decrease as the number of grid points increases.

Figure 4.20 shows the changes of the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ and the residual $\|\mathbf{R}^{(m)}\|$. At the beginning of the computation of each grid, the value of $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ keeps increasing since the algorithm keeps reducing the value of λ for a faster convergence. When a suitable value of λ is determined, the computational process becomes stabilized, and $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ behaves as expected; meanwhile, $\|\mathbf{R}^{(m)}\|$ decreases fast. At the final stage, $\|\mathbf{R}^{(m)}\|$ gradually stops decreasing, and $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ decreases since the suitable value of λ increases. We observe that the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$ that correspond to a 10×10 grid has briefly increased at the final stage of the algorithm. This occurrs since the residual $\|\mathbf{R}^{(m)}\|$ is still decreasing at these iterations. Accordingly, the algorithm keeps reducing λ for a faster convergence, which leads to an increase in the iterates $\|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$. In general, the process of the algorithm in this example is similar to those in the previous examples. Figures 4.21 to 4.23 show slices of the generated grids $\mathbf{x}'$ along z-axis, y-axis and x-axis. From the figures we can see that the shape of the generated grids matches the geometric features of the prescribed density $\rho'(x', y', z')$.

Table 4.6: Example 6 ($\lambda = 10^3$, $\lambda_{\text{amp}} = 10$, $\lambda_{\text{damp}} = 0.1$, $\lambda_{\max} = 10^6$, $\lambda_{\min} = 10$).

Number of cells	Error	$\sqrt[3]{\mathbf{d}}$	CPU time (s)	Iterations	Function evaluations
$10 \times 10 \times 10$	2.292×10^{-1}	0.1497	1.0385×10^2	24	30
$20 \times 20 \times 20$	1.672×10^{-1}	0.1364	6.6468×10^2	22	28
$40 \times 40 \times 40$	9.550×10^{-2}	0.1312	5.0173×10^3	24	29
$80 \times 80 \times 80$	5.091×10^{-2}	0.1303	1.7582×10^5	23	28



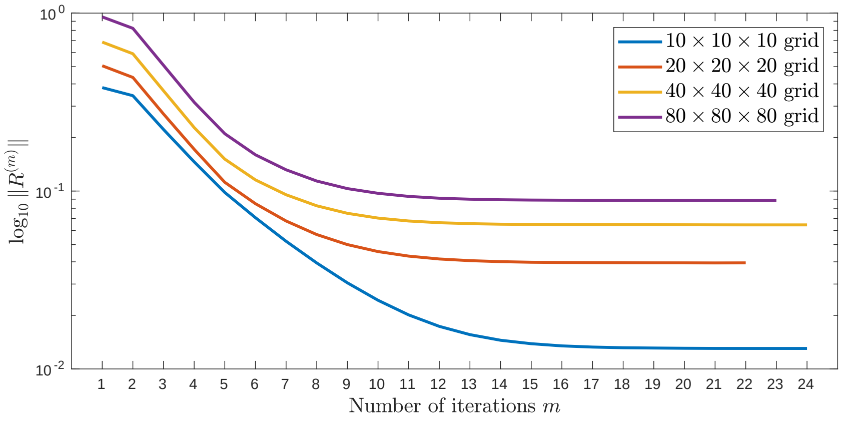


Figure 4.20: Example 6. Top: change of $\log_{10} \|\mathbf{U}^{(m+1)} - \mathbf{U}^{(m)}\|$; Bottom: change of $\log_{10} \|\mathbf{R}^{(m)}\|$.

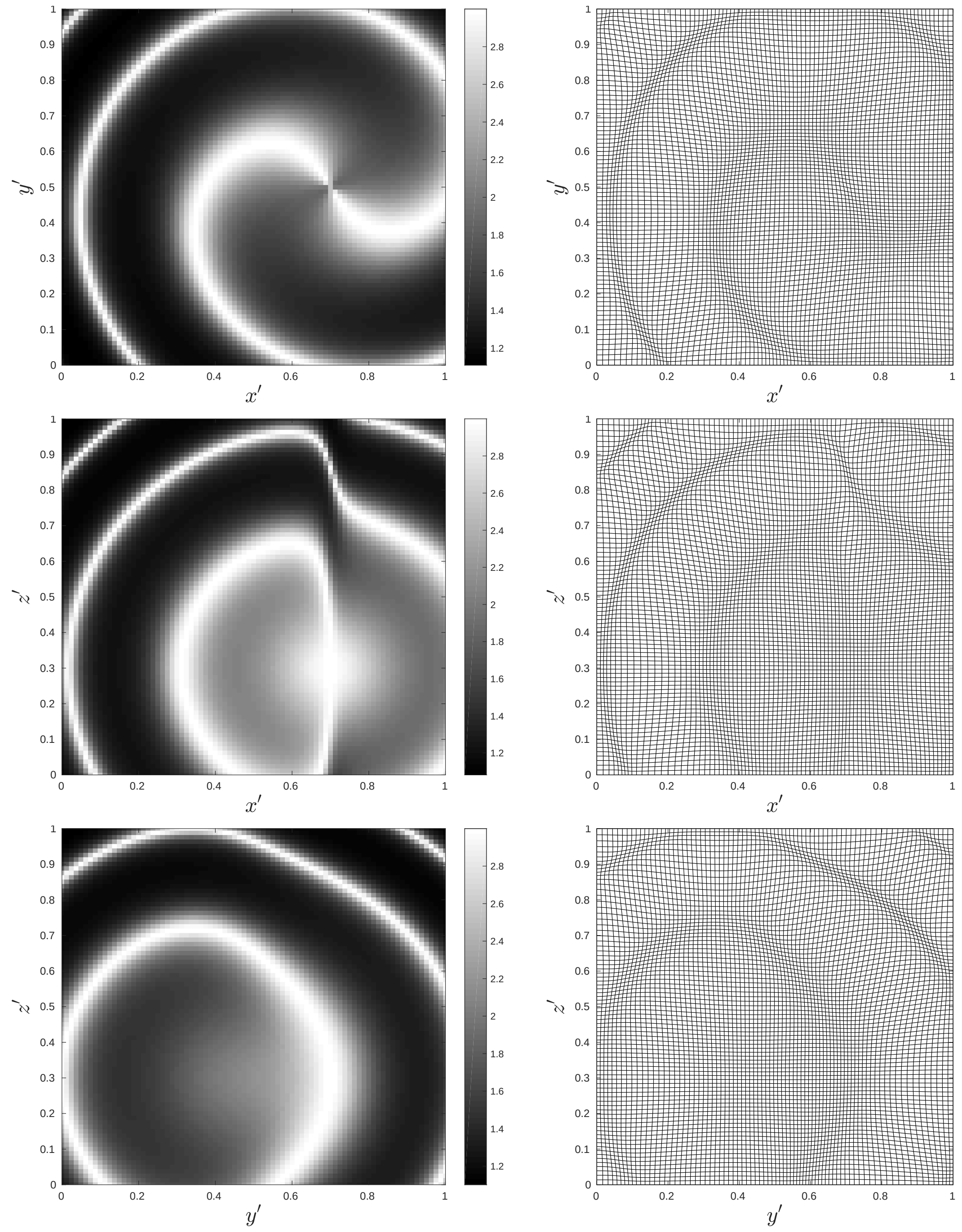


Figure 4.21: Example 6. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' = 0.525$, $y' = 0.525$, $x' = 0.525$.

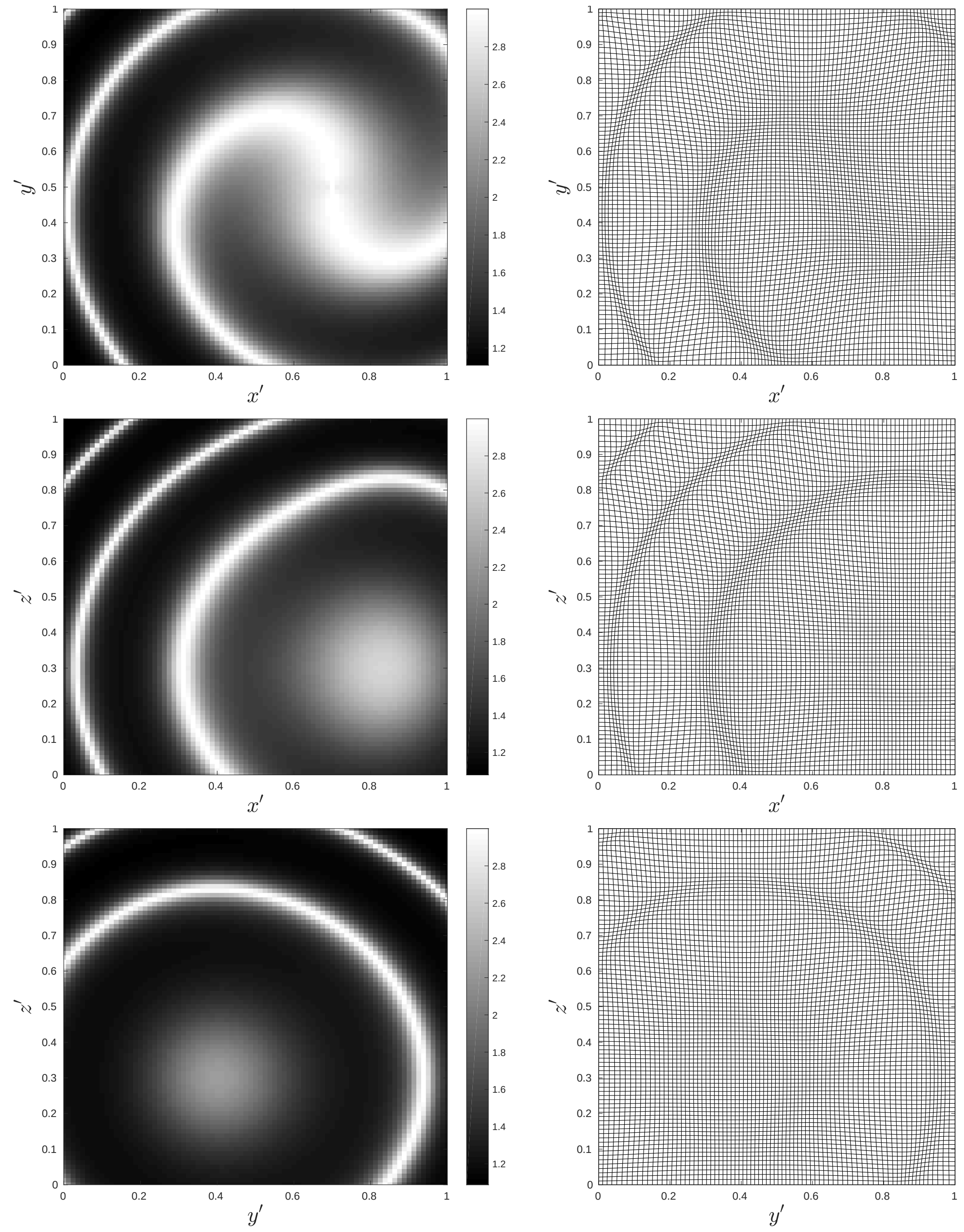


Figure 4.22: Example 6. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' = 0.25$, $y' = 0.25$, $x' = 0.25$.

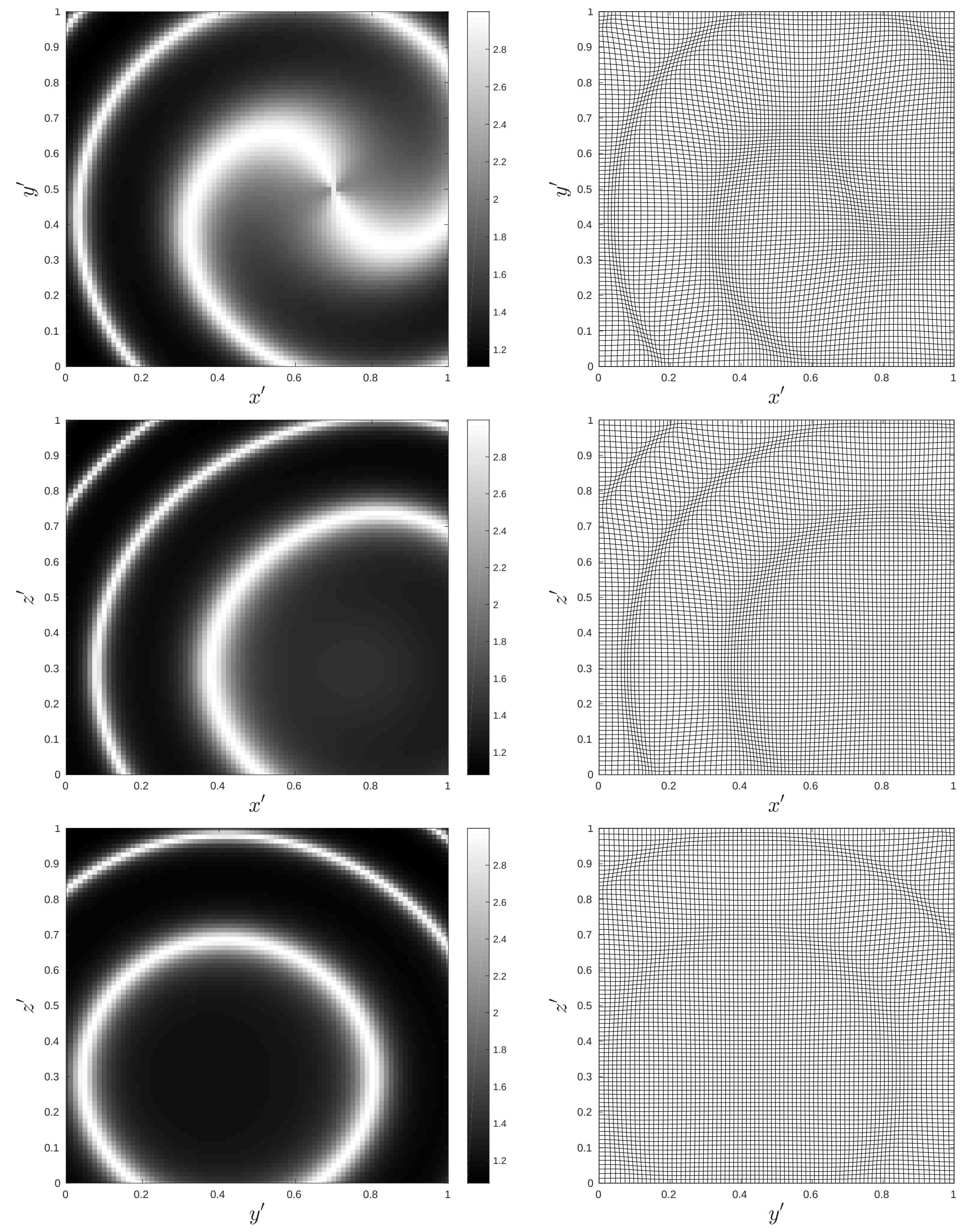


Figure 4.23: Example 6. From Left to right: $\rho'(x', y', z')$, generated 80×80 grid; From top to bottom: slice for $z' = 0.125$, $y' = 0.125$, $x' = 0.125$.

Chapter 5

Conclusion

In this thesis, we have proposed a numerical method for grid generation in two and three dimensions based on MK optimization. The two-dimensional grid generation method is developed by modifying an existing technique of Chen and Wan [6], which involves using an adaptive regularizer to reduce the computational cost of solving the HJB formulation of the MA equation that arises from the MK optimization. The three dimensional grid generation method is a generalization of the two-dimensional method. While the theory behind the generalization to higher dimensions is straightforward, the implementation is not. The implementation requires overcoming several computational challenges. For instance, the optimal controls in the two-dimensional case have closed-forms (Chen and Wan [6]), while the ones in the three-dimensional case have no closed-form and can only be approximated using numerical methods such as Newton's method.

Chapter 4 shows successful numerical results of our algorithms. In all of the examples, the geometric features of the prescribed densities are reflected by the deformation of the newly generated grids. By inspecting the results, we observe that the equidistribution error of a generated grid is more likely dominated by the spatial discretization error when the grid is sparse, as we can see a decrease occurring in the increment of the equidistribution error when the number of cells in the grid increases; when the grid is more refined, the equidistribution error is more likely caused by the truncation error arising from the discretization of the HJB equation.

In terms of efficiency, our numerical experiments have indicated that the use of adaptive regularizer leads to rapid convergence. In fact, when using constant regularizers (e.g., $\lambda = 10^4$), a small example on 10×10 (or $10\times10\times10$) grid can take up to 2000 (or 4000) function evaluations to obtain convergence to the displayed results. With adaptive regularizers, the numbers of function evaluations required to generate a 10×10 (or $10 \times 10 \times 10$) grid drops to less than 30 (or less than 50). The examples on $80 \times 80 \times 80$ in Chapter 4 (implemented in Matlab 2013b) are completed in approximately 24 hours on a computing machine with 132 GB of memory (CPU model: Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz). Without using adaptive regularizers, these large examples would take several weeks of computation time to obtain the same results. We note that in our work, the discrete system (3.39) is solved using unstructured method. The matrix $A(C^{(m)})$ under the current controls $\mathcal{C}^{(m)}$ in (3.39) is a sparse banded matrix. To further accelerate the computation, one can consider using a banded solver to solve the system (3.39).

The work contained in this thesis provides a unified method for solving the Monge-Ampère equation arising from grid generation with an experimental analysis of robustness and demonstrates improvements over existing techniques by reducing the number of iterations required. Future work and other open problems relating to this work involve:

- performing a detailed theoretical convergence/sensivity analysis to determine the optimal regularizer on each step;
- adapting the method to other finite difference/finite element methods for discretizing the HJB equation so that it can generate grids of triangular or other convex polygon meshes;
- modifying the method so that it computes a coarse grid and uses it as an initial guess for generating a finer grid in order to accelerating the method for fine grids.

References

[1] Guy Barles and Panagiotis E Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic analysis*, 4(3):271–283, 1991.

[2] Eric Berger. Please, please stop sharing spaghetti plots of hurricane models. https://arstechnica.com/science/2017/09/please-please-stop-sharing-spaghetti-plots-of-hurricane-models/, September 2017.

[3] Chris Budd. Monge-Ampère based methods for mesh generation on the sphere.

[4] Chris J Budd, Weizhang Huang, and Robert D Russell. Adaptivity with moving grids. *Acta Numerica*, 18:111–241, 2009.

[5] Yangang Chen and Justin W.L. Wan. Numerical method for image registration model based on optimal mass transportation. *Preprint*, 2015.

[6] Yangang Chen and Justin W.L. Wan. Monotone mixed narrow/wide stencil finite difference scheme for Monge-Ampère equation. *arXiv preprint arXiv:1608.00644*, 2016.

[7] Michael G Crandall, Hitoshi Ishii, and Pierre-Louis Lions. Users guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1):1–67, 1992.

[8] Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[9] Bernard Dacorogna and Jürgen Moser. On a partial differential equation involving the Jacobian determinant. In *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*, volume 7, pages 1–26. Elsevier, 1990.

[10] Carl de Boor. Good approximation by splines with variable knots. In *Spline functions and approximation theory*, pages 57–72. Springer, 1973.

[11] Gian Luca Delzanno, Luis Chacón, John M Finn, Y Chung, and Giovanni Lapenta. An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge–Kantorovich optimization. *Journal of Computational Physics*, 227(23):9841–9864, 2008.

[12] Peter R Eiseman. Adaptive grid generation. *Computer Methods in Applied Mechanics and Engineering*, 64(1-3):321–376, 1987.

[13] Brittany D Froese and Adam 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.

[14] Cristian E Gutiérrez and Haim Brezis. *The Monge-Ampère equation*. Springer, 2016.

[15] Weizhang Huang, Yuhe Ren, and Robert D Russell. Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM Journal on Numerical Analysis*, 31(3):709–730, 1994.

[16] Weizhang Huang and David M Sloan. A simple adaptive grid method in two dimensions. *SIAM Journal on Scientific Computing*, 15(4):776–797, 1994.

[17] Martin Knott and Cyril S Smith. On the optimal mapping of distributions. *Journal of Optimization Theory and Applications*, 43(1):39–49, 1984.

[18] Nikolai Vladimirovich Krylov. Control of a solution of a stochastic integral equation. *Theory of Probability & Its Applications*, 17(1):114–130, 1972.

[19] Guojun Liao and Dale Anderson. A new approach to grid generation. *Applicable Analysis*, 44(3-4):285–298, 1992.

[20] Pierre-Louis Lions. Hamilton-Jacobi-Bellman equations and the optimal control of stochastic systems. In *Proc. Intern. Congress Math., Warsaw*, 1983.

[21] Vladimir D Liseikin. *Grid generation methods*, volume 1. Springer, 1999.

[22] Kyle T Mandli and Clint N Dawson. Adaptive mesh refinement for storm surge. *Ocean Modelling*, 75:36–50, 2014.

[23] Jürgen Moser. On the volume elements on a manifold. *Transactions of the American Mathematical Society*, 120(2):286–294, 1965.

[24] Iain Smears. Hamilton-Jacobi-Bellman equations. analysis and numerical analysis. http://www.math.dur.ac.uk/Ug/projects/highlights/PR4/Smears HJB report. pdf.

[25] Mohamed Sulman, JF Williams, and Robert D Russell. Optimal mass transport for higher dimensional adaptive grid generation. *Journal of computational physics*, 230(9):3302–3330, 2011.

[26] Joe F Thompson. A survey of dynamically-adaptive grids in the numerical solution of partial differential equations. *Applied Numerical Mathematics*, 1(1):3–27, 1985.

[27] Mark K Transtrum and James P Sethna. Improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization. *arXiv preprint arXiv:1201.5885*, 2012.

[28] Hilary Weller, Philip Browne, Chris Budd, and Mike Cullen. Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge-Ampère type equation. *arXiv:1608.00644*, 2015.

[29] Hilary Weller, Todd Ringler, Matthew Piggott, and Nigel Wood. Challenges facing adaptive mesh modeling of the atmosphere and ocean. *Bulletin of the American Meteorological Society*, 91(1):105–108, 2010.