NEW DEVELOPMENT OF THE DEFORMATION METHOD

by

JIE LIU

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2006

ACKNOWLEDGEMENTS

My deepest gratitude goes to my supervising professor Dr. Guojun Liao, without whose advice, encouragement and support none of this would be possible. Many thanks to Dr. Hristo Kojouharov, Dr. Jianzhong Su, Dr. Chaoqun Liu and Dr. Yue Liu for taking their precious time to serve as my committee. I also want to thank all the faculty and staff in the Mathematics Department at The University of Texas at Arlington for their help and support through my graduate studies.

I would like to thank my parents for their support and guidance through my whole life. Without their encouragement and selfless contribution, I wouldn’t even think about a degree of Doctor of Philosophy. Special thanks to my brother and sister-in-law. I could not have achieved this goal without their support. Finally, I want to thank my husband Mingsheng for his love, support and encouragement. I also want to thank my son Phillip (Chenxin) and my daughter Cindy (Xinyue) for all the joys they brought to me.

April 11, 2006

ABSTRACT

NEW DEVELOPMENTS OF THE DEFORMATION METHOD

Publication No. ______

Jie Liu, PhD.

The University of Texas at Arlington, 2006

Supervising Professor: Guojun Liao

New developments of deformation method for grid generation are presented in this work. Theorems for three different cases and different methods for implementing deformation method are presented. One of the new developments is a 3D multi-block moving grid method. In this version, a Poisson equation is solved by finite difference method to get the vector field for moving grid. Special treatment applies to the common boundary of different blocks. Another new development is a numerical method for reconstructing a given differentiable transformations by solving a system of div-curl equation directly formed from each point of the graph. The determinacy and ellipticity of the system of the 3D div-curl equation are analyzed in detail. And the least-square finite element method is used to solve the div-curl equation in order to reconstruct a differentiable mapping. Both 2D and 3D implementations are presented in this work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

In order to solve partial differential equations (PDEs) numerically, we first need to discretize the continuous differential equation into a system of algebraic difference equations. Accompanying with this, the solution field (physical domain) needs to be discretized into elements or cells, i.e. grids (meshes) have to be generated. The grids generated for solving PDEs are very crucial for the accuracy of the numerical solution of PDEs, especially for problems with very rapid variations or sharp layers. Grid generation becomes an important tool for the computational simulation of various physical phenomena, like fluid flow, heat transfer, acoustic propagation, just name a few. After J. Thompson et al. wrote the book [1] about grid generation in 1985, it has been developed both technologically and analytically. Various grid generation codes (GRIDGEN, TRIANGLE, etc.), free or commercial, are developed by researchers in computational field simulation. Mathematical aspects of numerical grid generation are discussed in [13] to provide a deeper understanding of different algorithms and their limitations. General grid generation methods are discussed in the books by Knupp and Steinberg [2], Carey [3], and Thompson [4]. In 1996, a reflection on grid generation was done by J. Thompson [5] after two decades on its growth.

Grid generation methods are typically grouped into two categories: structured and unstructured. Structured grid utilizes quadrilateral (2D) and hexahedral (3D) elements in a computationally rectangular array. The topology of the elements is fixed. The connectivity among nodes in a structured grid is completely defined by the nodes indices. For example, a node $x_{i,j}$ in a 2D structured grid is connected with $x_{i-1,j}, x_{i+1,j}, x_{i,j-1},$ and $x_{i,j+1}$, This type of connectivity allow us to use finite difference method which is very efficient for solving PDEs. Classical methods for generating structured grids are transfinite interpolation (TFI) [2] and solving elliptic (or hyperbolic) systems of PDEs. Unstructured grid utilizes an arbitrary collection of elements to fill the domain. Triangles and quadrilaterals are used in 2D and tetrahedral and hexahedra for 3D. The number of edges sharing a node is not restricted. Algorithms like Delaunay triangulation and Vornoi diagram are typically used to generate unstructured grid. Finite element or Finite volume method are applied when we have unstructured grids.

Methods combining structured and unstructured grids are also developed. Those include multiblock grids, hybrid grids, chimera grids and hierarchical grids. Multiblock grids can be described as an unstructured collection of structured blocks. The domain is decomposed into several blocks. The grid for each block is structured while the connections between blocks are unstructured. Several block to block connection methods have developed. These include point to point, many points to one point and arbitrary connections. In first case, blocks must match topologically and physically at the boundary. In the second case, the blocks must be topologically similar but not the same at the boundary. The third case is where the blocks must be physically similar at the boundary but can have significant topological differences. Here are some advantage and disadvantage of multi-block grids.

Advantage:

- Unstructured collection of structured blocks. It is unstructured in block level and structured inside each block.
- Provide a measure of flexibility for complex geometrics.
- Allow different physical or mechanical models in different blocks.
- Allow different grid refinement strategies for different blocks.
- Leads to parallel computing with different blocks assigned to different computer processors.

Disadvantage:

- Construction of a multi-block grid requires domain decomposition, which is still an unsolved big problem in multi-block grid generation.
- Data communication between blocks is another challenge.

Comparison of advantages and drawbacks for different type of grids can be found in [6]. Detailed information about existing grid types and generation techniques is provided in [1],[2],[3],[4].

Accuracy of the solution and computational efficiency are the two main concerns in computational grid generation area. For problems with shock waves, boundary layers, etc. very fine grids over a small portion of the physical domain are required in order to resolve the large solution variations. For fixed grid, adding grid points to the whole domain to improve accuracy will cost computational efficiency. To improve computational efficiency by reduce grid points will reduce the accuracy. So in order to improve accuracy and efficiency at the same time, we introduce adaptive grid. The idea is to always put dense grids on the part which has large variation and coarse grids on the part which is smooth. For time dependent problems with salient features, since the position of the part with large variation is changing in different time step, we want to move the grid in order to keep track of the change of the solution in different time step.

Local mesh refinement (h-refinement) and moving mesh (r-refinement) are two main methods for generating adaptive grids. Local refinement method adapts the grid by locally adding points to the part with large variation and removing points from the part with low variation. This technique has gotten the most attention for the past several decades since the refinement is easily prescribed and error analysis is easily carried out [7]. The disadvantage of this method is that the data structure of the grid has to be adjusted every time the grid is adapted. And consequently we have to change the data structure of the solver. Moving grid methods relocate grid points to refine the grid where needed. The total number of points and the connection between grid points are always kept the same so that there is no need to change the data structure of the slover. A number of techniques have been developed by researchers for generating moving grids. Moving finite element methods where nodal points are driven by the residual of the finite element approximation have been developed by Miller, Carson and Baines (See [8][9][10][11][12]). In [14], a variety of approaches for generating moving adaptive methods are summarized and compared. In [19], a moving mesh method takes the advantage from both h-refinement and r-refinement is proposed. A moving mesh finite element method is designed to solve the incompressible Navier-Stokes equations in [20].

Different numerical methods are combined with moving grid techniques. Moving finite difference ([15],[16])and moving finite element ([17],[18])algorithms are developed mainly in the past. Some adaptive moving techniques in finite element and finite difference solutions of partial differential equations are reviewed in [21], In [22], a moving mesh finite volume method is developed. Recently, the idea of meshless adaptation is also implemented in [39][40]. An overview of the meshless methods can be found in [41].

In this work, some new developments of the moving deformation method developed by Liao et al ([28],[29]) are presented. Theorems for three different cases and different methods for implementing deformation method are presented in chapter 2. A 3D multi-block moving grid method based on deformation method is developed in chapter 3. In chapter 4, the div-curl system is analyzed. Least-square finite element method is used to solve the div-curl equation in order to reconstruct a differentiable mapping. The idea comes from the implementation of deformation method. This may have potential applications in image registration and computer vision simulation, which currently are currently hot research topics. Both 2D and 3D implementations are presented in this work.

CHAPTER 2

THE DEFORMATION METHOD

The deformation method is based on the idea of equivalent volume elements of a compact Riemannian manifold [27]. In 1992, Liao and Anderson proposed this new method in [29]. In this new approach a grid can be constructed by moving the grid points such that specified cell volumes can be achieved. A monitor function is defined and used to obtain a vector field by solving one linear Poisson equation. The grid points are moved according to a velocity field related to the vector field obtained. The mathematical principles behind this method guarantee that grid lines of the same grid family will not cross each other. In [29], the transformation Jacobian determinant, and consequently the cell volumes, was specified on the old grid before adaptation. In [32], the method is improved so that cell volumes can be specified as functions of the new grid after adaptation. In [31], this method is further extended into a real time moving grid method and used for solving one-dimensional unsteady problems. Some 1D and 2D applications and more analysis of adaptive moving grid by deformation method were done in [30]. In [35], an adaptive deformation method is applied to solve the compressible Euler equations for field flows. A least-square finite element deformation method is developed in [37] and applied in [34] to a nonlinear problem. A 2D moving grid geometric deformable model using deformation method is developed by X. Han in [36] for segmentation of image processing.

Let's take a look at the following three versions of deformation method.

2.1 Case 1

This is one of the steady versions of deformation method where the transformation Jacobian determinant is specified on the old grid ξ before adaptation.

Problem: Given a monitor function $f(\xi)$, find a mapping $\phi_1(\xi)$ such that

$$J(\phi_1) = \det \nabla \phi_1(\xi) = f(\xi) \tag{2.1}$$

We can use the following two steps to find such a mapping.

Step 1: Find a vector field $V(\xi)$ that satisfies:

$$\mathrm{div} V(\xi) = f(\xi) - 1 \tag{2.2}$$

Step 2: Form $V_t = \dfrac{V}{t+(1-t)f}$, then find $\phi_t(\xi)$ by solving the following ODE

$$\frac{d\phi_t(\xi)}{dt} = V_t(\phi_t) \qquad t \in [0,1] \tag{2.3}$$

Here $\phi_t(\xi) = \phi(\xi, t)$, and let $\phi_1(\xi) = \phi(\xi, t=1)$.

Now, let us show that $\phi_1(\xi)$ found by this way satisfies (2.1):

In order to prove this, let

$$\begin{aligned} H(t,\xi) &= J(\phi_t(\xi))\left[t+(1-t) f(\phi_t(\xi))\right] \\ &= (\det \nabla \phi_t(\xi))\left[t+(1-t) f(\phi_t(\xi))\right] \end{aligned} \tag{2.4}$$

We can show

$$\frac{\partial H}{\partial t} = 0 \tag{2.5}$$

Since $\phi_0(\xi) = \phi(\xi, t=0)$ is the identity mapping, we have $\det \nabla \phi_0(\xi) = 1$ and $\phi_0(\xi) = \xi$

So $$H(0,\xi) = (\det \nabla \phi_0(\xi)) f(\phi_0(\xi)) = f(\xi) \tag{2.6}$$

Also by (2.4) we have $$H(1,\xi) = \det \nabla \phi_1(\xi) \tag{2.7}$$

Thus (2.1) follows by (2.5), (2.6) and (2.7).

In order to prove (2.5), we need to introduce Abel’s Lemma first.

Abel’s Lemma:

Let M be a $n \times n$ matrix such that each element of the matrix is differentiable on t. If $\frac{d}{dt}(M) = AM$ where A is a $n \times n$ matrix, then $\frac{d}{dt}(\det M) = (\text{trace } A)(\det M)$.

This is a standard lemma, which can be found for in [26] or other standard ODE books. For completeness, we outline its proof here.

Proof:

Let $M = \begin{pmatrix} m_{11} & \cdots & m_{1n} \\ \cdots & \cdots & \cdots \\ m_{n1} & \cdots & m_{nn} \end{pmatrix}$

Assume $\frac{d}{dt}M = \begin{pmatrix} m'_{11} & \cdots & m'_{1n} \\ \cdots & \cdots & \cdots \\ m'_{n1} & \cdots & m'_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} m_{11} & \cdots & m_{1n} \\ \cdots & \cdots & \cdots \\ m_{n1} & \cdots & m_{nn} \end{pmatrix} = AM$

Then $$m'_{i,j} = \sum_{k=1}^{n} a_{ik} m_{kj} \qquad (i,\, j = 1,\dots,n) \tag{2.8}$$

$$\frac{d}{dt}(\det M) = \begin{vmatrix} m'_{11} & m'_{12} & \cdots & m'_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix} + \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m'_{21} & m'_{22} & \cdots & m'_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix}$$
$$+ \dots + \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m'_{n1} & m'_{n2} & \cdots & m'_{nn} \end{vmatrix} \tag{2.9}$$

Plug (2.8) into (2.9) and do row equivalent operations to each of the determinants, we can reduce (2.9) to

$$\frac{d}{dt}(\det M) = \begin{vmatrix} a_{11}m_{11} & a_{11}m_{12} & \cdots & a_{11}m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix} + \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ a_{22}m_{21} & a_{22}m_{22} & \cdots & a_{22}m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix}$$

$$+ \dots + \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{nn}m_{n1} & a_{nn}m_{n2} & \cdots & a_{nn}m_{nn} \end{vmatrix}$$

$$= a_{11} \det M + a_{22} \det M + \dots + a_{nn} \det M$$

$$= (a_{11} + a_{22} + \dots + a_{nn}) \det M$$

$$= (traceA)(\det M)$$

□

Now let's prove (2.5).

Proof:

$$\frac{\partial H}{\partial t} = \frac{\partial}{\partial t}\Big[\big(\det \nabla \phi_t(\xi)\big)\big(t + (1-t)\, f\,(\phi_t(\xi))\big)\Big]$$

$$= \frac{\partial}{\partial t}\Big[\big(\det \nabla \phi_t(\xi)\big)\Big]\Big[t + (1-t)\, f\,(\phi_t(\xi))\Big] \tag{2.10}$$

$$+ \big(\det \nabla \phi_t(\xi)\big)\frac{\partial}{\partial t}\Big[t + (1-t)\, f\,(\phi_t(\xi))\Big]$$

Since $\frac{d}{dt}(\nabla \phi) = \nabla\left(\frac{d\phi}{dt}\right) = \nabla\big(V_t(\phi(\xi))\big) = \big(\nabla_\phi V_t\big)(\nabla \phi)$, by Abel's Lemma we get:

$$\frac{\partial}{\partial t}(\det \nabla \phi) = \big(\operatorname{trace}(\nabla_\phi V_t)\big)(\det \nabla \phi) \tag{2.11}$$

$$\nabla_\phi V_t = \begin{vmatrix} \frac{\partial V_1}{\partial \phi_1} & \frac{\partial V_2}{\partial \phi_1} & \frac{\partial V_3}{\partial \phi_1} \\ \frac{\partial V_1}{\partial \phi_2} & \frac{\partial V_2}{\partial \phi_2} & \frac{\partial V_3}{\partial \phi_2} \\ \frac{\partial V_1}{\partial \phi_3} & \frac{\partial V_2}{\partial \phi_3} & \frac{\partial V_3}{\partial \phi_3} \end{vmatrix}$$

So
$$\operatorname{trace}(\nabla_\phi V_t) = \frac{\partial V_1}{\partial \phi_1} + \frac{\partial V_2}{\partial \phi_2} + \frac{\partial V_3}{\partial \phi_3} = \operatorname{div}_\phi V_t \tag{2.12}$$

Putting (2.12) into (2.11), we have

$$\frac{\partial}{\partial t}(\det \nabla \phi) = \big(\operatorname{div}_\phi V_t\big)(\det \nabla \phi) \tag{2.13}$$

Plugging (2.13) into (2.10), we have:

$$\frac{\partial H}{\partial t} = \left(\operatorname{div}_\phi V_t\right)\left(\det\nabla\phi\right)\left[t+(1-t)f\right] + \left(\det\nabla\phi\right)\frac{\partial}{\partial t}\left[t+(1-t)f\right]$$

$$= \left(\operatorname{div}_\phi V_t\right)\left(\det\nabla\phi\right)\left[t+(1-t)f\right] + \left(\det\nabla\phi\right)\left[1-f+(1-t)(\nabla f)V_t\right] \quad (2.14)$$

$$= \left(\det\nabla\phi\right)\left\{\left(\operatorname{div}_\phi V_t\right)\left[t+(1-t)f\right] + \left[1-f+(1-t)(\nabla f)V_t\right]\right\}$$

By step 2, we have

$$V_t = \frac{V}{t+(1-t)f}$$

So $$V = V_t\left[t+(1-t)f\right]$$

$$\Rightarrow divV = \left(divV_t\right)\left[t+(1-t)f\right] + V_t(1-t)\nabla f$$

$$\Rightarrow \left(divV_t\right)\left[t+(1-t)f\right] = divV - V_t(1-t)\nabla f \quad (2.15)$$

Plugging (2.15) into (2.14), we get:

$$\frac{\partial H}{\partial t} = \left(\det\nabla\phi\right)\left\{\left[divV - V_t(1-t)\nabla f\right] + \left[1-f+(1-t)(\nabla f)V_t\right]\right\}$$

$$= \left(\det\nabla\phi\right)\left(divV + 1 - f\right) \quad (2.16)$$

By step 1, plugging (2.2) into (2.16), we get

$$\frac{\partial H}{\partial t} = \left(\det\nabla\phi\right)\left(f-1+1-f\right) = 0$$

□

Now, our main problem is how to implement step 1, that is how to find $V(\xi)$ such that $\operatorname{div}V(\xi) = f(\xi)-1$. We have three different methods.

Method 1: Direct construction.

Method 2: Solve the Poisson equation $\Delta\omega = f - 1$ for ω, then let $V = \nabla\omega$.

The V found out by this way satisfies

$divV = div(\nabla\omega) = \Delta\omega = f - 1$.

Method 3: Solve the div-curl system $\begin{cases} divV = f - 1 \\ curlV = 0 \end{cases}$. Least-square finite element method is a good way to solve it.

We will discuss method 2 in the application of Chapter three for multi-block moving grid. In Chapter four we use method 3 to reconstruct a mapping. Here let us see some details about method 1.

In 2D, we need to find a vector field $V(V_1, V_2)$ on $\Omega = [0,1]\times[0,1]$ such that $\text{div}V = f - 1 = g$ for a normalized monitor function $\iint\limits_{\Omega} f = 1$

Let $G(x_1, x_2) = \int_0^{x_1} g(t, x_2)\,dt$

Define:
$$V : \begin{cases} V_1(x_1, x_2) = G(x_1, x_2) - h(x_1)G(1, x_2) \\ V_2(x_1, x_2) = h'(x_1)\int_0^{x_2} G(1, t)\,dt, \end{cases} \tag{2.17}$$

where $h(x_1)$ is a function satisfying $h(0) = 0$, $h(1) = 1$ and $h'(0) = h'(1) = 0$. For example: $h(t) = \frac{1}{2}(1 - \cos\pi t)$. Then

$$\begin{aligned}
divV &= V_{1x_1} + V_{2x_2} \\
&= \frac{\partial}{\partial x_1}\left[G(x_1,x_2) - h(x_1)G(1,x_2)\right] + \frac{\partial}{\partial x_2}\left[h'(x_1)\int_0^{x_2} G(1,t)\,dt\right] \\
&= g(x_1,x_2) - h'(x_1)G(1,x_2) + h'(x_1)G(1,x_2) \\
&= g(x_1,x_2) \\
&= f(x_1,x_2) - 1.
\end{aligned}$$

So the vector constructed by (2.17) satisfies.

In 3D, we need to find a vector field $V(V_1,V_2,V_3)$ such that $divV = f - 1$.

Define:

$$V:\begin{cases}
V_1(x_1,x_2,x_3) = \int_0^{x_1} g(t_1,x_2,x_3)\,dt_1 - h(x_1)\int_0^1 g(t_1,x_2,x_3)\,dt_1 \\
V_2(x_1,x_2,x_3) = h'(x_1)\left(\int_0^{x_2}\int_0^1 g(t_1,t_2,x_3)\,dt_1dt_2 - h(x_2)\int_0^1\int_0^1 g(t_1,t_2,x_3)\,dt_1dt_2\right) \\
V_3(x_1,x_2,x_3) = h'(x_1)h'(x_2)\int_0^{x_3}\int_0^1\int_0^1 g(t_1,t_2,x_3)\,dt_1dt_2dt_3,
\end{cases}$$

where $h(x_1)$, $h(x_2)$ are functions satisfying $h(0)=0$, $h(1)=1$, and $h'(0)=h'(1)=0$.

$$\begin{aligned}
divV &= V_{1x_1} + V_{2x_2} + V_{3x_3} \\
&= g(x_1,x_2,x_3) - h'(x_1)\int_0^1 g(t_1,x_2,x_3)\,dt_1 + h'(x_1)\int_0^1 g(t_1,x_2,x_3)\,dt_1 \\
&\quad - h'(x_1)h'(x_2)\int_0^1\int_0^1 g(t_1,t_2,x_3)\,dt_1dt_2 + h'(x_1)h'(x_2)\int_0^1\int_0^1 g(t_1,t_2,x_3)\,dt_1dt_2 \\
&= g(x_1,x_2,x_3) \\
&= f(x_1,x_2,x_3) - 1
\end{aligned}$$

Another interesting direct construction method is worked out by Liao and Su in [28].

2.2 Case 2

This is another static version of deformation method where the transformation Jacobian determinant is specified on the new grid $\phi(\xi)$ before adaptation.

Problem: Given g and f (properly normalized), find a transformation $\phi : \partial\Omega \to \partial\Omega$ such that

$$g(\xi)\,J\left(\phi(\xi)\right) = f\left(\phi(\xi)\right), \quad \xi \in \Omega \qquad (2.18)$$

g and f must satisfy $\int_\Omega \frac{1}{f} = \int_\Omega \frac{1}{g}$.

We can use the following three steps to find such a transformation.

Step 1. Compute V such that

$div(V(\xi)) = \frac{1}{g(\xi)} - \frac{1}{f(\xi)}$ in Ω, and $V(\xi)\cdot\vec{n} = 0, \qquad \xi \in \partial\Omega$.

Step 2. For each fixed node ξ, solve the ODE

$$\frac{\partial\varphi(\xi,t)}{\partial t} = \eta\left(\varphi(\xi,t),t\right) \qquad 0 \le t \le 1$$

with $\varphi(\xi,0) = \xi$, where $\eta(x,t) = \dfrac{V(x)}{t\dfrac{1}{f(x)} - (1-t)\dfrac{1}{g(x)}}$

Step 3. Define $\phi(\xi) = \varphi(\xi,1)$, then ϕ will be the solution.

Now, let's show that the ϕ found out by these three steps satisfies (2.18).

Let $$H(\xi,t) = (J(\varphi)(\xi,t))\left(t\frac{1}{f(\varphi(\xi,t))} + (1-t)\frac{1}{g(\varphi(\xi,t))}\right) \tag{2.19}$$

If we can show (2.19) is independent of t, i.e.

$$\frac{\partial H}{\partial t} = 0 \tag{2.20}$$

then $H(\xi,0) = J(\varphi(\xi,0))\frac{1}{g(\varphi(\xi,0))} = 1/g(\xi)$ and

$$H(\xi,1) = (J(\varphi(\xi,1)))\frac{1}{f(\varphi(\xi,1))} = J/f(\phi(\xi))$$

$$\frac{\partial H}{\partial t} = 0 \Rightarrow H(\xi,0) = H(\xi,1) \Rightarrow 1/g(\xi) = J/f(\phi(\xi)) \Rightarrow gJ = f$$

The proof of (2.20) is very similar to the proof of the first case. Detailed proof can be found in [43].

2.3 Case 3

This is the version of deformation method with real time adaptation.

Problem: Given a monitor function $f(\xi,t)$ (normalized with $\int \frac{1}{f} = |\Omega|$, where $|\Omega|$ is the volume of the domain), find a transformation $\phi : \Omega_1 \to \Omega_2$ such that:

$$J(\phi(\xi,t)) = f(\phi(\xi,t),t) \text{ for } t > 0 \tag{2.21}$$

(assuming it is true at $t = 0$.)

where $J\left(\phi(\xi,t)\right)=\det\nabla\phi(\xi,t)$ is the Jacobian determinant of the transformation.

The transformation ϕ can be found by the following two steps.

Step 1: Find a vector field $V(\phi,t)$ such that:

$$\operatorname{div}_{\phi} V(\phi,t)=-\frac{\partial}{\partial t}\frac{1}{f(\phi,t)}=-\frac{\partial}{\partial t}g(\phi,t) \ ,$$

where $g\left(\phi(\xi,t),t\right)=\dfrac{1}{f\left(\phi(\xi,t),t\right)}$.

Step 2: Solve the ODE for the transformation $\phi(\xi,t)$:

$$\frac{\partial\phi(\xi,t)}{\partial t}= f(\phi,t)V(\phi,t)=\eta(\phi,t)$$

We can show that the $\phi(\xi,t)$ found by this way satisfies (2.21):

In order to prove this, let:

$$H=\left(\det\nabla\phi(\xi,t)\right)g\left(\phi(\xi,t),t\right)=\frac{J(\phi,t)}{f(\phi,t)}=Jg \quad .$$

If we can show $\dfrac{\partial H}{\partial t}=0$, then $H=\dfrac{J}{f}=\text{const}$.

Proof:

$$\frac{\partial H}{\partial t}=\frac{\partial J}{\partial t}g\left(\phi(\xi,t),t\right)+J\,\frac{\partial g\left(\phi(\xi,t),t\right)}{\partial t} \tag{2.22}$$

By Abel’s Lemma,

since $\dfrac{\partial}{\partial t}\left(\nabla_{\xi}\phi\right)=\nabla_{\xi}\left(\dfrac{\partial\phi}{\partial t}\right)=\left(\nabla_{\phi}\left(\dfrac{\partial\phi}{\partial t}\right)\right)\left(\nabla_{\xi}\phi\right)=\left(\nabla_{\phi}(\eta)\right)\left(\nabla_{\xi}\phi\right)$, we get

$$\frac{\partial J}{\partial t} = \frac{\partial}{\partial t}\left(\det \nabla\phi\right) = \left(\operatorname{trace}\left(\nabla_\phi \eta\left(\phi,t\right)\right)\right)\left(\det\nabla\phi\right)$$

$$= \left(\operatorname{div}_\phi \eta\left(\phi,t\right)\right)\left(\det\nabla\phi\right) = \left(\operatorname{div}_\phi\left(f\left(\phi,t\right)V\left(\phi,t\right)\right)\right)J$$

$$= \left[f\left(\phi,t\right)\operatorname{div}_\phi\left(V\left(\phi,t\right)\right) + \left\langle \nabla_\phi f\left(\phi,t\right), V\left(\phi,t\right)\right\rangle\right]J$$

$$= \left[-f\frac{\partial g}{\partial t} + \left\langle \nabla_\phi f, V\right\rangle\right]J$$

That is $$\frac{\partial J}{\partial t} = \left[-f\frac{\partial g}{\partial t} + \left\langle \nabla_\phi f, V\right\rangle\right]J \qquad (2.23)$$

Also we have $$\frac{\partial g\left(\phi,t\right)}{\partial t} = \left\langle \nabla_\phi g, \frac{\partial\phi}{\partial t}\right\rangle + \frac{\partial g}{\partial t} \qquad (2.24)$$

Plugging (2.23) and (2.24) into (2.22), we get:

$$\frac{\partial H}{\partial t} = \left[-f\frac{\partial g}{\partial t} + \left\langle \nabla_\phi f, V\right\rangle\right]Jg + J\left(\left\langle \nabla_\phi g, \frac{\partial\phi}{\partial t}\right\rangle + \frac{\partial g}{\partial t}\right)$$

$$= J\left[-fg\frac{\partial g}{\partial t} + \left\langle \nabla_\phi f, V\right\rangle g + \left\langle \nabla_\phi g, fV\right\rangle + \frac{\partial g}{\partial t}\right] \quad \text{(note:} \frac{\partial\phi}{\partial t} = fV \text{ is used)}$$

$$= J\left[-\frac{\partial g}{\partial t} + \left\langle \nabla_\phi f, V\right\rangle g + f\left\langle \nabla_\phi g, V\right\rangle + \frac{\partial g}{\partial t}\right] \quad \text{(note: } g = \frac{1}{f} \Rightarrow fg = 1 \text{ is used)}$$

$$= J\left[\left\langle\left(g\nabla_\phi f + f\nabla_\phi g\right), V\right\rangle\right] \quad \left(\text{note: } fg = 1 \Rightarrow \nabla(fg) = g\nabla f + f\nabla g = 0 \text{ is used}\right)$$

$$= J\left[\left\langle 0, V\right\rangle\right]$$

$$= 0$$

□

The numerical implementations for all the three cases are similar. The method discussed in case1 works after adjusting the right hand side of the equation corresponds to the certain case.

CHAPTER 3

3-D MULTI-BLOCK DEFORMATION METHOD

In this chapter we apply the deformation method to 3D multi-block structured grids. A 2D implementation can be found in [38].

3.1 Numerical Implementation

It is hard to apply structured grid on complex domains. A domain can be decomposed into different blocks so that we can take advantage of structured grid on each block. When we implement deformation method in multi-block setting, a monitor function is defined for the entire domain. After normalizing the monitor function for the whole domain [38], we implement deformation method by solving a Poisson equation on each block by using finite difference method. And then take the gradient on the solution to get the vector field for moving the grid points.

3.2 Numerical Example

In this example, a 3D back-step is decomposed into two blocks. The first block is a $[0,1]\times[0,2]\times[0,1]$ column. A 20×40×20 initial uniform grid is generated on it. The Second block is a $[1,2]\times[0,1]\times[0,1]$ cube. The initial uniform structured grid on it is 20×20×20. These two blocks have a common boundary at $x=1$, $y\in[0,1]$ and $z\in[0,1]$. (See figure 3.1). The initial grid is deformed into a moving grid concentrated around a ball of radius $r = 0.2$. The ball keeps the same radius and moves from the first block to the second block gradually through the interface of the two blocks.

Let l denote the time step. The coordinate values of the center of the moving ball (a, b, c) are defined as:

$$\begin{cases} a = 0.5 \\ b = 1.5 - 0.05l \\ c = 0.5 \end{cases} \quad \textit{when} \quad 0 \le l \le 20$$

$$\begin{cases} a = 0.5 + 0.05(l - 20) \\ b = 0.5 \\ c = 0.5 \end{cases} \quad \textit{when} \quad 20 < l \le 40$$

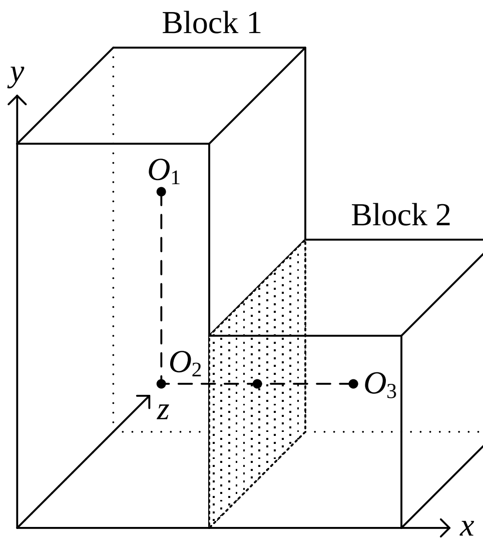

Figure 3.1 Two block of a 3D back-step

That means the center of the sphere starts at $O_1(0.5, 1.5, 0.5)$ and then drop to $O_2(0.5, 0.5, 0.5)$ when $0 \le l \le 20$. After that, when $20 < l \le 40$, the center keeps on moving through the interface of the two blocks and then reaches $O_3(1.5, 0.5, 0.5)$.

The computation includes two steps in our example:

Step 1: For $0 \le t \le 0.5$, we form a grid adapted to the sphere at $O_1(0.5, 1.5, 0.5)$ with $r = 0.2$.

Step 2: For $0 \le l \le 20$, the sphere goes to $O_2(0.5, 0.5, 0.5)$ and for $20 < l \le 40$ the sphere moves to $O_3(1.5, 0.5, 0.5)$ gradually (See figure 3.1).

Now, let us define a monitor function on the entire domain.

To define the monitor function, a level set function d is introduced, which vanishes on the moving ball.

$$d = (x-a)^2 + (y-b)^2 + (z-c)^2 - r^2 \qquad (r = 0.2)$$

Then, the monitor function for $0 \le t \le 0.5$ can be defined as:

$$\tilde{f} = \begin{cases} 1 & d < -0.05 \\ 1 - 2t + 2t(0.2 - 8d) & -0.05 \le d < 0 \\ 1 - 2t + 2t(0.2 + 8d) & 0 \le d < 0.05 \\ 1 & d \ge 0.05 \end{cases}$$

This is the monitor function before normalization. We find the normalized monitor function f by

$$f = \frac{\tilde{f} \int_\Omega \frac{1}{\tilde{f}} dA}{|\Omega|}.$$

After we obtain the normalized monitor function, we calculate the right hand side of the Poisson equation

$$\Delta\omega = -\frac{\partial}{\partial t}\left(\frac{1}{f}\right) \text{ in } \Omega$$

$$\frac{\partial \omega}{\partial n} = 0 \quad \text{on } \Gamma$$

by $$\frac{\partial}{\partial t}\left(\frac{1}{f}\right) = \frac{\dfrac{1}{f\left(\boldsymbol{x}, t+dt\right)} - \dfrac{1}{f\left(\boldsymbol{x}, t\right)}}{dt}.$$

After the right hand sides of the Poisson equation for both block 1 and block 2 are set up, now we discretize the Poisson equation using the adjacent six points. The finite difference representation of the Poisson equation is (See figure 3.2):

$$\frac{\omega_{i-1,j,k} - 2\omega_{i,j,k} + \omega_{i+1,j,k}}{\Delta x^2} + \frac{\omega_{i,j-1,k} - 2\omega_{i,j,k} + \omega_{i,j+1,k}}{\Delta y^2} + \frac{\omega_{i,j,k-1} - 2\omega_{i,j,k} + \omega_{i,j,k+1}}{\Delta z^2} = rhs_{i,j,k}$$

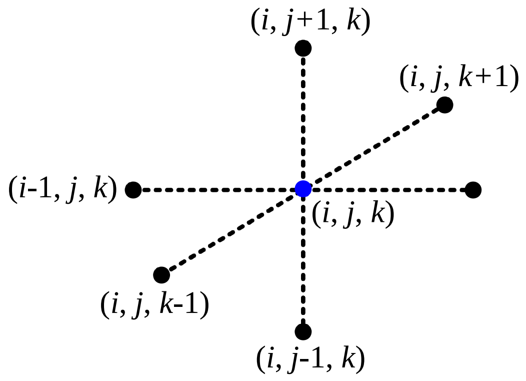

Figure 3.2 Finite difference using adjacent six points

Here, we assume $\Delta x = \Delta y = \Delta z = h$, then for all the interior points we have

$$\omega_{i,j,k} = \frac{1}{6}\left(\omega_{i-1,j,k} + \omega_{i+1,j,k} + \omega_{i,j-1,k} + \omega_{i,j+1,k} + \omega_{i,j,k-1} + \omega_{i,j,k+1} - h^2 \times rhs_{i,j,k}\right).$$

The resulting system of linear algebraic equations is then solved by using successive overrelaxation (SOR) method in the following two steps.

$$\tilde{\omega}_{i,j,k} = \frac{1}{6}\left(\omega^{new}_{i-1,j,k} + \omega^{old}_{i+1,j,k} + \omega^{new}_{i,j-1,k} + \omega^{old}_{i,j+1,k} + \omega^{new}_{i,j,k-1} + \omega^{new}_{i,j,k+1} - h^2 \times rhs_{i,j,k}\right)$$

$$\omega_{i,j,k}^{new} = (1-\lambda)\,\omega_{i,j,k}^{old} + \lambda\tilde{\omega}_{i,j,k}$$

On the boundary, Neumann boundary condition is implemented by $\frac{\partial\omega}{\partial n} = 0$ on Γ.

The boundary includes 8 corners, 12 edges and 6 faces for each block.

For corner 1 (See Figure 3.3):

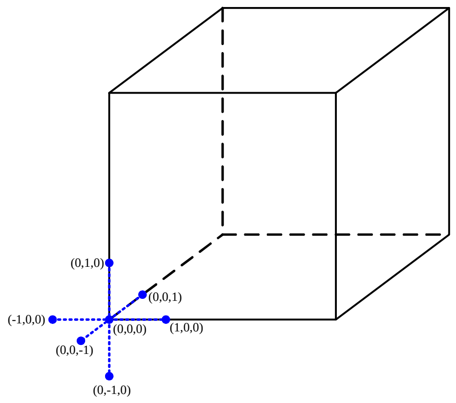

Figure 3.3 Corner 1

$$\frac{\omega_{1,0,0} - \omega_{-1,0,0}}{2\Delta x} = 0 \quad \Rightarrow \omega_{1,0,0} = \omega_{-1,0,0}$$

$$\frac{\omega_{0,1,0} - \omega_{0,-1,0}}{2\Delta y} = 0 \quad \Rightarrow \omega_{0,1,0} = \omega_{0,-1,0}$$

$$\frac{\omega_{0,0,1} - \omega_{0,0,-1}}{2\Delta z} = 0 \quad \Rightarrow \omega_{0,0,1} = \omega_{0,0,-1}$$

$$\begin{aligned}\omega_{0,0,0} &= \frac{1}{6}\left(\omega_{-1,0,0} + \omega_{1,0,0} + \omega_{0,-1,0} + \omega_{0,1,0} + \omega_{0,0,-1} + \omega_{0,0,1} - h^2 \times rhs_{0,0,0}\right) \\ &= \frac{1}{6}\left(2\omega_{1,0,0} + 2\omega_{0,1,0} + 2\omega_{0,0,1} - h^2 \times rhs_{0,0,0}\right)\end{aligned}$$

Other corner points are implemented in a similar way.

For edge 1 (See figure 3.4):

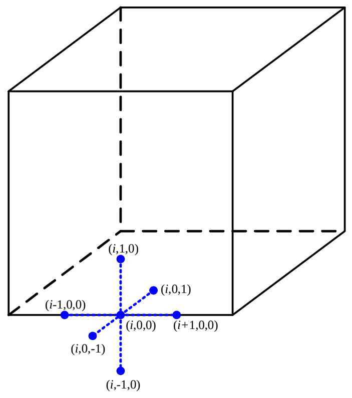

Figure 3.4 Edge 1

$$\frac{\omega_{i,1,0}-\omega_{i,-1,0}}{2\Delta y}=0 \Rightarrow \omega_{i,1,0}=\omega_{i,-1,0}$$

$$\frac{\omega_{i,0,1}-\omega_{i,0,-1}}{2\Delta z}=0 \Rightarrow \omega_{i,0,1}=\omega_{i,0,-1}$$

$$\begin{aligned}\omega_{i,0,0} &= \frac{1}{6}\left(\omega_{i-1,0,0}+\omega_{i+1,0,0}+\omega_{0,-1,0}+\omega_{0,1,0}+\omega_{0,0,-1}+\omega_{0,0,1}-h^2\times rhs_{0,0,0}\right)\\ &= \frac{1}{6}\left(\omega_{i-1,0,0}+\omega_{i+1,0,0}+\omega_{0,1,0}+\omega_{0,1,0}+\omega_{0,0,1}+\omega_{0,0,1}-h^2\times rhs_{0,0,0}\right)\\ &= \frac{1}{6}\left(\omega_{i-1,0,0}+\omega_{i+1,0,0}+2\omega_{0,1,0}+2\omega_{0,0,1}-h^2\times rhs_{i,0,0}\right)\end{aligned}$$

Other boundary edges are implemented in the similar way.

For face 1 (See figure 3.5):

$$\frac{\omega_{i,1,j}-\omega_{i,-1,j}}{2\Delta y}=0 \Rightarrow \omega_{i,1,j}=\omega_{i,-1,j}$$

$$\begin{aligned}\omega_{i,0,j} &= \frac{1}{6}\left(\omega_{i-1,0,j} + \omega_{i+1,0,j} + \omega_{i,-1,j} + \omega_{i,1,j} + \omega_{i,0,j-1} + \omega_{i,0,j+1} - h^2 \times rhs_{i,0,j}\right) \\ &= \frac{1}{6}\left(\omega_{i-1,0,j} + \omega_{i+1,0,j} + \omega_{i,1,j} + \omega_{i,1,j} + \omega_{i,0,j-1} + \omega_{i,0,j+1} - h^2 \times rhs_{i,0,j}\right) \\ &= \frac{1}{6}\left(\omega_{i-1,0,j} + \omega_{i+1,0,j} + 2\omega_{i,1,j} + \omega_{i,0,j-1} + \omega_{i,0,j+1} - h^2 \times rhs_{i,0,j}\right)\end{aligned}$$

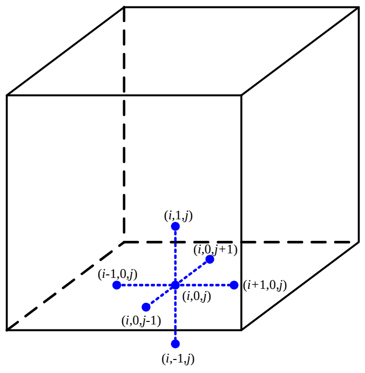

Figure 3.5 Face 1

Other boundary faces are implemented in the similar way.

Special cares need to be taken for common boundary points. Notice, part of the boundary of each block actually is not really the boundary for the whole domain. The common boundary (interface) of the two blocks (Shaded part of figure 3.1, includes: 4 corners, 4 edges and 1 face) is the part which we need to pass data to each other block. Neumann boundary condition does not apply here as for other boundary part. For the points on the interface, we have to use points from the neighbor block. For example, for $1 < j < my-1, 1 < k < mz-1$, when we calculate for block 1:

$$\omega A_{mx,j,k} = \frac{1}{6}\left(\omega A_{mx-1,j,k} + \omega B_{1,j,k} + \omega A_{i,j-1,k} + \omega A_{i,j+1,k} + \omega A_{i,j,k-1} + \omega A_{i,j,k+1} - h^2 \times rhs_{i,j,k}\right).$$

When we calculate for block 2:

$$\omega B_{0,j,k} = \frac{1}{6}\left(\omega A_{mx-1,j,k} + \omega B_{0,j,k} + \omega B_{i,j-1,k} + \omega B_{i,j+1,k} + \omega B_{i,j,k-1} + \omega B_{i,j,k+1} - h^2 \times rhs_{i,j,k}\right).$$

The 4 corners and 4 edges on the interface are implemented in the similar way by borrowing points from each block. For example:

$$\omega A_{mx,0,0} = \frac{1}{6}\left(2\omega A_{mx,0,0} + \omega B_{1,0,0} + \omega A_{mx-1,0,0} + \omega A_{mx,1,0} + \omega A_{mx,0,1} - h^2 \times rhs_{mx,0,0}\right)$$

$$\omega B_{0,0,0} = \frac{1}{6}\left(2\omega B_{0,0,0} + \omega A_{mx-1,0,0} + \omega B_{1,0,0} + \omega B_{0,1,0} + \omega B_{0,0,1} - h^2 \times rhs_{0,0,0}\right)$$

$$\omega A_{mx,0,k} = \frac{1}{6}\left(\omega A_{mx-1,0,k} + \omega B_{1,0,k} + \omega A_{mx,0,k+1} + \omega A_{mx,0,k-1} + \omega A_{mx,1,k} + \omega A_{mx,0,k} - h^2 \times rhs_{mx,0,0}\right)$$
$$\omega B_{0,0,k} = \frac{1}{6}\left(\omega B_{1,0,k} + \omega A_{mx,0,k} + \omega B_{0,0,k+1} + \omega B_{0,0,k-1} + \omega B_{0,1,k} + \omega B_{0,0,k} - h^2 \times rhs_{mx,0,0}\right)$$

Here, ωA is for values in block 1, ωB is for values in block 2. Other points are implemented in the similar way.

After boundary points are carefully taken care of. We solve the Poisson equation for each block.

Then we compute the vector field V, i.e. the nodal velocities by:

$$V = \nabla \omega$$

At last, the new position of grid points are computed by the following deformation ODE.

$$\frac{\partial \phi}{\partial t} = f\,V$$

The results are shown through Figure 3.6 to Figure 3.23. The implementation of more blocks can be done in the similar way as this two-block example.

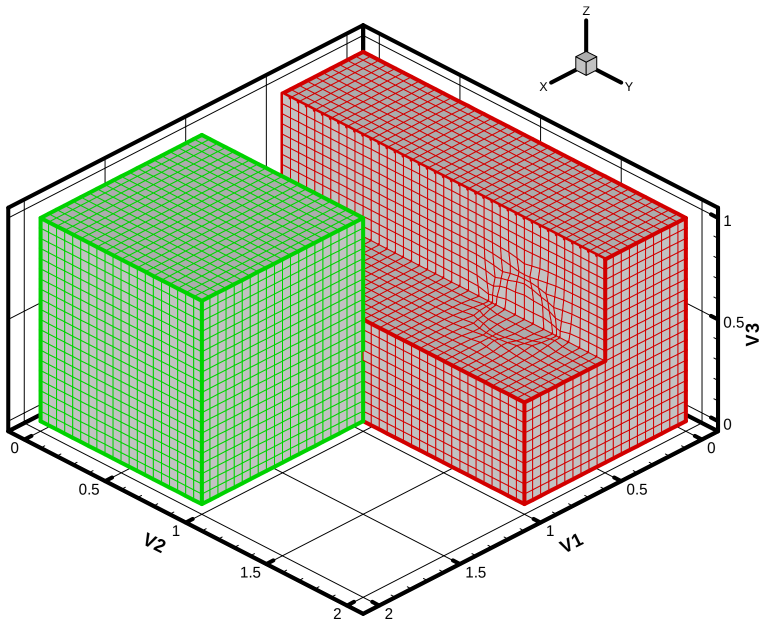

Figure 3.6 3D plot for time step $l = 0$ (Cutaway plot)

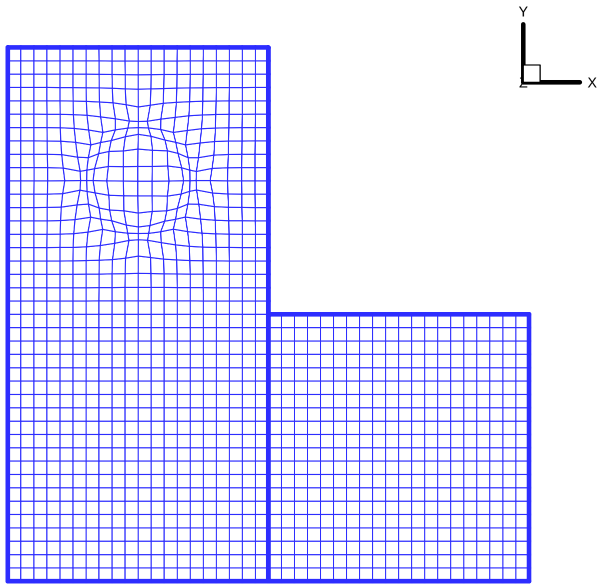

Figure 3.7 A slice extract at constant $z = 0.5$ for time step $l = 0$

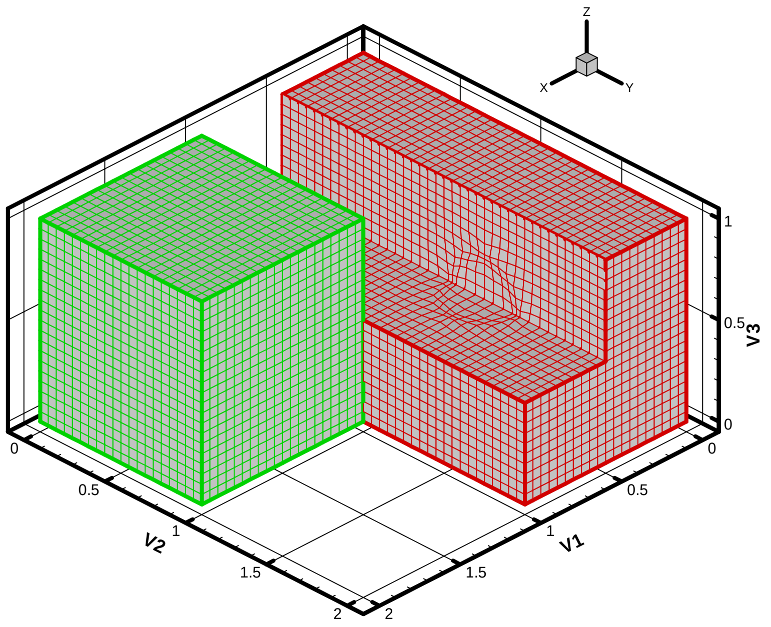

Figure 3.8 3D plot for time step $l = 5$ (Cutaway plot)

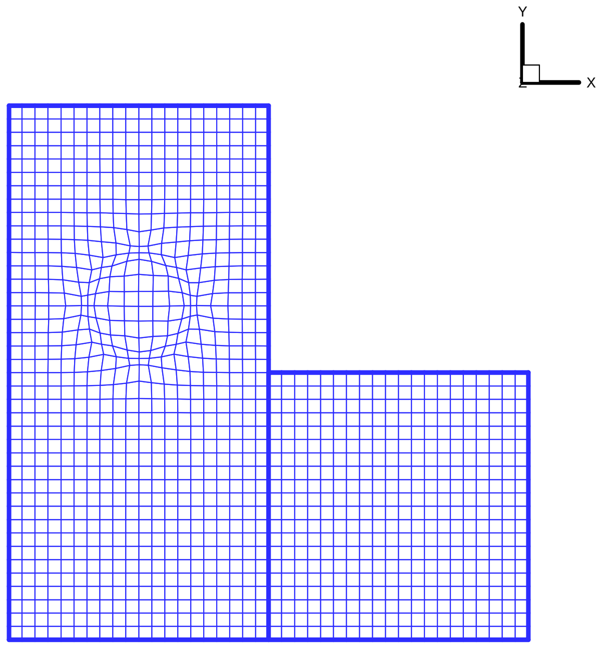

Figure 3.9 A slice extract at constant $z = 0.5$ for time step $l = 5$

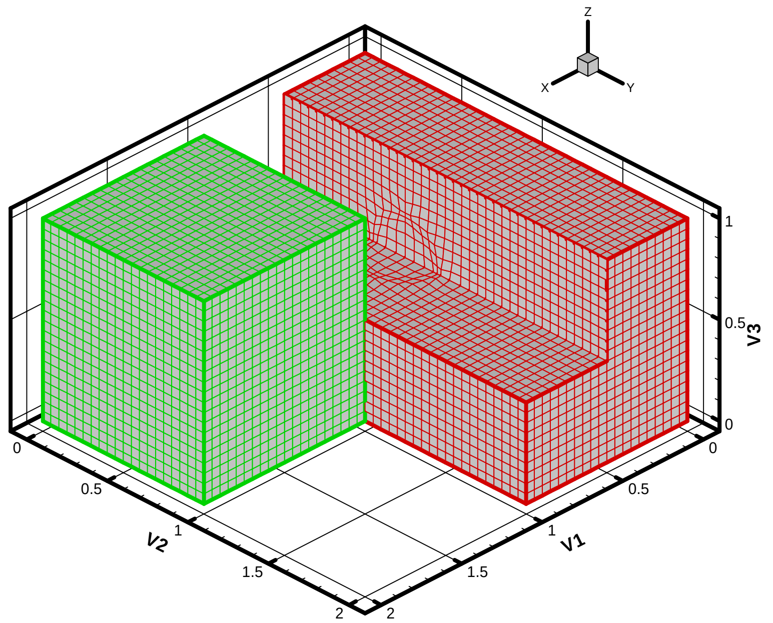

Figure 3.10 3D plot for time step $l = 15$ (Cutaway plot)

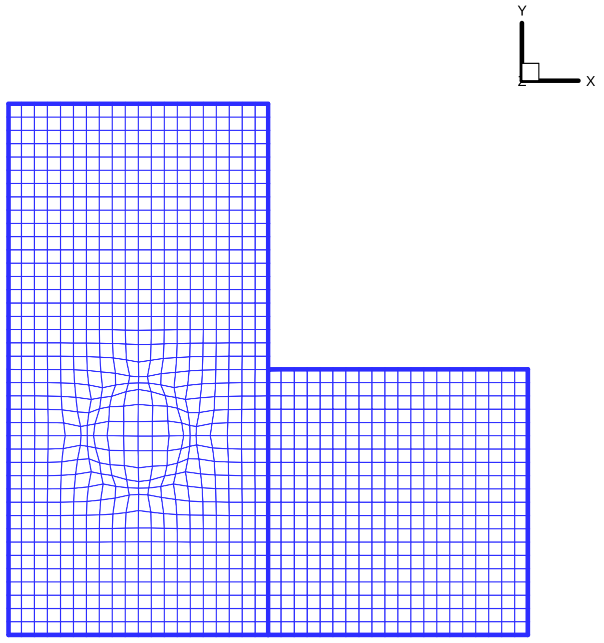

Figure 3.11 A slice extract at constant $z = 0.5$ for time step $l = 15$

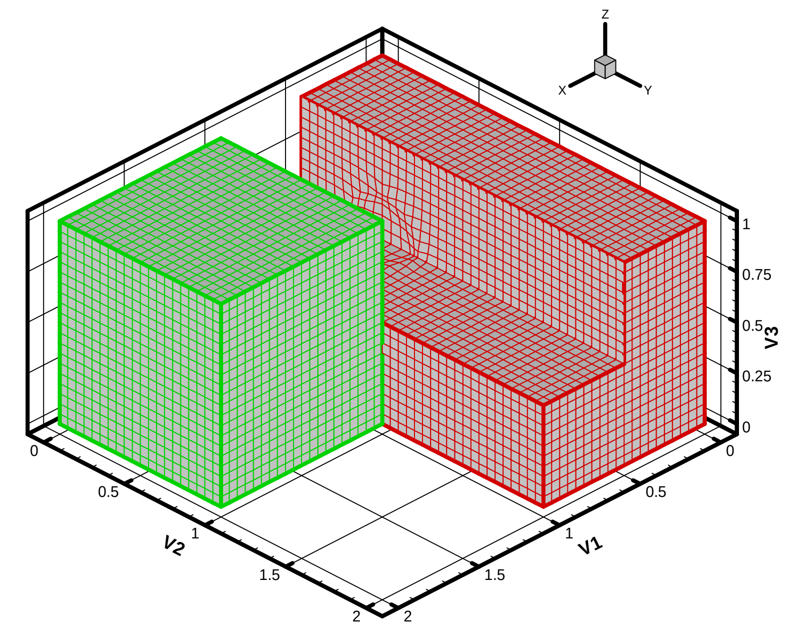

Figure 3.12 3D plot for time step $l = 20$ (Cutaway plot)

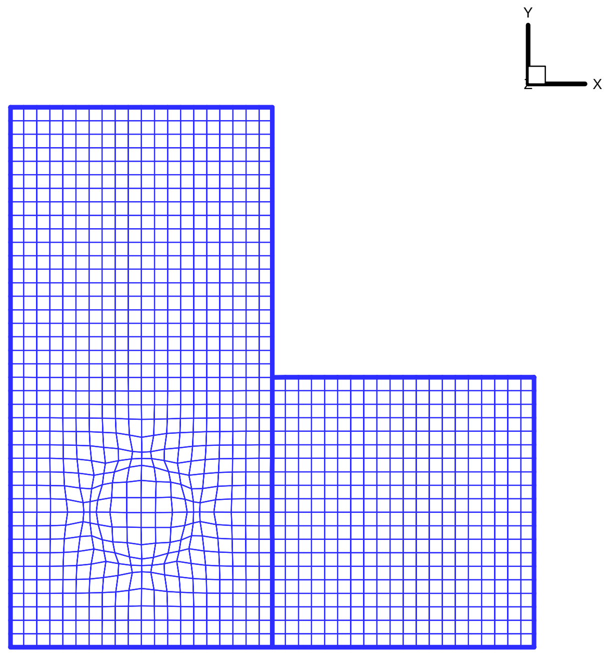

Figure 3.13 A slice extract at constant $z = 0.5$ for time step $l = 20$

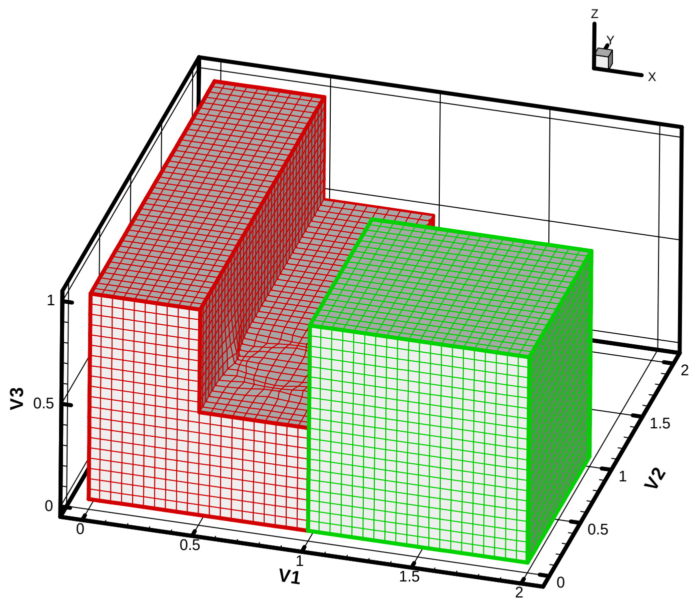

Figure 3.14 3D plot for time step $l = 25$ (Cutaway plot)

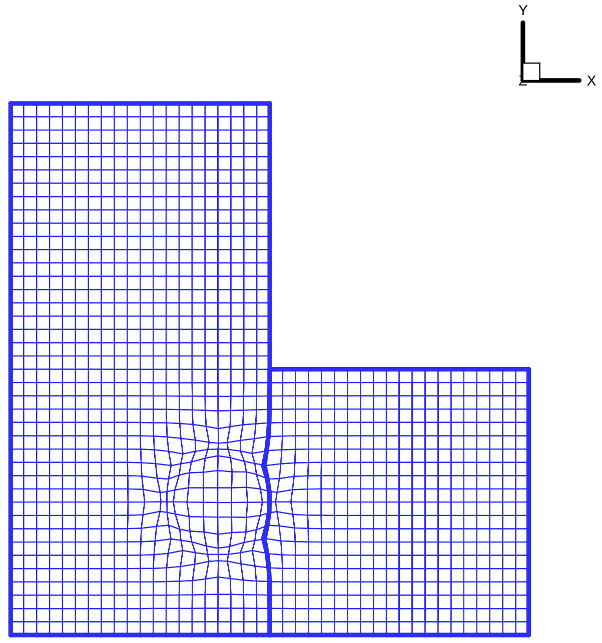

Figure 3.15 A slice extract at constant $z = 0.5$ for time step $l = 25$

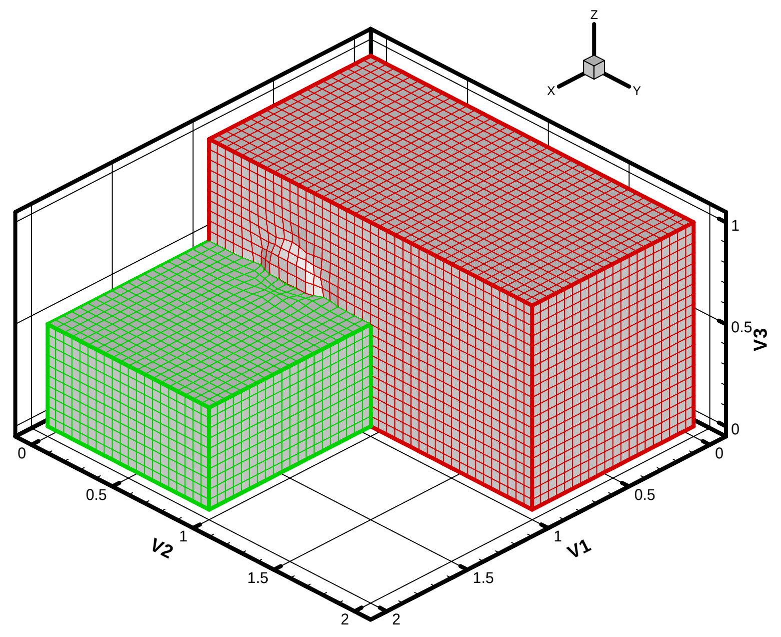

Figure 3.16 3D plot for time step $l = 28$ (Cutaway plot)

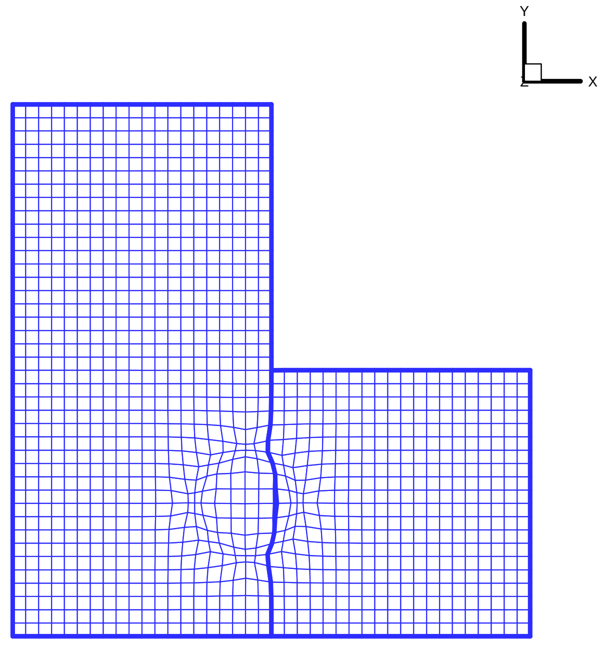

Figure 3.17 A slice extract at constant $z = 0.5$ for time step $l = 28$

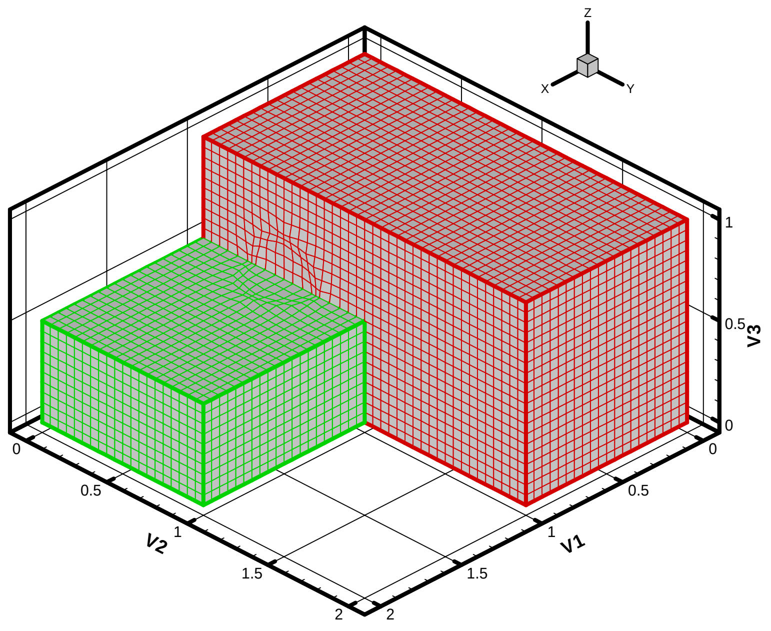

Figure 3.18 3D plot for time step $l = 30$ (Cutaway plot)

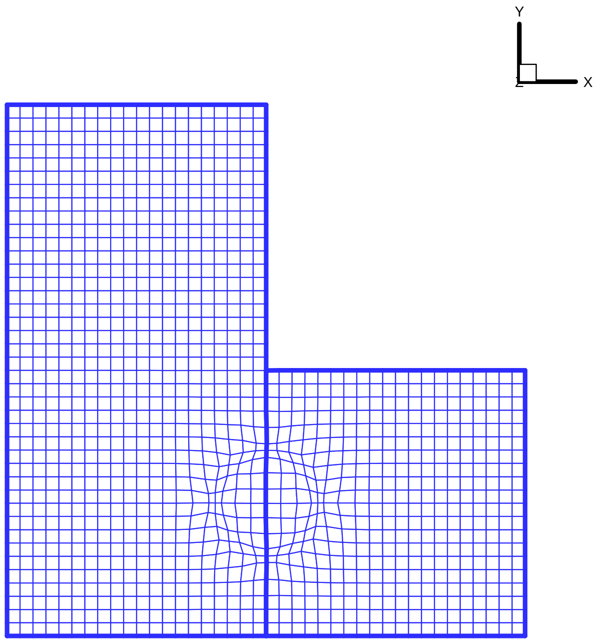

Figure 3.19 A slice extract at constant $z = 0.5$ for time step $l = 30$

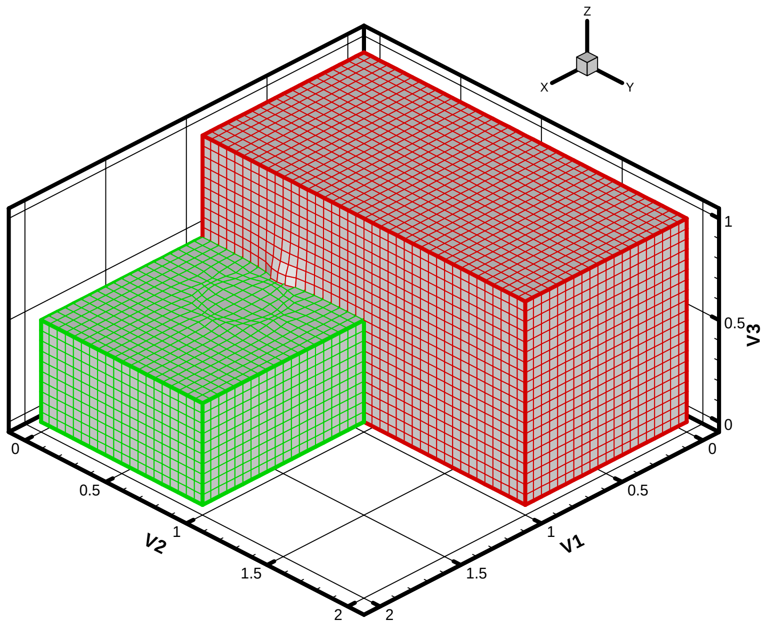

Figure 3.20 3D plot for time step $l = 35$ (Cutaway plot)

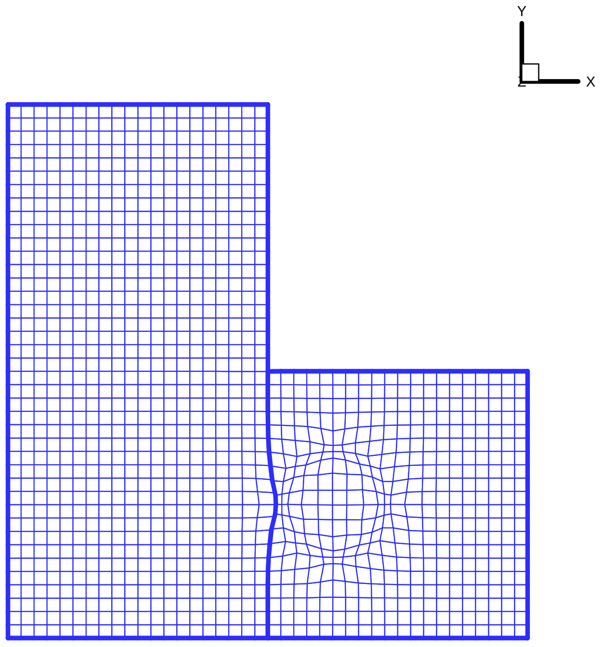

Figure 3.21 A slice extract at constant $z = 0.5$ for time step $l = 35$

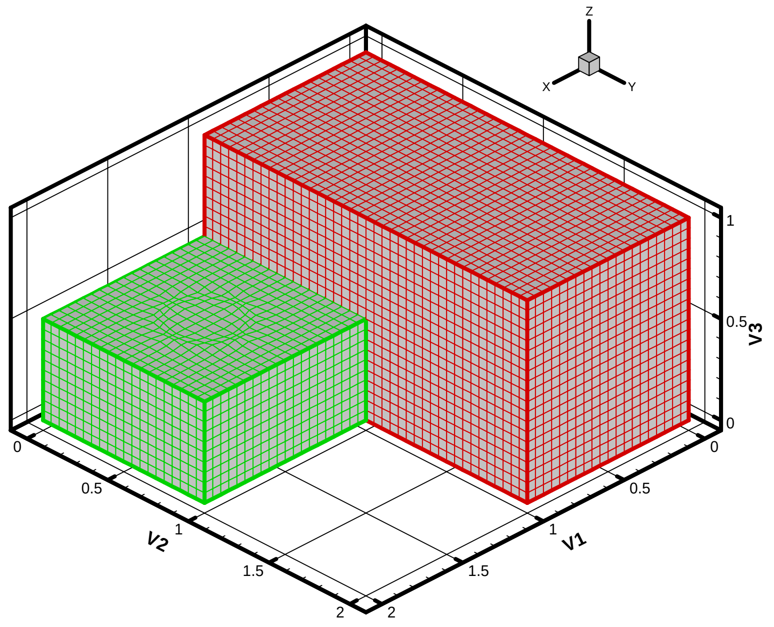

Figure 3.22 3D plot for time step $l = 40$ (Cutaway plot)

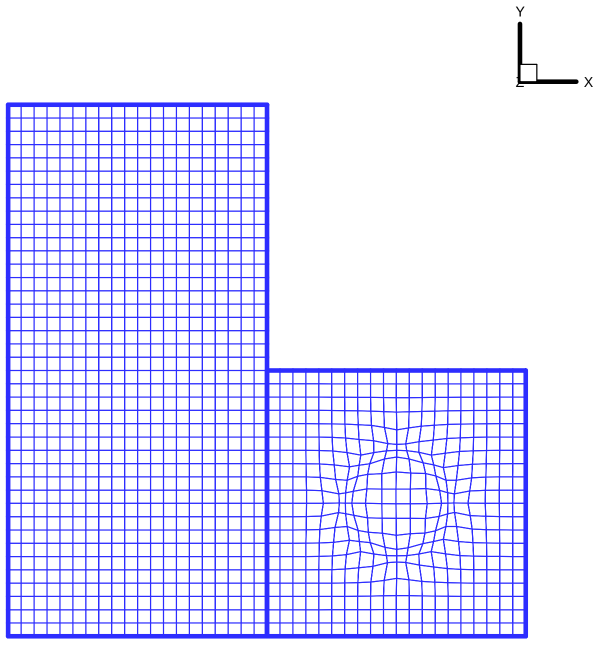

Figure 3.23 A slice extract at constant $z = 0.5$ for time step $l = 40$

CHAPTER 4

RECONSTRUCTION OF TRANSFORMATIONS

In this chapter we'll explore a new idea which may have applications to image processing using div-curl system of equations. The idea comes from the implementation of deformation method for finding transformations. Notice that for all the three cases in chapter one, the first step is to find a vector field by solving a divergence equation with different right hand side. After we add an equation of curl of the vector field, we can set up a div-curl system of equation. The least-square finite element method is a good way to solve it [42]. Now let's think in the opposite direction. For a transformation given on a uniform initial grid we can find out the divergence and curl at each point. Thus we can set up the div-curl system of equations for each point. Solving this system we can reconstruct the given transformation. This idea can apply to the construction of any differentiable and invertible transformations.

This idea may apply to image registration, which is the process of establishing point-by-point correspondence between two images of a scene. This process is needed in various computer vision applications. Sets of data acquired by sampling the same scene or object at different times, or from different perspectives, will be in different coordinate systems. Image registration is the process of transforming the different sets of data into one coordinate system. Registration is necessary in order to be able to compare or integrate the data obtained from different measurements. More research needs to be done to apply this method of reconstructing transformation to the field of image registration.

4.1 Div-curl system

We will first take a look at the div-curl system. Let's discuss the three-dimensional case. Let D be an open bounded domain in $\mathbb{R}^3$ with a piecewise smooth boundary $\Gamma = \Gamma_1 \bigcup \Gamma_2$. (x, y, z) denotes a point in D. $F = P\vec{i} + Q\vec{j} + R\vec{k}$ is a vector field in D. $\vec{n}$ is the unit outward normal vector of the boundary. Then the 3D div-curl system of equations is:

$$\begin{cases} divF = \alpha & in\, D \\ curlF = \vec{\beta} & in\, D \\ \vec{n}\cdot F = 0 & on\, \Gamma_1 \\ \vec{n}\times F = 0 & on\, \Gamma_2 \end{cases} \tag{4.1}$$

where $\vec{\beta} = \beta_1\vec{i} + \beta_2\vec{j} + \beta_3\vec{k}$

Our purpose is to solve for P, Q, R, for a total of three unknowns. But we have four scalar equations in this system. So, it appears that this system is 'overdetermined'. Let's reconsider this system by introducing a dummy variable θ as in [44], where $\theta \equiv 0$ in D and $\theta = 0$ on Γ_1 so that the system becomes:

$$\begin{cases} divF = \alpha & in\, D \\ \nabla\theta + curlF = \vec{\beta} & in\, D \\ \vec{n}\cdot F = 0 & on\, \Gamma_1 \\ \theta = 0 & on\, \Gamma_1 \\ \vec{n}\times F = 0 & on\, \Gamma_2 \end{cases} \tag{4.2}$$

We can show that system (4.2) is equivalent to system (4.1). Detailed proof can be found in [42]. Notice, system (4.2) is a system with four unknowns and four equations.

In Cartesian coordinates, we have:

$$curl\,F = \left(\frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z}\right)\vec{i} + \left(\frac{\partial P}{\partial z} - \frac{\partial R}{\partial x}\right)\vec{j} + \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right)\vec{k}$$

$$\nabla\theta = \frac{\partial\theta}{\partial x}\vec{i} + \frac{\partial\theta}{\partial y}\vec{j} + \frac{\partial\theta}{\partial z}\vec{k}$$

$$div\,F = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z}$$

So system (4.2) can be written as:

$$\begin{cases} \frac{\partial\theta}{\partial x} + \frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z} = \beta_1 \\ \frac{\partial\theta}{\partial y} + \frac{\partial P}{\partial z} - \frac{\partial R}{\partial x} = \beta_2 \\ \frac{\partial\theta}{\partial z} + \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = \beta_3 \\ \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} = \alpha \end{cases} \tag{4.3}$$

Define $\tilde{F} = \begin{pmatrix} P \\ Q \\ R \\ \theta \end{pmatrix}$ and $\tilde{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \alpha \end{pmatrix}$, then this system can be written in a matrix form:

$A_0\tilde{F} + A_1\frac{\partial\tilde{F}}{\partial x} + A_2\frac{\partial\tilde{F}}{\partial y} + A_3\frac{\partial\tilde{F}}{\partial z} = \tilde{\beta}$ where

$$A_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad A_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \qquad A_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

For any nonzero triplets (x, y, z), the characteristic polynomial for system (4.3) is

$$\det(A_1 x + A_2 y + A_3 z) = \det \begin{pmatrix} 0 & -z & y & x \\ z & 0 & -x & y \\ -y & x & 0 & z \\ x & y & z & 0 \end{pmatrix} = \left(x^2 + y^2 + z^2\right)^2 \neq 0$$

Thus, system (4.2) is elliptic and properly determined. And so is system (4.1).

Least-square finite element method is a good way to solve the div-curl system. Detailed analysis can be found in [42]. The numerical implementation procedures can be found in [40]. Here we'll take a glance at it.

4.2 Least-square FEM

Let's consider the linear boundary-value problem:

$$Au = f \text{ in } \Omega$$

$$Bu = g \text{ on } \Gamma \tag{4.4}$$

Where $Au = \sum_{i=1}^{n_d} A_i \frac{\partial u}{\partial x_i} + A_0 u$ ($n_d = 2$ for 2D, $n_d = 3$ for 3D). B is a boundary operator. f and g are given vector-valued functions. u is a vector with m unknown functions of $\boldsymbol{x}\left(x_1, \cdots x_{n_d}\right)$.

$$u = \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_m \end{pmatrix}, \; f = \begin{pmatrix} f_1 \\ f_2 \\ \cdots \\ f_{n_d} \end{pmatrix}, \; g = \begin{pmatrix} g_1 \\ g_2 \\ \cdots \\ g_{n_d} \end{pmatrix}$$

Let's define the residual as $R = Au - f$, then if $R = 0$ we get the exact solution for u . The least-square finite element method is to minimize R in a least-square sense, that is, to minimize the following functional:

$$I\left(v\right) = \left\|R\right\|_0^2 = \int_\Omega \left(Av - f\right)^2 d\omega .$$

A necessary condition for u to minimize $I\left(v\right)$ is :

$$\lim_{t \to 0} \frac{d}{dt} I\left(u + tv\right) = 0 \; .$$

Since

$$I\left(u+tv\right) = \int_\Omega \left[A\left(u+tv\right) - f\right]^2 d\omega$$

$$= \int_\Omega \left[\left(Au\right)^2 + \left(Av\right)^2 t^2 + f^2 + 2\left(Au\right)\left(Av\right)t - 2\left(Av\right) ft - 2\left(Au\right) f\right] d\omega,$$

we have

$$\lim_{t\to 0}\frac{d}{dt}I\left(u+tv\right)=\lim_{t\to 0}\int_{\Omega}2\left[\left(Av\right)^{2}t+\left(Au\right)\left(Av\right)-\left(Av\right)f\right]d\omega$$

$$=\int_{\Omega}2\left[\left(Au\right)\left(Av\right)-\left(Av\right)f\right]d\omega$$

$$=0.$$

Thus

$$\int_{\Omega}\left[\left(Au\right)\left(Av\right)\right]d\omega=\int_{\Omega}\left[\left(Av\right)f\right]d\omega .$$

That is

$$\left(Au,Av\right)=\left(f,Av\right). \tag{4.5}$$

This is the variational principle of equation (4.4).

In finite element, we discritize the domain into elements and then introduce finite element basis. Let φ_j be the element shape function, we write the expansion of the unknown variables in each element as

$$u_h^e\left(\boldsymbol{x}\right)=\sum_{j=1}^{N_n}\varphi_j\left(\boldsymbol{x}\right)\begin{pmatrix}u_1\\u_2\\\cdots\\u_m\end{pmatrix}_j \tag{4.6}$$

where N_n is the number of nodes for one element.

Introducing (4.6) into (4.5) we get a linear system of algebraic equations:

$$KU=F \tag{4.7}$$

Here

$$K_e=\int_{\Omega}\left(A\varphi_1,A\varphi_2,\cdots,A\varphi_{N_n}\right)^{T}\left(A\varphi_1,A\varphi_2,\cdots,A\varphi_{N_n}\right)d\Omega$$

$$F_e = \int_\Omega \left(A\varphi_1, A\varphi_2, \cdots, A\varphi_{N_n} \right)^T f d\Omega$$

are the element matrices used to assemble the global matrix K and F .

4.3 Solve Div-Curl System

Let’s take a look at the definition of divergence and curl of a vector field first.

If $\boldsymbol{V} = P\vec{i} + Q\vec{j} + R\vec{k}$ is a vector field on $\mathbb{R}^3$ and the partial derivatives of $P(x, y, z)$, $Q(x, y, z)$ and $R(x, y, z)$ all exist, then

$$div\boldsymbol{V} = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z}$$

$$curl\boldsymbol{V} = \left(\frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z} \right)\vec{i} + \left(\frac{\partial P}{\partial z} - \frac{\partial R}{\partial x} \right)\vec{j} + \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right)\vec{k}$$

In 2D, all the terms related to R and z vanish. So we have

$$div\boldsymbol{V} = \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}$$

$curl\boldsymbol{V} = \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right)\vec{k}$, where $\vec{k}$ is the unit outward normal vector, usually denoted as $\vec{n}$.

The matrix form of the div-curl system can be written as:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial P}{\partial x} \\ \frac{\partial Q}{\partial x} \\ \frac{\partial R}{\partial x} \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial P}{\partial y} \\ \frac{\partial Q}{\partial y} \\ \frac{\partial R}{\partial y} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial P}{\partial z} \\ \frac{\partial Q}{\partial z} \\ \frac{\partial R}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \\ \frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z} \\ \frac{\partial P}{\partial z} - \frac{\partial R}{\partial x} \\ \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \end{pmatrix}$$

In 3D, linear hexahedral elements are used and the finite element approximation at each hexahedral is given by

$$V_h^e(x) = \sum_{i=1}^{8} \left[\varphi_i \begin{pmatrix} p_i \\ q_i \\ r_i \end{pmatrix} \right]$$

where p_i, q_i, r_i are the nodal values at the ith node of the hexahedral element and φ_i 's are the shape functions.

The element matrices used to assemble the algebraic system $KV = F$ are

$$K_e = \int_{\Omega_e} \begin{pmatrix} (A\varphi_1)^T A\varphi_1 & \cdots & (A\varphi_1)^T A\varphi_8 \\ \vdots & \ddots & \vdots \\ (A\varphi_8)^T A\varphi_1 & \cdots & (A\varphi_8)^T A\varphi_8 \end{pmatrix} d\Omega$$

$$F_e = \int_{\Omega_e} \begin{pmatrix} (A\varphi_1)^T \boldsymbol{f} \\ \vdots \\ (A\varphi_8)^T \boldsymbol{f} \end{pmatrix} d\Omega$$

where $$A\varphi_i = \begin{pmatrix} \frac{\partial \varphi_i}{\partial x} & \frac{\partial \varphi_i}{\partial y} & \frac{\partial \varphi_i}{\partial z} \\ 0 & -\frac{\partial \varphi_i}{\partial z} & \frac{\partial \varphi_i}{\partial y} \\ \frac{\partial \varphi_i}{\partial z} & 0 & -\frac{\partial \varphi_i}{\partial x} \\ -\frac{\partial \varphi_i}{\partial y} & \frac{\partial \varphi_i}{\partial x} & 0 \end{pmatrix} \quad \text{for} \quad i = 1, 2, \ldots, 8.$$

and $$\boldsymbol{f} = \begin{pmatrix} \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \\ \frac{\partial R}{\partial y} - \frac{\partial Q}{\partial z} \\ \frac{\partial P}{\partial z} - \frac{\partial R}{\partial x} \\ \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \end{pmatrix}$$, calculated from the given transformation.

4.4 Numerical Examples

Numerically, if we set the position array for the given transformation as $xn(i,j,k)$, $yn(i,j,k)$, $zn(i,j,k)$, and the array for our initial grid as $x(i,j,k)$, $y(i,j,k)$, $z(i,j,k)$, then

$$\frac{\partial P}{\partial x} = \frac{xn(i+1,j,k) - xn(i-1,j,k)}{x(i+1,j,k) - x(i-1,j,k)}$$

$$\frac{\partial P}{\partial y} = \frac{xn(i,j+1,k) - xn(i,j-1,k)}{y(i,j+1,k) - y(i,j-1,k)}$$

$$\frac{\partial P}{\partial z} = \frac{xn(i,j,k+1) - xn(i,j,k-1)}{z(i,j,k+1) - z(i,j,k-1)}$$

$$\frac{\partial Q}{\partial x} = \frac{yn(i+1,j,k) - yn(i-1,j,k)}{x(i+1,j,k) - x(i-1,j,k)}$$

$$\frac{\partial Q}{\partial y} = \frac{yn(i,j+1,k) - yn(i,j-1,k)}{y(i,j+1,k) - y(i,j-1,k)}$$

$$\frac{\partial Q}{\partial z} = \frac{yn(i,j,k+1) - yn(i,j,k-1)}{z(i,j,k+1) - z(i,j,k-1)}$$

$$\frac{\partial R}{\partial x} = \frac{zn(i+1,j,k) - zn(i-1,j,k)}{x(i+1,j,k) - x(i-1,j,k)}$$

$$\frac{\partial R}{\partial y} = \frac{zn(i,j+1,k) - zn(i,j-1,k)}{y(i,j+1,k) - y(i,j-1,k)}$$

$$\frac{\partial R}{\partial z} = \frac{zn(i,j,k+1) - zn(i,j,k-1)}{z(i,j,k+1) - z(i,j,k-1)}$$

Following are some of the numerical examples.

We define $dxyz = \max\left(\left|XN_i - X_i\right|\right)$ ($i = 1, \ldots$ nmax).

Here $\left|XN_i - X_i\right| = \sqrt{\left(xn_i - x_i\right)^2 + \left(yn_i - y_i\right)^2 + \left(zn_i - z_i\right)^2}$ is the distance between each corresponding pair of points of the given and the reconstructed transformations. And nmax is the maximum number of nodes. $dxyz$ is used to measure the accuracy of our reconstruction method.

The grid size of the following examples are 64×64 over the unit square $[0,1] \times [0,1]$ for 2D and $40 \times 40 \times 40$ over the unit cube $[0,1] \times [0,1] \times [0,1]$ for 3D. That means the distance between adjacent points in the uniform grid is $\frac{1}{64}$=0.015625 for 2D and $\frac{1}{40}$=0.025 for 3D. We'll compare the $dxyz$ in each case with these to have the sense of how accurate we are.

4.4.1 2D examples

Example 1: A unit square with a sine curve. The results are shown in Figure 4.1 to Figure 4.4. $dxyz = 1.099 \times 10^{-3}$

Example 2: A unit square with a rectangle and an arc. The results are shown in Figure 4.5 to Figure 4.8. $dxyz = 1.122 \times 10^{-2}$

Example 3: A unit square with a circle. The results are shown in Figure 4.9 to Figure 4.12. $dxyz = 1.142 \times 10^{-3}$

4.4.2 3D examples

Example 4: A unit cube with a ball inside. The results are shown in Figure 4.13 to Figure 4.16. $dxyz = 4.225 \times 10^{-3}$

Example 5: A unit cube with an ellipsoid inside. The results are shown in Figure 4.17 to Figure 4.28. $dxyz = 4.162 \times 10^{-3}$

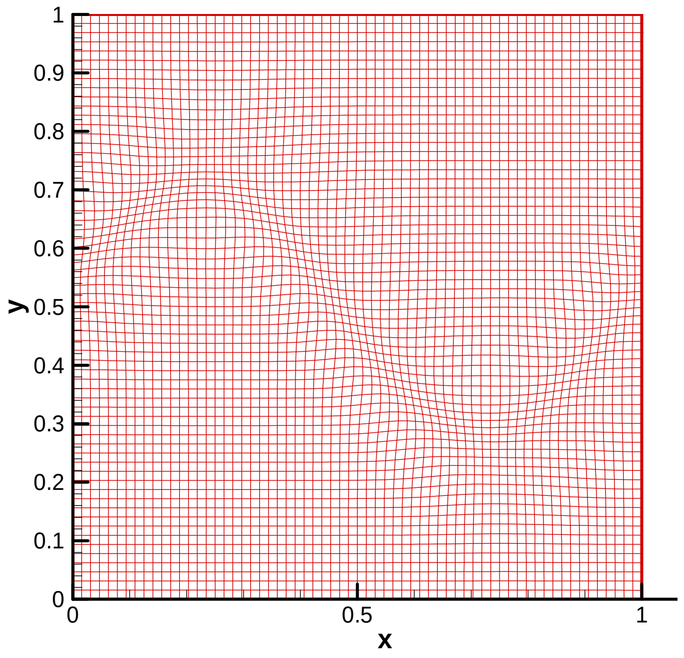

Figure 4.1 Example 4.1 Given transformation with a Sine curve

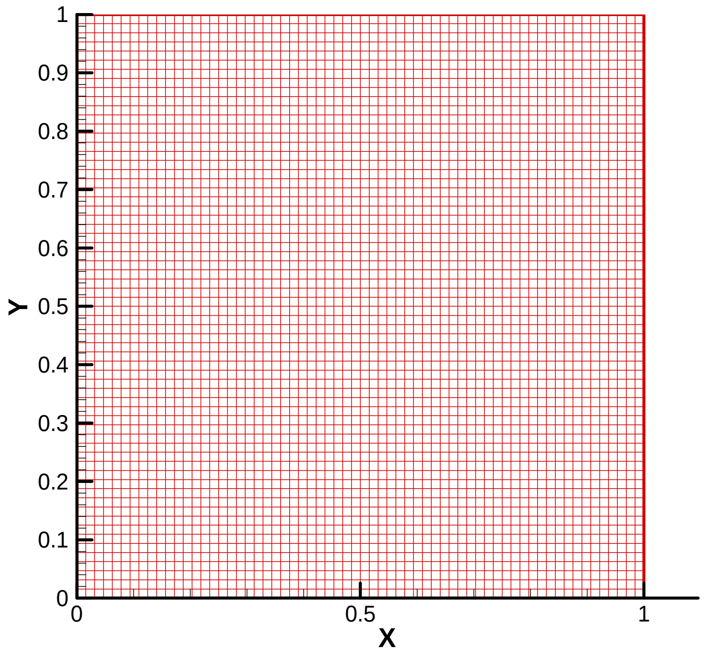

Figure 4.2 Example 4.1 Reconstruction at time step $t = 0$

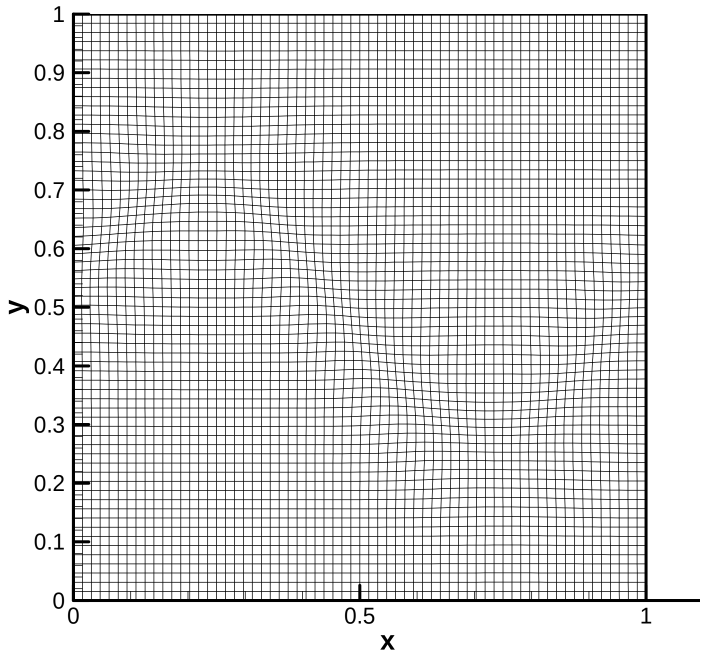

Figure 4.3 Example 4.1 Reconstruction at time step $t = 5$

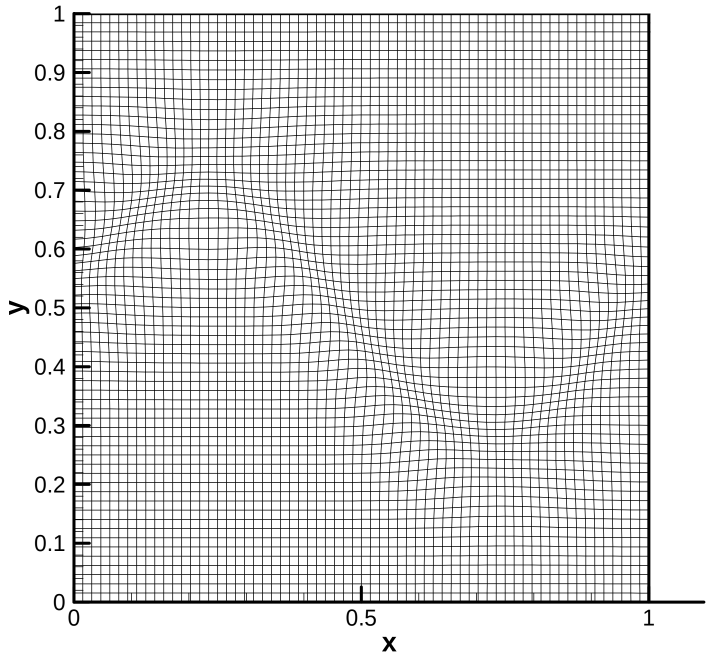

Figure 4.4 Example 4.1 Reconstruction at time step $t = 10$

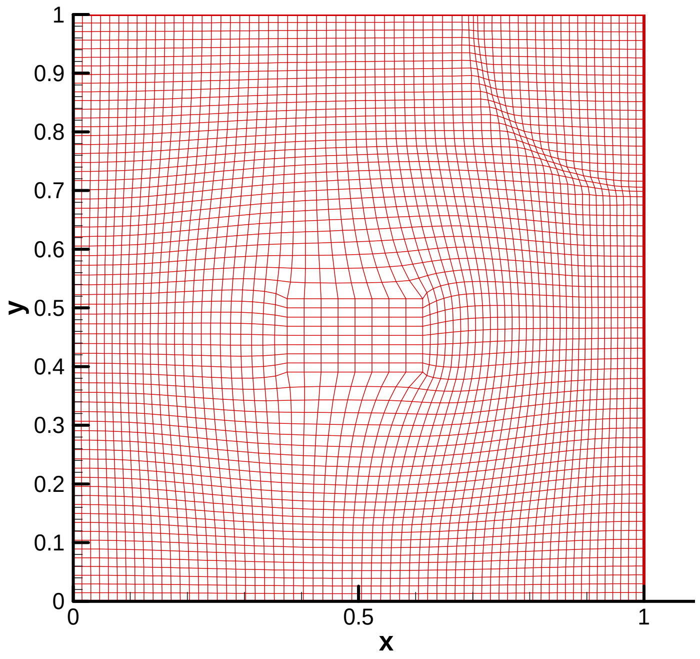

Figure 4.5 Example 4.2 Given transformation with a rectangle and an arc

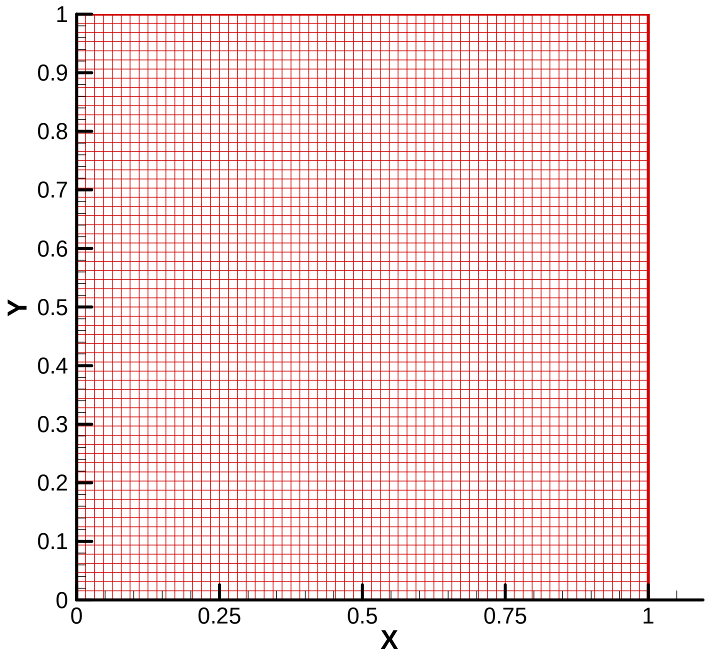

Figure 4.6 Example 4.2 Reconstruction at time step $t = 0$

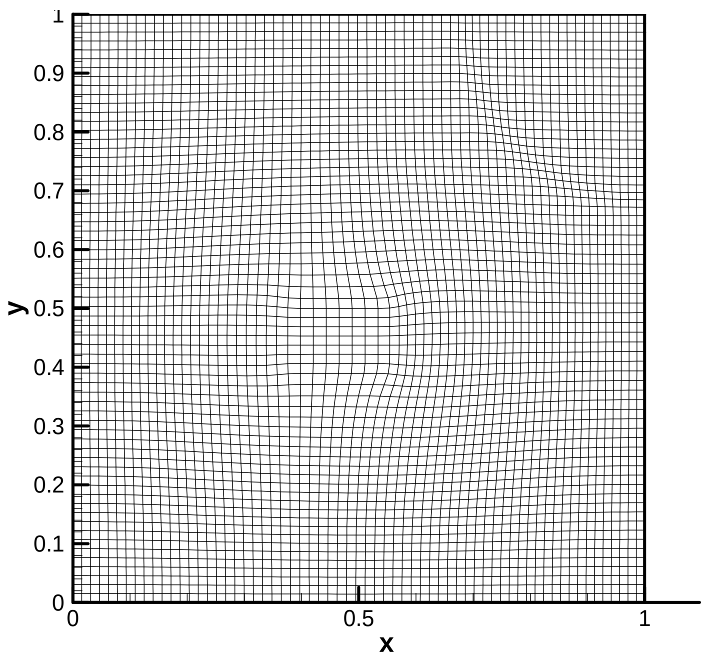

Figure 4.7 Example 4.2 Reconstruction at time step $t = 5$

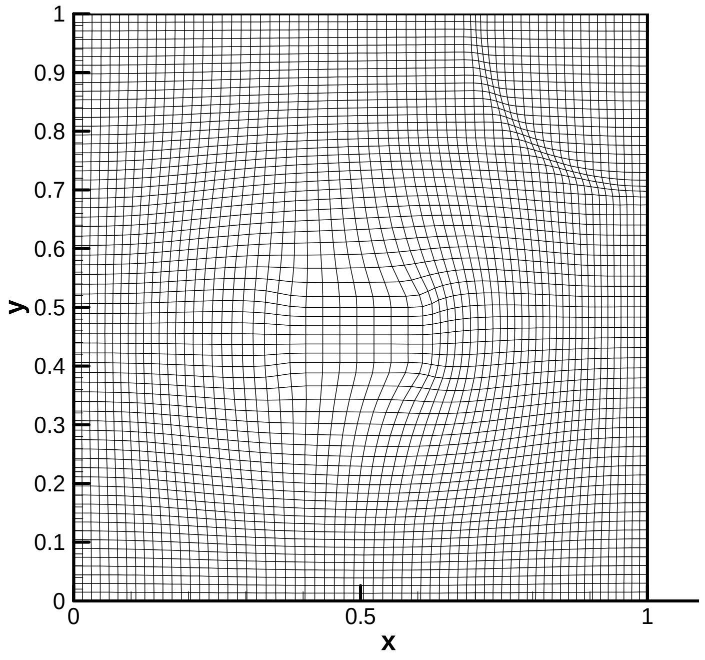

Figure 4.8 Example 4.2 Reconstruction at time step $t = 10$

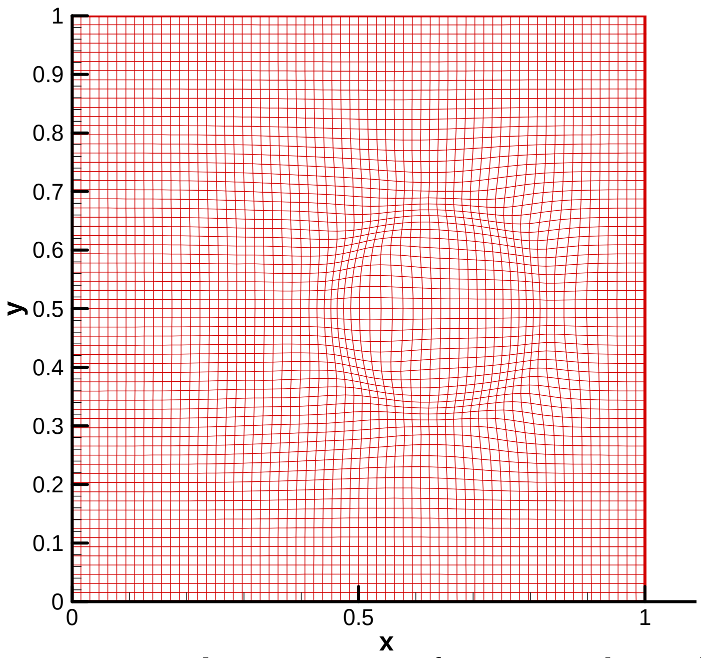

Figure 4.9 Example 4.3 Given transformation with a circle

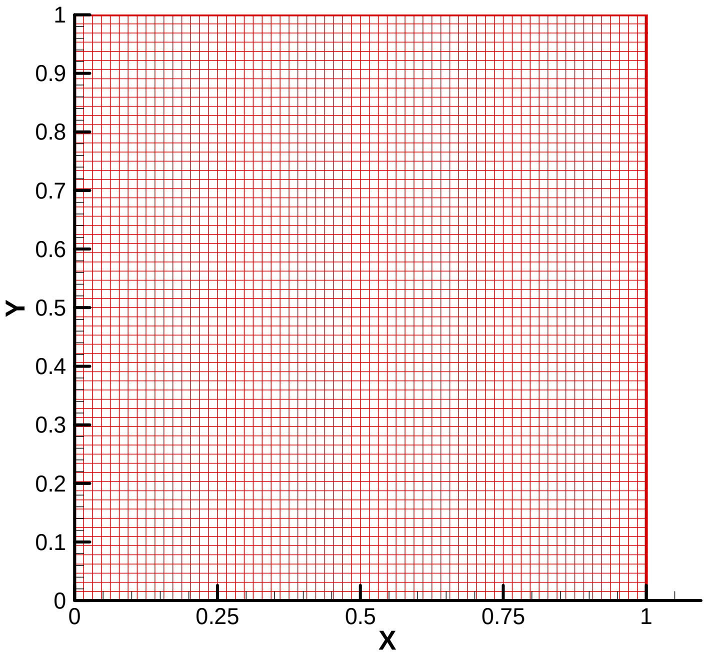

Figure 4.10 Example 4.3 Reconstruction at time step $t = 0$

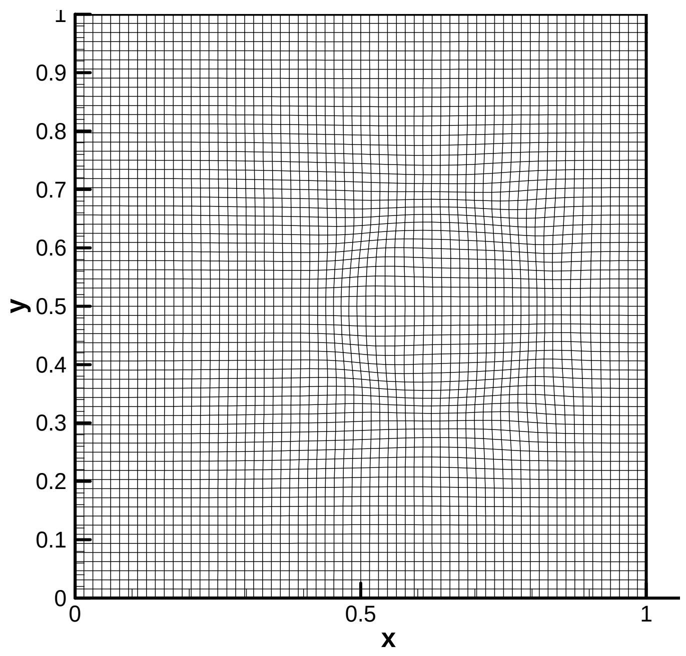

Figure 4.11 Example 4.3 Reconstruction at time step $t = 5$

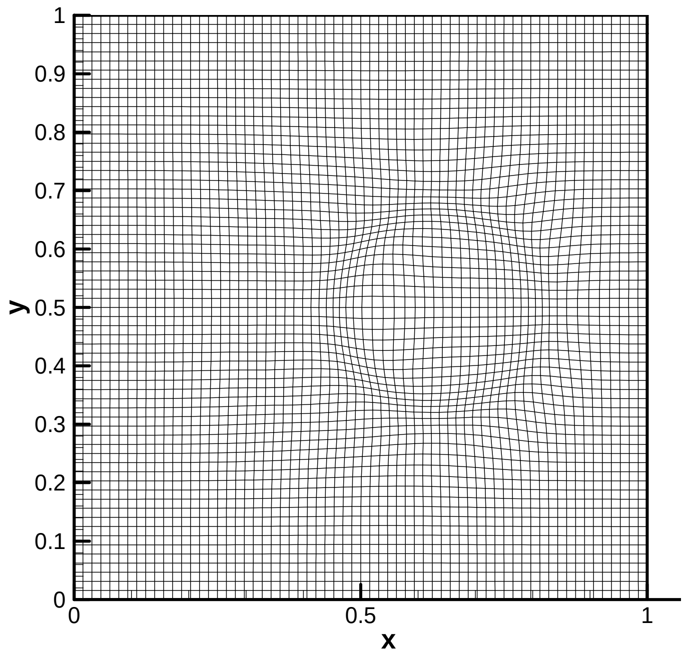

Figure 4.12 Example 4.3 Reconstruction at time step $t = 10$

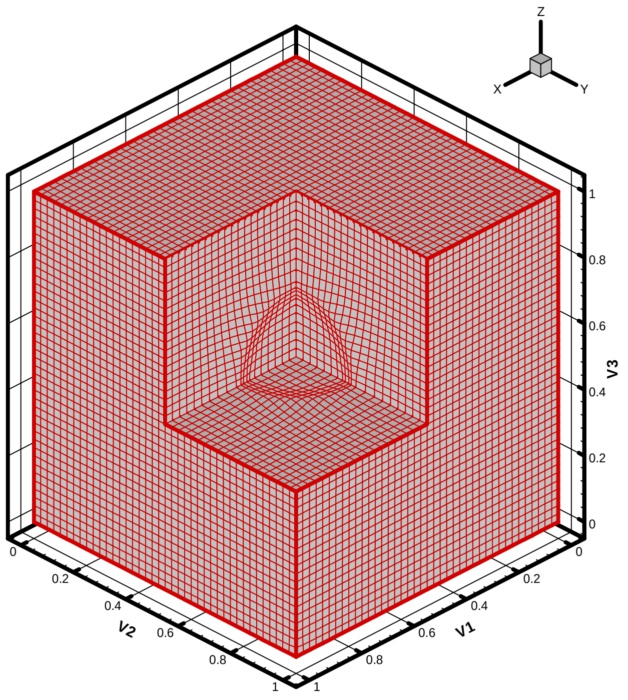

Figure 4.13 Example 4.4 Given transformation: A cube with a ball inside (cutaway plot)

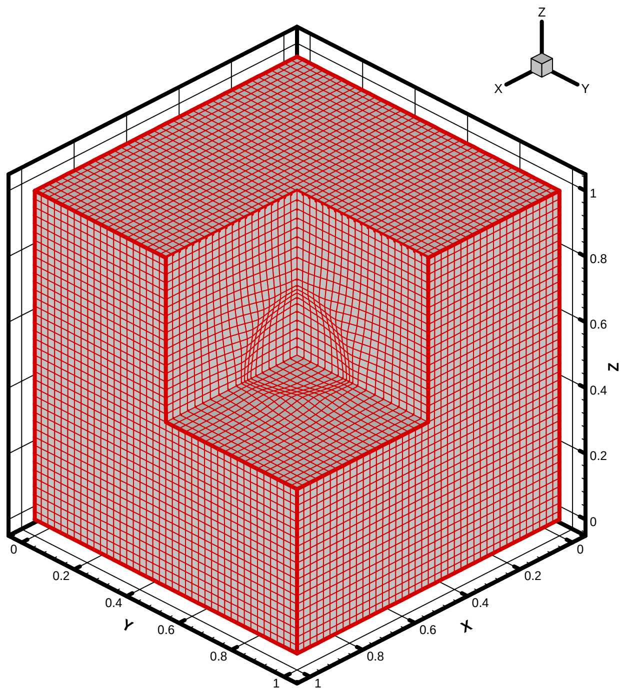

Figure 4.14 Example 4.4 Reconstruction

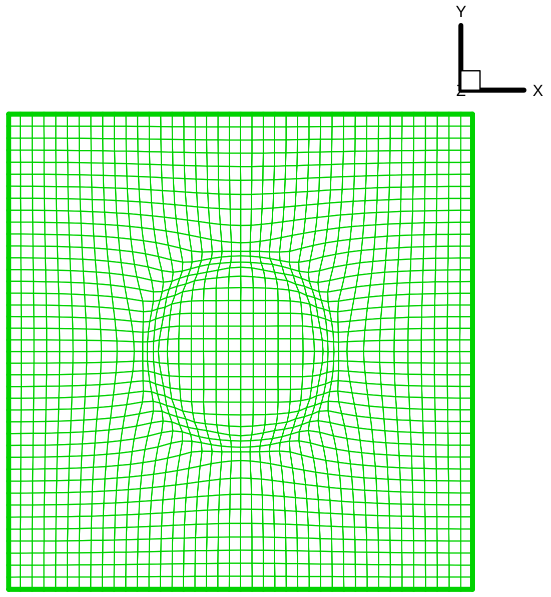

Figure 4.15 Example 4.4 Given transformation: A cube with a ball inside (A slice cut at $z = 0.5$)

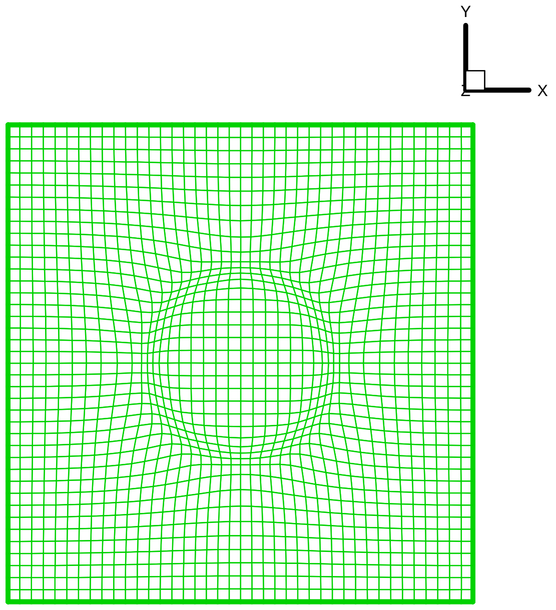

Figure 4.16 Example 4.4 Reconstruction (A slice cut at $z = 0.5$)

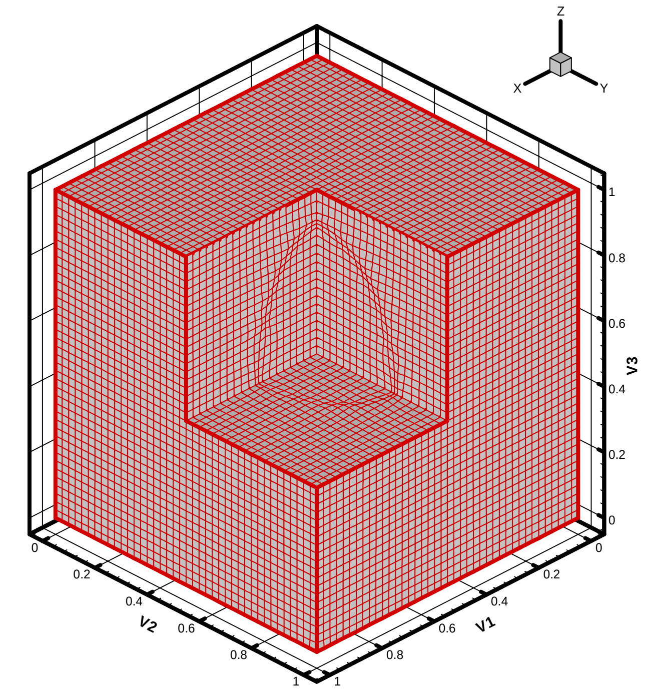

Figure 4.17 Example 4.5 Given transformation: A cube with an ellipsoid inside (cutaway plot)

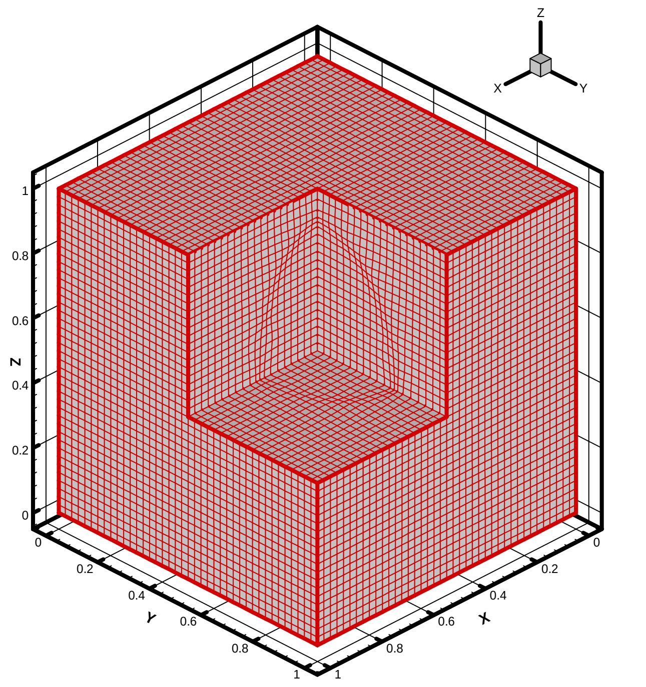

Figure 4.18 Example 4.5 Reconstruct (cutaway plot)

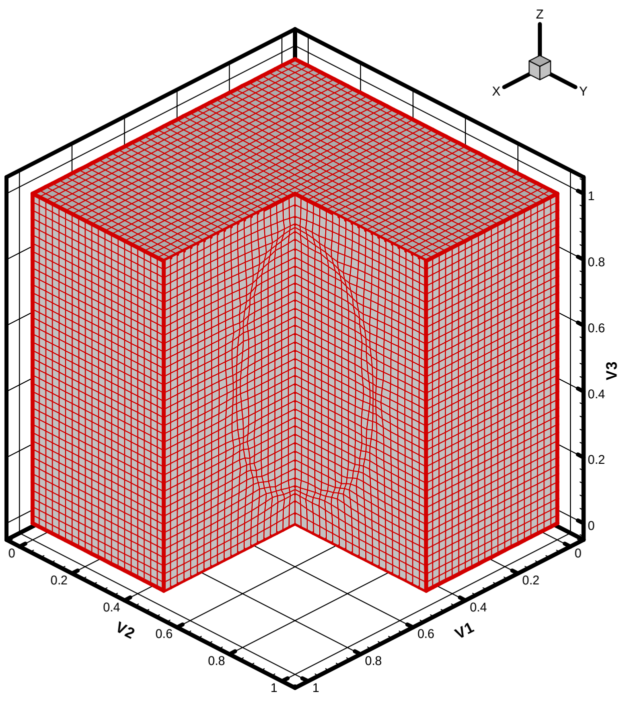

Figure 4.19 Example 4.5 Given transformation: A cube with an ellipsoid inside (cutaway plot)

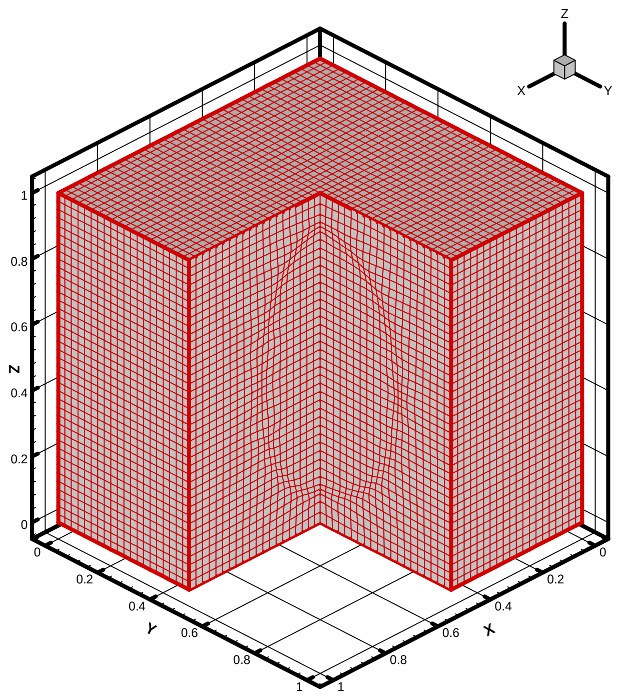

Figure 4.20 Example 4.5 Reconstruct (cutaway plot)

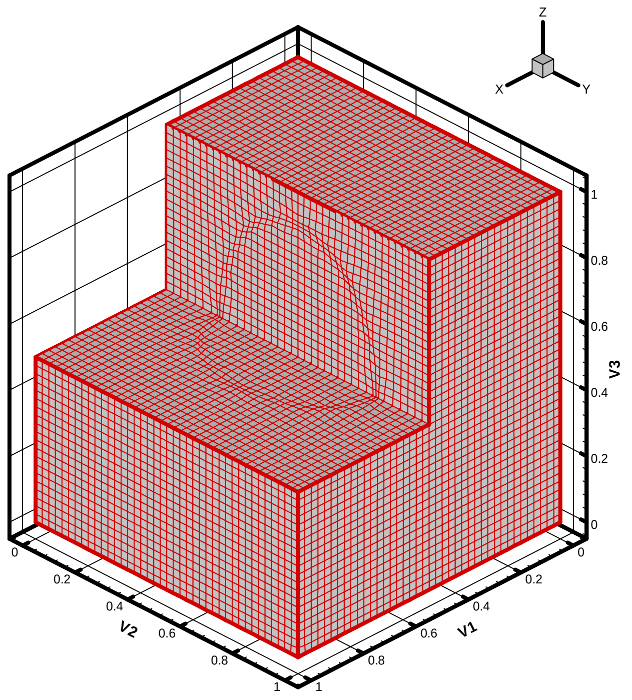

Figure 4.21 Example 4.5 Given transformation: A cube with an ellipsoid inside (cutaway plot)

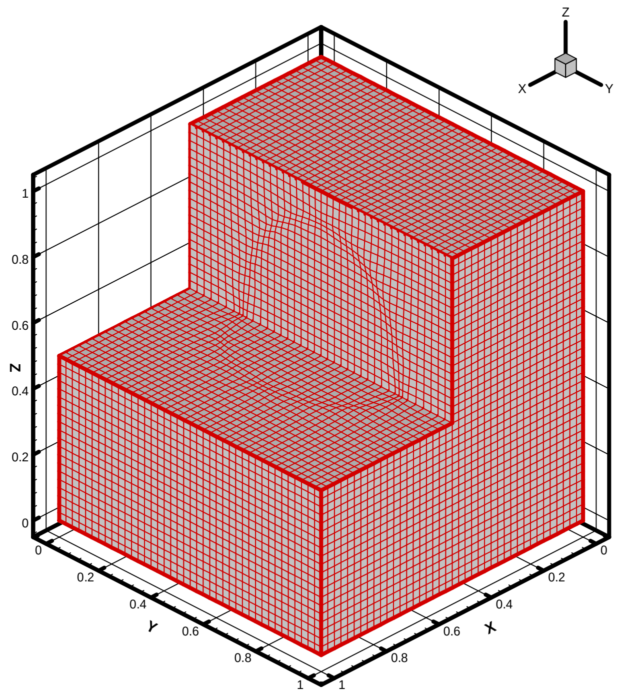

Figure 4.22 Example 4.5 Reconstruct (cutaway plot)

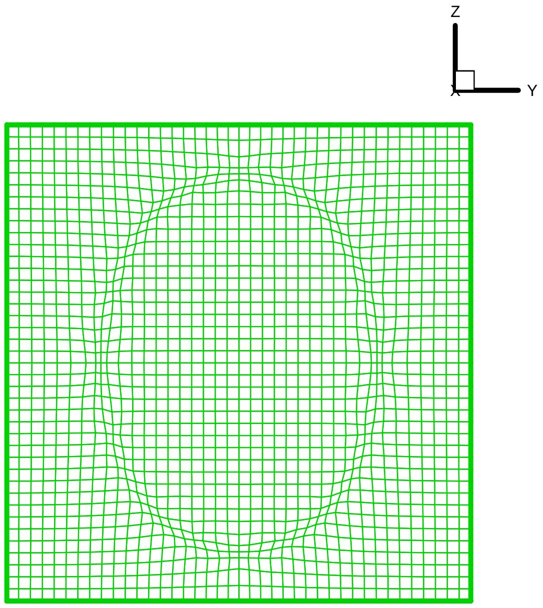

Figure 4.23 Example 4.5 Given transformation: A cube with an ellipsoid inside (A slice cut at $x = 0.5$)

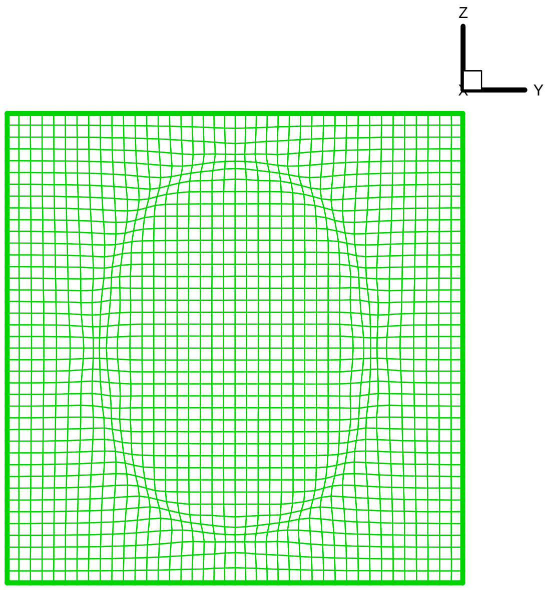

Figure 4.24 Example 4.5 Reconstruction (A slice cut at $x = 0.5$)

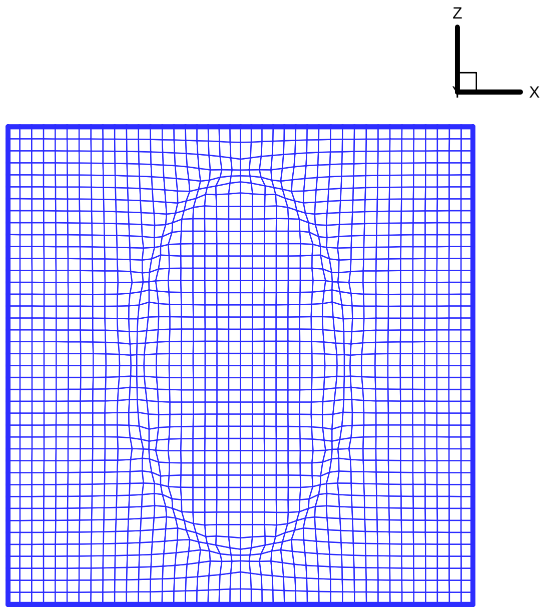

Figure 4.25 Example 4.5 Given transformation: A cube with an ellipsoid inside (A slice cut at $y = 0.5$)

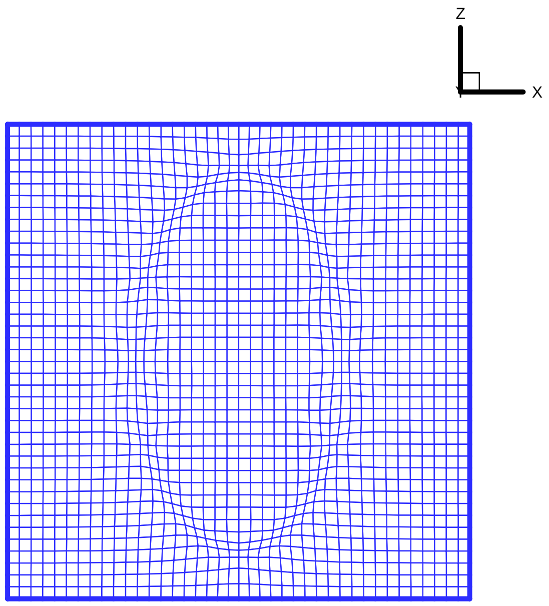

Figure 4.26 Example 4.5 Reconstruction (A slice cut at $y = 0.5$)

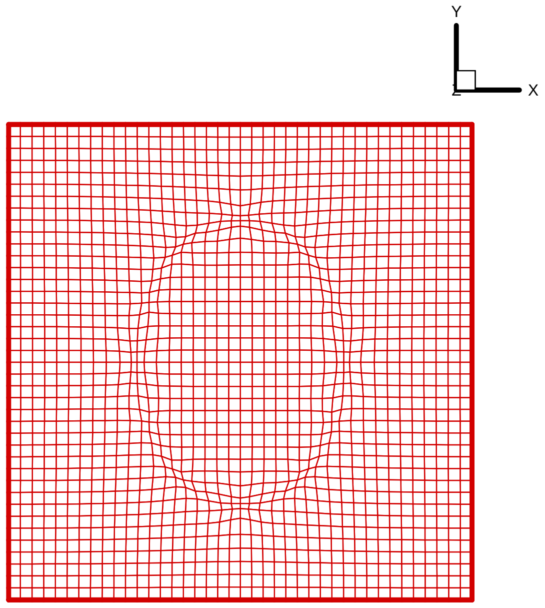

Figure 4.27 Example 4.5 Given transformation: A cube with an ellipsoid inside (A slice cut at $z = 0.5$)

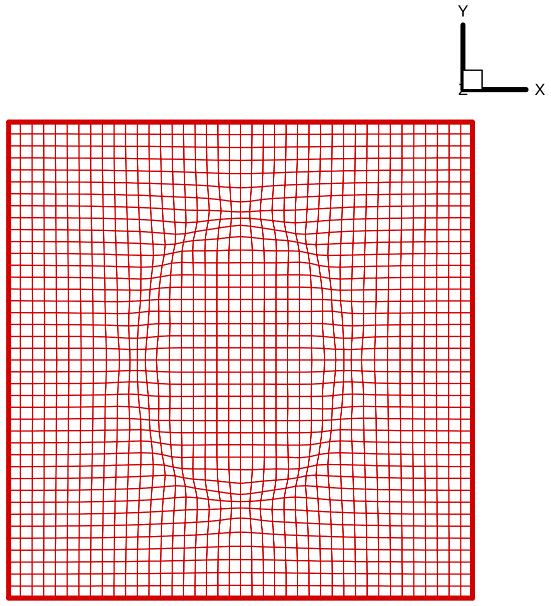

Figure 4.28 Example 4.5 Reconstruction (A slice cut at $z = 0.5$)

CHAPTER 5

CONCLUSION

The three versions of deformation method for adaptive grid generation are presented. If we generate a monitor function f corresponding to the grid size we need, and set the Jacobian determinant $J = f$, by either solving a Poisson equation by finite difference method or solving a div-curl system by least-square finite element method, we can adapt the grids by controlling the cell volume to the desirable cell sizes. Since $f > 0$, we can prove the grid we get is non-folded up to three dimensions.

A three dimension multi-block deformation method is implemented. Multi-block grids allow us to take advantage of the computational efficiency of structured grids and the flexibility for complex geometry. Special treatments for the common boundaries of different blocks are very crucial. Our 3D numerical example shows that the front pass through the common boundary of the two blocks successfully.

The deformation method may be used to find a desired transformation. A brand new idea comes out from applying the deformation method by solving the div-curl system of equations. This idea is to reconstruct a given transformation by directly varying the divergence and curl of each corresponding points. Using these information in the right hand side of the div-curl system and solving the system, we can reconstruct any differentiable transformation. The least-square finite element method is used to solve the div-curl system. Our numerical examples in both two dimension and three dimension show great accuracy. Our future work is to find out the connection of reconstruction to image registration and apply it to the computational image processing problem.

REFERENCES

[1] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, Numerical Grid Generation, North-Holland, Amsterdam, 1985.

[2] P. Knupp and S. Steinberg, The Fundamentals of Grid Generation, CRC Press, 1993.

[3] G. Carey, Computational Grid Generation, Adaptation and Solution Strategies, Taylor and Francis, 1997.

[4] J. F. Thompson, B. Soni, N. Weatherill, Handbook of Grid Generation, CRC Press, 1998.

[5] J. Thompson, A reflection on grid generation in the 90s: trends, needs and influences, 5th International Conference on Numerical Grid Generation in Computational Field Simulations, Mississippi State University, pp.1029-1110, 1996.

[6] B. Hamann, R. J. Moorhead, A survey of grid generation methodologies and scientific visualization efforts, *Chapter 3 in Scientific Visualization: Overviews, Methodologies, and Techniques*, pp. 59-101, 1997.

[7] D. Arney, J. Flaherty, An adaptive mesh-moving and local refinement method for time-dependent partial differential equations, ACM Transaction in Mathematical Software, 16, 1990.

[8] K. Miller, R. Miller, Moving Finite Elements I, *SIAM J. Numer.* Anal. 18, 1019-1032, 1981.

[9] K. Miller, Moving Finite Elements II, *SIAM J. Numer. Anal.* 18, 1033-1057, 1981.

[10] N. Carlson, K. Miller, Design and application of a gradient-weighted moving finite element code, Part I, in 1D, *SIAM J. Sci. Comput.* 19, 728-765, 1998.

[11] N. Carlson, K. Miller, Design and application of a gradient-weighted moving finite element code, Part II, in 2D, *SIAM J. Sci. Comput.* 19, 766-798, 1998.

[12] M. Baines, Moving finite elements, Oxford University Press, New York, 1994.

[13] J. Castillo, Mathematical Aspects of Numerical Grid Generation, Society for Industrial and Applied Mathematics, 1991.

[14] W. Cao, W. Huang and R. D. Russell, Approaches for Generating Moving Adaptive Meshes: Location versus Velocity, *Appl. Num. Math.*, 47 (2003), 121-138.

[15] E. A. Dorfi and L. Drury, Simple adaptive grids for 1D initial value problems, J. Comput. Phys. 69, 175-195, 1987.

[16] W. Huang, Y. Ren and R.D. Russell, Moving mesh partial differential equations (MMPDES) based on the equidistribution principle, *SIAM J. Numer. Anal.* 31, 709-730, 1994.

[17] W. Cao, W. Huang and R. D. Russell, An r-adaptive finite element method based upon moving mesh PDEs, Journal of Computational physics, 170, 871-892, 2001.

[18] R. Li, T. Tang, and P. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, *J. Comput. Physics*, 177, 365-393, 2002.

[19] R. Li, T, Tang, and P. Zhang, Moving mesh methods in multiple dimensions base on harmonic maps, *J. Comput. Physics*, 170, 562-588, 2001.

[20] Y. Di, R. Li, T. Tang, and P. Zhang, Moving mesh finite element methods for the incompressible Navier-Stokes equations, *SIAM, J. Sci. Comput.*, 26, 1036-1056, 2005.

[21] D. Hawken, J. Gottlieb and J. Hansen, Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations, *J. Comput. Physics*, 95, 254-302, 1991.

[22] A. van Dam, P. A. Zegeling, A robust moving mesh finite volume method applied to 1D Hyperbolic conservation laws from magnetohydrodynamics, *J. Comput. Physics*, 2006.

[23] W. Cao, W. Huang and R. D. Russell, An error indicator monitor function for an r-adaptive finite-element method

[24] W. Cao, W. Huang and R.D. Russell, A study of monitor functions for two-dimensional adaptive mesh generation, *SIAM J. Sci. Comput.* 20, 1978-1994, 1999.

[25] H. M. Tsai, A. S. F. Wong, J. Cai, Y. Zhu, and F. Liu, Unsteady flow calculations with a parallel multiblock moving mesh algorithm, *AIAA Journal*, 39, No. 6, 2001.

[26] Thomas J.R. Hughes and Jerrold E. Marsden, A short course in fluid mechanics: mathematics lectures series 6, Publish or Perish, Inc. 1976

[27] J. Moser, Volume elements of a Rieman Manifold, *Trans AMS*, 120, 1965

[28] G. Liao and J. Su, Grid generation via deformation, *Appl. Math. Lett.*, 5, 1992.

[29] G. Liao and D. Anderson, A new approach to grid generation, *Appl. Anal.*, 44, 1992.

[30] P. B. Bochev, G. Liao, and G. C. de la Pena. Analysis and computation of adaptive moving grids by deformation. *Numerical Methods for Partial Differential Equations*, 12, 1996.

[31] B. Semper and G. Liao, "A moving grid finite-element method using grid deformation", *Numer. Meth. PDE*, 11:603, 1995.

[32] G. Liao, T. Pan, and J. Su, "Numerical Grid Generator Based on Moser's Deformation Method", *Numer. Meth. Part. Diff. Eq.* 10, 21 (1994).

[33] G. Liao, G. de la Pena, "A deformation method for moving grid generation", *proceedings, 8th International Meshing Roundtable*, pp. 155-162. South Lake Tahoe, CA, October, (1999).

[34] D. Fleitas, J. Xue, J. Liu and G. Liao, Least-squares finite element adaptive grid deformation in a non-linear time dependent problem. In Advances in applied mathematics (2004 SIAM GATORS), Gainsville, Florida, 2004.

[35] F. Liu, S. Ji, and G. Liao, An adaptive grid method and its application to steady Euler flow calculations, SIAM J. Sci. Comput. 20, 811-825, 1998.

[36] X. Han, C. Xu, and J. L. Prince, A 2D Moving Grid Geometric Deformable Model, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR2003) , June, 2003, pp. I: 153-160.

[37] X. Cai, D. Fleitas, B. Jiang, and G. Liao, Adaptive grid generation based on least-squares finite-element method. *Computers and Mathematics with Applications*, 48, 2004.

[38] J. Xue, Moving grids by the deformation method, Dissertation, 2004.

[39] W. Morris, A meshfree adaptive numerical method, Dissertaion, 2004.

[40] D. Fleitas, The least-square finite element method for grid deformation and meshfree applications, Dissertation, 2005.

[41] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming and P. Krysl, Meshless Methods: an overview and recent developments, Computer methods in applied mechanics and engineering, 139, 3-47, 1996.

[42] B. Jiang, *The Least-Squares Finite Element Method: Theory and Applications in computational Fluid Dynamics and Electromagnetics*. Springer, Berlin, 1998.

[43] M. Grajewski, M. Koster, S. Kilian and S. Turek, Numerical Analysis and Practical Aspects of a Robust and Efficient Grid Deformation Method in the Finite Element Context, preprint, 2005.

[44] C. L. Chang and M. Gunzburger, A finite element method for first order elliptic systems in three dimensions. Appl. Math. Comput. 23, 135-146, 1987.

BIOGRAPHICAL INFORMATION

Jie Liu was born in Jiangxi, China, in 1971. She received her B.S degree in Physics from Jiangxi Normal Univerysity, Nanchang, China, in 1994. After graduation, from 1994-1998, she taught Physics and Science in Huxin Middle School, Shanghai, China. In 1998, she entered the University of Texas at Arlington to pursue graduate studies. She received her M.S degree in Mathematics in 2002. And in 2006 she successfully defended her dissertation, entitled “New Developments of the Deformation method” under the direction of Dr. Guojun Liao, and graduated with a Doctor of Philosophy degree in Mathematical Science. During her graduate studies in the University of Texas at Arlington, she taught many undergraduate mathematics courses and received Outstanding Graduate Student Teaching Award in 2001.