Fitting Subdivision Surfaces to Unorganized Points

by

Cheng Kin Shing Dominic

A thesis submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy
at The University of Hong Kong.

August 2005

Abstract of thesis entitled

Fitting Subdivision Surfaces to Unorganized Points

Submitted by

Cheng Kin Shing Dominic

for the degree of Doctor of Philosophy

at The University of Hong Kong

in August 2005

In this thesis, we study the problem of fitting a subdivision surface to an unorganized point cloud. We choose to use the subdivision surface because it is a compact representation that is capable of representing shapes with arbitrary topology.

On the practical side, we describe a general fitting procedure as an optimization process. Starting with an initial mesh, the positions of the control points are modified to minimize the objective function. The objective function consists of the geometric distance component and the smoothing term component. Since new control points are inserted during the fitting process, this involves a multi-staged optimization process.

We initially follow the traditional approach, reported in a number of studies in recent years, of performing foot point projection and solving a linear optimization problem alternatively and repeatedly. The point distance (PD) error function is commonly employed in the objective function and the resulting method is known as the alternating method. It is the PD error function which makes the alternating method converge only linearly. In the hope of improving the convergence rate for the fitting process, we investigate the tangent distance (TD) error function which has been used in the field of computer vision and the squared distance (SD) error function recently proposed by Pottmann et al. With the use of these distance error functions, we observe faster convergence rates. We also incorporate some slight modifications to improve the stability of the fitting process and resolve several ill-conditioned problems.

Besides outlining a general fitting procedure, we also make several theoretical contributions. We show that methods based on the PD error function are gradient descent

method and methods based on the TD error function are Gauss-Newton method. We also prove that methods based on the SD error function are Newton method. Comprehensive experiments are conducted to investigate the convergence rates and to reveal the advantages and disadvantages of the methods based on various distance error functions. We find that the observed experimental behaviors for PDM, TDM and SDM can be explained by optimization theories regarding the gradient descent method, the Gauss-Newton method and the Newton method. We also apply the trust region method and the line search method to stabilize the fitting process. This improves the performance of the fast converging methods (TDM and SDM).

In summary, this thesis describes the practical flow for fitting subdivision surfaces to point clouds and analyzes fitting methods based on various distance error functions. It clearly demonstrates the relationships between different practical fitting methods and the corresponding optimization techniques.

(417 words)

Fitting Subdivision Surfaces to Unorganized Points

by

Cheng Kin Shing Dominic

A thesis submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy
at The University of Hong Kong.

August 2005

To My Family Members and Friends

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed _____

Acknowledgements

I would like to sincerely thank my supervisor and other colleagues in my research group who helped me a lot throughout my study in these years. I would also like to express my gratitude to all the teachers and tutor partners whom I cooperated during my teaching assistance duties. Last but not least, I would like to express my thanks wholeheartedly to my family members and friends who are willing to share my feelings during these few memorable years. Without their love and support, the completion of this work would not have been possible.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this research, we solve an important problem in the field of computer graphics. Nowadays, point clouds, the resulting products from 3D scanning devices, are important input sources to the modeling modules in computer graphics applications. But, for further geometrical operations, a more compact representation is often required. Subdivision surface is an appropriate choice since subdivision surfaces can represent shapes of arbitrary topologies. We use Loop's subdivision surface $\mathcal{S}$ [1] in our implementation, but the material discussed in this thesis can be applied to other linear subdivision schemes as well. To make our approach work for general situations, we work on raw point clouds.

First, the general flow for fitting subdivision surfaces to unorganized point clouds is described. To start the fitting process, an initial subdivision surface is generated from the point cloud by applying the dual marching cubes approach. Other approaches, which can give initial meshes of correct topologies with reasonable number of control points, can be used in this step. Then, the control points are modified by optimizing the goal function through iterative minimization. Throughout the fitting process, the number of control points is increased and some other parameters are adjusted when necessary. Hence the whole process is a multi-staged optimization problem.

From the optimization point of view, surface fitting problem is a nonlinear least squares problem. The detailed problem formulation will be given in section 1.1. Simply speaking, the goal function consists of a geometric distance term and a smoothing term. Different distance error functions can be used as the local model of the geometric error term in the goal function. Throughout decades, many researchers tackle this problem by following the paradigm of the alternating method, of which the key idea is to sepa-

rate the variables into two groups and then solve a linear least squares problem and a one-dimensional minimization alternatively. In these methods, surprisingly, the point distance (PD) error function seems to be the undoubtable choice for the local model of the geometric distance term. Since only linear convergence rate can be obtained from these methods, we consider other distance error functions, in particular, the tangent distance (TD) error function and the squared distance (SD) error function. (The detailed descriptions about various distance error functions will be given in section 2.3.) With the replacement of the distance error function, we are able to obtain faster convergence rates. Furthermore, we describe how to make use of the Levenberg-Marquardt (LM) method and the Armijo method for improving the stability of the optimization methods.

On the other hand, we find that there does not exist a comprehensive theoretical study on optimization methods for surface fitting. So, besides devising efficient methods for subdivision surface fitting, we also aim at providing analysis of the optimization method. We show that PDM, methods that use the PD error function, are the gradient descent method and TDM, methods that use the TD error function, are the Gauss-Newton method. We also prove that SDM, the newly devised method that uses the SD error function, can be derived from the Newton method. Based on the theoretical study of these methods, the behaviors observed in the experiments can be explained. The relevant fundamental theories in optimization and their relations with the optimization methods used in the surface fitting problem will be given in section 5.

1.1 Problem Formulation

Throughout the thesis, we adopt the notation defined in this section. The target shape is denoted by Γ, which is a point cloud. It is assumed that the underlying surface of the target point cloud is a twice differentiable smooth surface. The subdivision surface $\mathcal{S}$ is the active subdivision surface used to fit to Γ. $\mathcal{S}$ has a control mesh which consists of n control points. Throughout the fitting process, the positions of the n control points $\mathbf{P} = \{P_i\}$ are modified according to the optimization results. In some situations, n can be increased so as to increase the degree of freedom for achieving better fitting result.

During the process of forming local quadratic models of the goal function and evaluating fitting error, points on the subdivision surface $\mathcal{S}$ are sampled. These points,

$P(u_k, v_k)$, on $\mathcal{S}$ are denoted to be sample points (u_k and v_k are surface parameters) and N is the number of sample points.

After defining the notations, we would like to formulate the problem formally. The surface fitting process can be formulated as the following nonlinear least squares problem:

$$Min_{(\mathbf{P},\mathbf{s},\mathbf{t})}\Phi(\mathbf{P},\mathbf{s},\mathbf{t}) = \sum_{i=1}^{N} SDF(\mathbf{P}, s_i, t_i) + SMT(\mathbf{P}), \tag{1.1}$$

where the variables are $\mathbf{P} = \{P_i\}$ (control points of the subdivision surface $\mathcal{S}$) and $\mathbf{s} = \{s_i\}$, $\mathbf{t} = \{t_i\}$ (sets of surface parameters of the foot points on Γ for sample points $P(u_k, v_k)$). Sample points $P(u_k, v_k)$ are linear combinations of the control points P_i. $SDF(\cdot)$, the squared distance error function, gives the squared distances between sample points $P(u_k, v_k)$ and their corresponding foot points $R(s_k, t_k)$ on Γ. $SMT(\cdot)$ is the smoothing term, which is a quadratic function of the control points $\mathbf{P}$. The procedure for solving this problem will be discussed in details in section 2.2 in the next chapter. During the fitting process, the influence of $SMT(\cdot)$ is adjusted and the number of control points (n) of the subdivision surface $\mathcal{S}$ is gradually increased via local subdivisions in regions of large errors. Hence our fitting process is a multi-staged optimization.

1.2 Related Work

The problem of computing a compact surface representation of a target shape given by a set of unorganized data points has many applications in computer graphics, CAD, and computer vision. A typical example is computing a piecewise smooth surface, which can be a B-spline surface (including a NURBS surface) or a subdivision surface, that approximates a given target shape within a pre-specified error tolerance. Compared with the traditional methods based on B-spline surfaces, the approach using subdivision surfaces has gained increasing attention due to the facts that subdivision surfaces can deal with object of general topology and they have arbitrary connectivity of the control meshes [2, 3]. Approaches of different categories were proposed over the past decade. These include local fitting approaches [4–11], active surface approaches [12–23], implicit surface approaches [24–27] and other approaches [28–32]. Among the previous works, some are more relevant to the particular problem that we are solving. In Hoppe et al.'s method [4, 5], an initial dense mesh is generated from a set of unorganized points and is then decimated to fit the target shape via optimiza-

tion of an energy function. Finally, a smooth subdivision surface is obtained from the mesh again via optimization. In Ma et al.'s method [7–9], base surfaces are built for obtaining the parameter values of the data points. Then a least squares procedure is used to fit B-spline surfaces on patches of general quadrilateral topology and Catmull-Clark surfaces on extraordinary corner patches. In [10], closest point search on Loop's surface is performed by combining the Newton iteration and non-linear minimization, followed by an optimization with respect to the L^2 metric. In these methods, when the geometric error between the fitting surface and the target shape needs to be measured or minimized, the PD error function is used. Optimization methods belong to such a scheme, also called the alternating method [33], is typically used for solving separable nonlinear least squares problems, and is known to have only linear convergence.

In this research work, we propose to use the TD error function and the SD error function in the goal function. The fitting behaviors of various distance error functions are observed in the experiments. Our work differs from the two closely related recent works (Pottmann et al.'s work [34] and Kobbelt et al.'s work [35], the extension of [10]) in several aspects. Pottmann et al.'s work makes use of SDM in the problem of B-spline surface fitting. Compared with their work, we perform multi-staged optimization which allows control point insertions. Moreover, we tackle the problem of automatic initial shape specification by applying the dual marching cubes algorithms. Kobbelt et al.'s work applies the blend of PDM and TDM in subdivision surface fitting problem. In our work, besides PDM and TDM, we also consider SDM. More importantly, our work fills the gap between the classical optimization methods and the practical fitting procedures used in decades in the field of computer graphics by presenting the theoretical analysis of old and new fitting methods. On top of that, we also propose stable algorithms by applying a trust region method and a line search method to the optimization process.

Chapter 2

Background

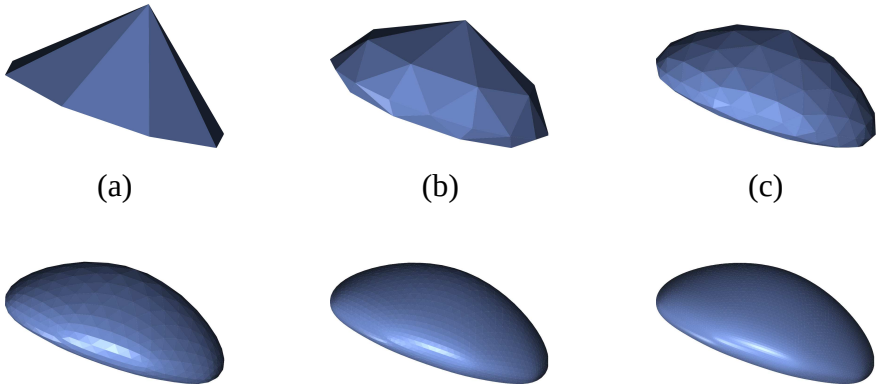
2.1 Subdivision Surfaces

Subdivision surface is defined by a control mesh and a set of subdivision rules. Given a control mesh, the limit surface can be obtained by applying the subdivision rules to the mesh successively.

In 1974, Chaikin devised a subdivision scheme for curve [36]. It was later proved by Riesenfeld [37] that the limit curve under Chaikin's subdivision scheme is a quadratic B-spline curve. In 1978, Doo-Sabin surfaces [38] and Catmull-Clark surfaces [39] were devised. After that, there were many subdivision schemes [1–3, 40, 41] proposed in the literature. Subdivision surface can be classified in different ways: primal [1, 39–41] or dual [38]; triangular [1, 40] or quadrilateral [38, 39, 41] and interpolating [40, 41] or approximating [1, 39]. Primal schemes refer to schemes that involve face refinements while dual schemes refer to schemes that involve vertex refinements. Face refinements mean that new faces are generated from one original face according to the subdivision rule. Vertex refinements mean that new vertices are generated from one original vertex according to the subdivision rule. Surfaces in triangular scheme consist of triangular faces while surfaces in quadrilateral scheme consist of quadrilateral faces. Under interpolating schemes, points in control meshes are also points in the limit surface. Under approximating schemes, points in control meshes are in general not points in the limit surface.

In our work, we choose to use Loop's surface [1]. Loop's scheme, a generation of quartic triangular B-splines devised in 1987, is primal, triangular, approximating and

can produce tangent plane continuous surfaces. Although we choose to implement this particular scheme, our proposed method and analysis can be applied to any linear subdivision scheme. In linear subdivision scheme, control points of subdivision surfaces in the next level can be expressed as a linear combination of control points in the current level. Consequently, points on the limit surface can be expressed as a linear combination of the initial control points. Figure 2.1 shows the meshes from the first level to the sixth level for a polygonal cone. Figure 2.2 shows the first, the second and the third levels of the mesh for the Bunny model.



(a) (b) (c)

(d) (e) (f)

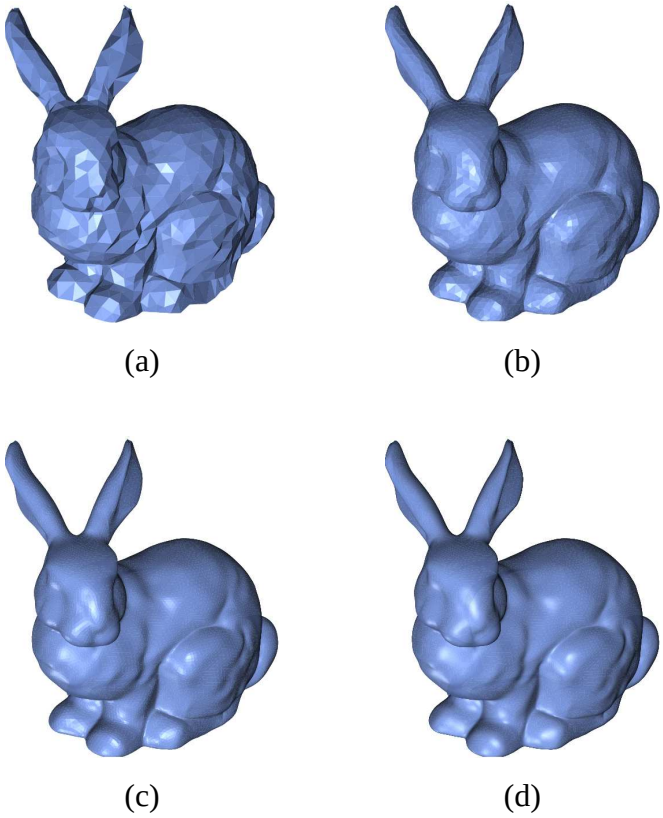
Figure 2.1: Loop's surface: (a) The 1^{st} level mesh (no. of triangles: 16). (b) The 2^{nd} level mesh (no. of triangles: 64). (c) The 3^{rd} level mesh (no. of triangles: 256). (d) The 4^{th} level mesh (no. of triangles: 1024). (e) The 5^{th} level mesh (no. of triangles: 4096). (f) The 6^{th} level mesh (no. of triangles: 16384).

Now, we describe Loop's subdivision scheme. This scheme is designed to apply to triangular polyhedra. For one level of subdivision, a triangle is split into four triangles by adding on each edge a new vertex P_N given by

$$P_N = \frac{3}{8}(P_a + P_b) + \frac{1}{8}(P_c + P_d),$$

where P_a, P_b are the two vertices of the edge, and P_c, P_d are the other vertices of the two triangles that are incident to the edge P_aP_b. See Figure 2.3.
Then the original vertices P_i are modified by the following rule:

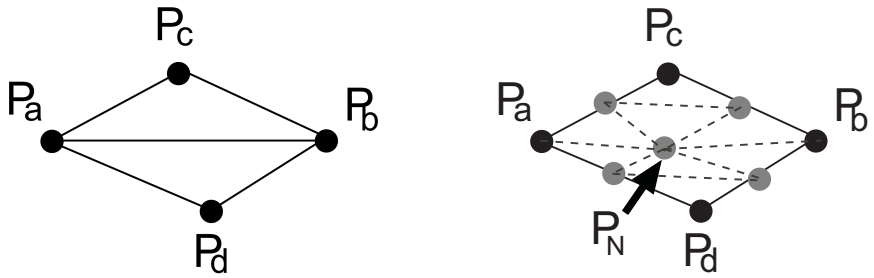
$$P_i = (1 - k\beta)P_i + \beta \sum_{j=1}^{k} P_{n(i,j)},$$



(a) (b)

(c) (d)

Figure 2.2: Loop's surface: (a) The 1^{st} level mesh (flat shading) (no. of triangles: 3040). (b) The 2^{nd} level mesh (flat shading) (no. of triangles: 12160). (c) The 3^{rd} level mesh (flat shading) (no. of triangles: 48640). (d) The 3^{rd} level mesh (smooth shading) (no. of triangles: 48640).



(a) (b)

Figure 2.3: (a) Before the insertion of P_N. (b) After the insertion of P_N.

where k is the degree of the vertex P and $P_{n(i,j)}$ is the j^{th} neighboring point of P_i, $\beta = 3/16$ if $k = 3$, and

$$\beta = \frac{1}{k}\left[\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos^2\left(\frac{2\pi}{k}\right)\right)^2\right]$$

if $k > 3$.

Figure 2.4 illustrates the modification of P_i.

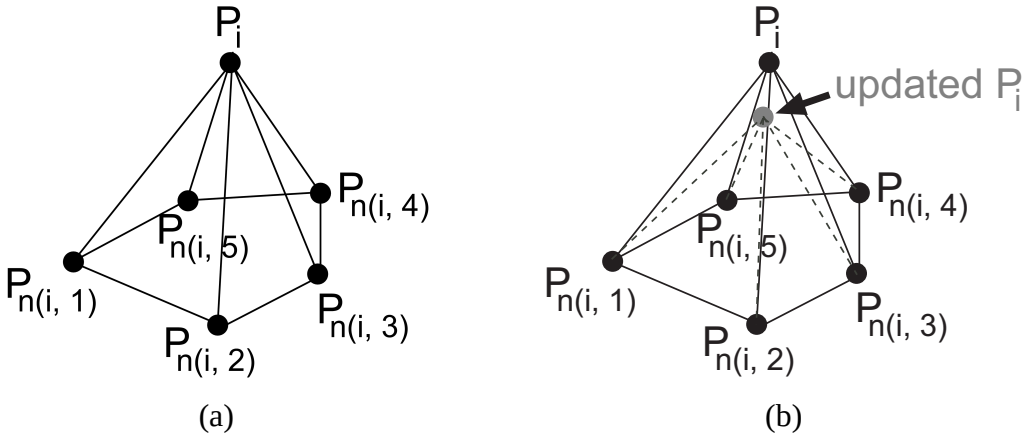


Figure 2.4: (a) Before the update of P_i. (b) After the update of P_i.

2.2 Solving Approach: Separation of Variables

As we described in the Problem Formulation section in the previous chapter, the subdivision surface fitting process solves for the variables $\mathbf{P}$ and $\mathbf{s}, \mathbf{t}$ in Equation 1.1. However, the variables are not all solved at one time. Instead, they are separated into two groups, (i) $\mathbf{P}$ and (ii) $\mathbf{s}, \mathbf{t}$. For fixed $\mathbf{P}$, it is easy to minimize each term $SDF(\mathbf{P}, s_i, t_i)$ with respect to s_i and t_i. This step is done by the foot point projection. During foot point projection, for each sample point $P(u_i, v_i)$ on $\mathcal{S}$, its closest point $R(s_i, t_i)$ on Γ is found. After the step of foot point projection, the variables $\mathbf{s}$ and $\mathbf{t}$ are fixed to be $\overline{\mathbf{s}}$ and $\overline{\mathbf{t}}$. Then, in the next step, the minimization problem becomes:

$$Min_{(\mathbf{P})}\Psi(\mathbf{P}) = \sum_{i=1}^{N} SDF(\mathbf{P}, \overline{s_i}, \overline{t_i}) + SMT(\mathbf{P}), \tag{2.1}$$

which is a linear least squares problem. In this way, the nonlinear least squares problem is solved by performing foot point projection and solving a linear squares problem

repeatedly and alternatively. In other words, separation of variables turns a nonlinear least squares problem into a series of linear least squares problems which can be solved more efficiently. Furthermore, during the fitting process, less memory space is required when the problem is solved using this approach since the number of variables at a particular stage is much fewer. In this research work, we study various local models for $SDF(\cdot)$ in the goal function and aim at obtaining better fitting performance. Three different distance error functions are discussed in the next section.

2.3 Distance Error Functions

During the fitting process, a goal function needs to be defined. The geometric error between the fitting subdivision surface and the target contributes the main part of the goal function. In practice, various different error functions are defined as the local quadratic models of the objective function.

Given a sample point $P(u_{k,0}, v_{k,0})$ on the fitting subdivision surface, its foot point, $R(s_k, t_k)$, on the target shape Γ is determined. In other words, $R(s_k, t_k)$ is the closest point on Γ to $P(u_{k,0}, v_{k,0})$. Then, the distance error function for a variable point $P(u_k, v_k)$ (in the neighborhood of $P(u_{k,0}, v_{k,0})$) to Γ can be defined in several ways.

2.3.1 Point Distance Error

The (squared) point distance (PD) error function is defined by

$$F_{PD}^{+}(P(u_k, v_k), s_k, t_k) = \|P(u_k, v_k) - R(s_k, t_k)\|_2^2. \tag{2.2}$$

Optimization schemes using the PD error function are called Point Distance Minimization (PDM). PDM is widely used in existing optimization applications such as [42–44]. From the theoretical point of view, PDM is just the gradient descent method. It is well known to have linear convergence rate. The simplicity of this error function may explain its popularity.

2.3.2 Tangent Distance Error

The (squared) tangent distance (TD) error function is defined by

$$F_{TD}^{+}(P(u_k, v_k), s_k, t_k) = [(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{N}]^2, \tag{2.3}$$

where $\mathcal{N}$ is the unit normal vector at the foot point $R(s_k, t_k)$.

Optimization schemes using the TD error function are called Tangent Distance Minimization (TDM). Blake et al. [12] used the TD error in the problem of extracting boundary curves of the objects in images. From the theoretical point of view, TDM is the Gauss-Newton method. For zero residual problem, it is well known that the Gauss-Newton method has quadratic convergence rate [45–47]. For large residual problem, the Gauss-Newton method can have unstable behavior and cannot achieve quadratic convergence rate.

2.3.3 Squared Distance Error

Recently, Pottmann et al. [34, 48, 49] proposed a general paradigm of shape approximation based on the minimization of a novel quadratic approximant of the squared distance function. Let $d = \|P(u_{k,0}, v_{k,0}) - R(s_k, t_k)\|_2$ be the Euclidean distance between $P(u_{k,0}, v_{k,0})$ and $R(s_k, t_k)$. Let ρ_1 and ρ_2 be the principal curvature radii of the surface Γ at $R(s_k, t_k)$. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the unit vectors in the corresponding principal curvature directions. Let $\mathcal{N}$ be the unit normal vector, i.e., $\mathcal{N} = \mathcal{T}_1 \times \mathcal{T}_2$. A quadratic approximant of the squared distance function from a variable point $P(u_k, v_k)$ in the neighborhood of $P(u_{k,0}, v_{k,0})$ to Γ is given by

$$\begin{aligned} F_{SD}^{+}(P(u_k, v_k), s_k, t_k) &= \frac{d}{d-\rho_1}[(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{T}_1]^2 \\ &+ \frac{d}{d-\rho_2}[(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{T}_2]^2 \\ &+ [(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{N}]^2. \end{aligned} \tag{2.4}$$

It is noticed that $F_{SD}^{+}(\cdot)$ can be negative since the coefficients $\frac{d}{d-\rho_1}$ and $\frac{d}{d-\rho_2}$ can be negative. In practice, this function is modified as follows for ensuring that the resulting matrix for the local quadratic model is positive definite:

$$\begin{aligned} F_{SD}^{+}(P(u_k, v_k), s_k, t_k) &= \left\lfloor \frac{d}{d-\rho_1} \right\rfloor_0 [(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{T}_1]^2 \\ &+ \left\lfloor \frac{d}{d-\rho_2} \right\rfloor_0 [(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{T}_2]^2 \\ &+ [(P(u_k, v_k) - R(s_k, t_k))^T \mathcal{N}]^2, \end{aligned} \tag{2.5}$$

where $\lfloor x \rfloor_0 = max\{x, 0\}$.

The signs of d, ρ_1 and ρ_2 are determined with respect to the local frame at the target point. The radii of curvature $\rho_{1/2}$ is positive if the curvature center in the normal section along the tangent vector $\mathcal{T}_{1/2}$ is outside the target shape. Otherwise, $\rho_{1/2}$ is negative. Figure 2.5 illustrates the cases for positive curvature and negative curvature.

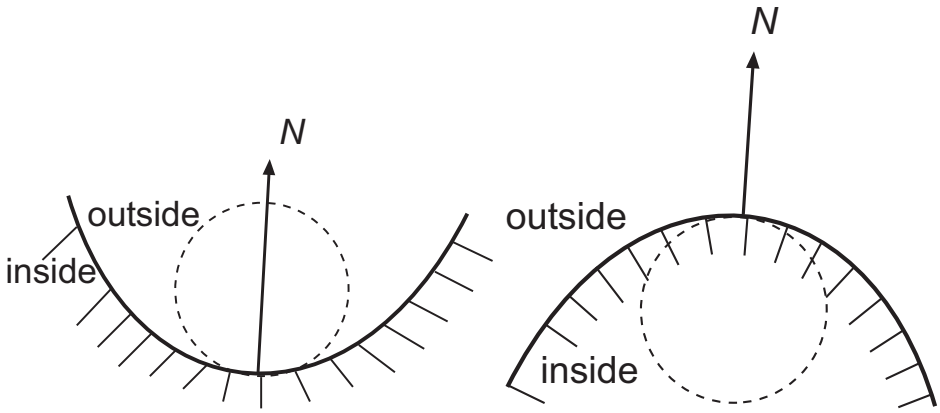


Figure 2.5: Left: Positive curvature; Right: Negative curvature.

This function is called the squared distance (SD) error function. Optimization schemes using the SD error function are called Squared Distance Minimization (SDM). The ellipsoid in Figure 2.6 shows an iso-distance surface defined by the SD error function for surface fitting.

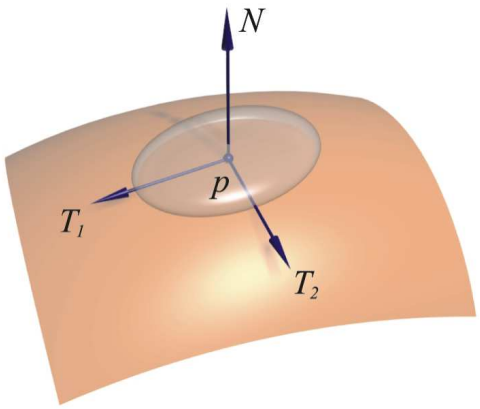


Figure 2.6: An iso-surface of the SD error function, with a local coordinate frame at the point p.

Pottmann et al. applied SDM successfully to solve a series of geometric optimiza-

tion problems, including the problem of fitting B-spline curves and surfaces to some smooth target shapes [34, 48]. Yang et al. [50] studied how to define initial shapes and adjust the number of control points when using SDM for B-spline curve approximation. Wang et al. [51] solved the B-spline curve fitting problem using SDM by projecting the points in the target point cloud to the B-spline curve. In our work, we use the SD error function in the problem of fitting subdivision surfaces to point clouds. SD error function is an appropriate error metric for shape approximation since it measures the distance from a data point to a surface rather than the distance from a data point to a particular point on the surface. Unlike the PD error function, the SD error function takes the local geometry of the target surface Γ into account. At low curvature regions of the target, the points of the fitting surface can move tangentially to attain a better distribution without causing much penalty from the SD error term. However, tangential movement is inhibited by the PD error term. For the TD error term, it is not stable near high-curvature regions because tangent planes are poor approximations to the surface in high-curvature regions. Inappropriate large step size is used in TDM and this is due to the omission of important curvature related parts in the true Hessian of the goal function. From the theoretical viewpoint, one can view SDM as a modification of the Newton method. It converges much faster than the commonly used PDM.

In chapter 5, relationships between PDM, TDM, SDM and the standard optimization methods are given in more details. Furthermore, we will show that SDM can be derived from the Newton method. Hence, we expect that SDM has better convergence behavior than PDM (which is known to have only linear convergence).

Chapter 3

Fitting Procedure

In this chapter, we describe the general flow of fitting subdivision surfaces to point clouds. More detailed issues about some steps in the procedure are discussed in the next chapter. Different optimization methods, such as PDM, TDM and SDM, share the same flow. Figure 3.1 illustrates the overall procedure.

3.1 Normalization of Target Shape

The dimensions of input point clouds are arbitrary. Different 3D scanning systems or modeling modules produce models of different dimensions. In order to avoid the dependence of the parameters in the optimization system on the dimensions of the input point clouds, target shapes are normalized by uniform scaling such that all data points fall in the cube $[0, 1]^3$. Specifically, the longest dimension among x, y and z is scaled to 1.0. This step makes the terms in the least squares formulation unitless.

3.2 Pre-computations of Distance Field and Curvature Information

We tackle the subdivision surface fitting problem using the approach of separation of variables. Foot point projection is one of the key steps. This step is costly if foot point projection is done in a brute-force manner. To improve the efficiency of setting up the local model of the goal function, distance field for the target shape Γ is computed. Also, curvatures, which are required in SDM, are also computed in a preprocessing

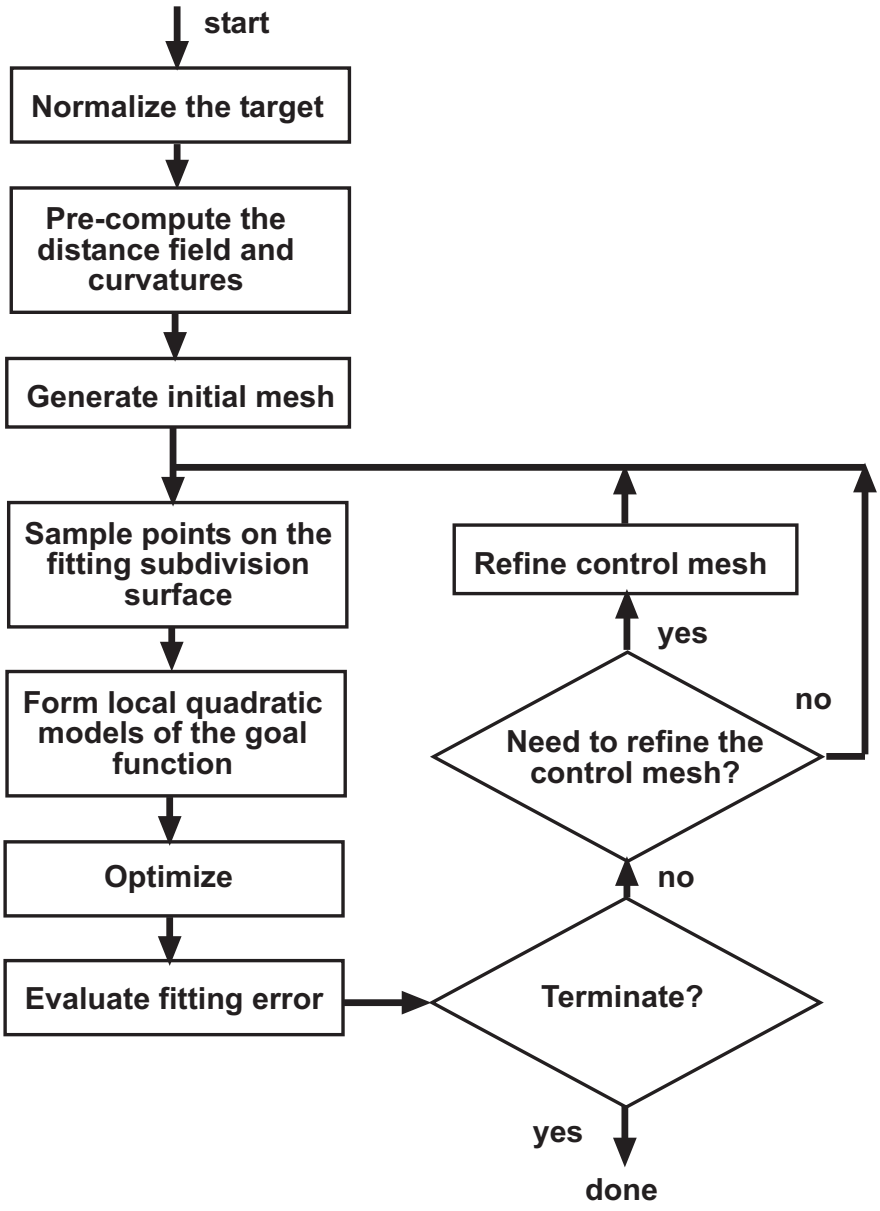


Figure 3.1: The fitting procedure.

step.

3.3 Initial Mesh Generation

To start the optimization process, an initial control mesh for the fitting subdivision surface is required. Since the approach described in this thesis is not a global approach, the initial mesh cannot be too far away from the target. To obtain an initial control mesh, we start with an octree for the target shape and then follow the procedure in the dual marching cubes algorithm [52]. For more details, see section 4.3 in the next chapter.

3.4 Points Sampling on Active Subdivision Surface

Setup of the equation for minimizing the goal function and error evaluation require a set of sample points $P(u_{k,0}, v_{k,0})$. These sample points are the points on the limit surface of the active subdivision surface. They are generated using the approach devised by Stam [53, 54].

3.5 Equation Setup

After the N sample points $P(u_{k,0}, v_{k,0})$ are generated in the previous step, the following goal function can be set up:

$$F^+(\mathbf{P}, \mathbf{s}, \mathbf{t}) = \frac{1}{N} \sum_{k=1}^{N} F_D^+(P(u_k, v_k), s_k, t_k) + \lambda F_s, \tag{3.1}$$

where $P(u_k, v_k)$ is a variable sample point associated with $P(u_{k,0}, v_{k,0})$, s_k and t_k are the surface parameters of the foot point on Γ for the sample point $P(u_{k,0}, v_{k,0})$ and $F_D^+(\cdot)$ is the local quadratic models for the geometric error part of the goal function. Depending on which distance error function is used, $F_D^+(\cdot)$ can be $F_{PD}^+(\cdot)$, $F_{TD}^+(\cdot)$ or $F_{SD}^+(\cdot)$ (the distance functions for PDM, TDM and SDM described in the previous chapter). F_s is a smoothing term and λ is the coefficient for F_s.

Compared with the PD error, the TD error and the SD error require the normal vector information. Additionally, the SD error further requires the curvature information. Those normal vector and curvature information can be obtained efficiently after the pre-processing steps.

3.6 Equation Solving

Since the variable sample point $P(u_k, v_k)$ is a linear combination of the control points P_i, the function $F_D^+(\cdot)$ is a quadratic function of the P_i. Since F_s is also a quadratic function of the control points P_i, the whole goal function is a quadratic function of the control points P_i. The updated control points P_i can be computed by solving a linear system of equations. Moreover, since each variable sample point $P(u_k, v_k)$ is only influenced by a small number of control points, the matrix for the resulting linear system of equations is sparse. So, in the implementation, instead of storing the whole matrix, a sparse data structure [55] is used. After that, instead of using some direct methods, we use an iterative method, the conjugate gradient (CG) method [45, 56–60], to solve the equation. Then the positions of the control points of the subdivision surface are updated accordingly.

3.7 Error Evaluation

After the control points have been updated, the maximum error E_m and root-mean-square error E_{rms} are evaluated. E_m is defined by the maximum of the distances of all the sample points $P(u_{k,0}, v_{k,0})$ on the fitting surface $\mathcal{S}$ to the target shape Γ, i.e.

$$E_m = \max_k \{||P(u_{k,0}, v_{k,0}) - R(s_k, t_k)||_2\}.$$

E_{rms} is defined as:

$$E_{rms} = \left[\frac{1}{N}\sum_k ||P(u_{k,0}, v_{k,0}) - R(s_k, t_k)||_2^2\right]^{\frac{1}{2}}.$$

If E_{rms} falls below a pre-specified error threshold, the fitting process can be terminated. Otherwise, the control mesh needs to be refined.
(E_{rms} can roughly reflect the visually-perceived error. If E_{rms} is small, the visually-perceived errors for most regions should be small. On the other hand, if E_{rms} is large,

the visually-perceived errors for most regions should be large. If E_m is much larger than E_{rms}, some regions are likely to have relatively larger visually-perceived error than other regions. The reason is the lack of degrees of freedom in those regions which need local subdivisions.)

3.8 Control Mesh Refinement

Whenever E_m stops getting improved, new control points are inserted in the regions of large errors so as to increase degrees of freedom for better fitting.
Then, the fitting process will continue from the step ”sampling on the active subdivision surface”.

Chapter 4

Implementation Issues

In this chapter, we discuss some implementation issues in various steps of the fitting procedure.

4.1 Curvature Pre-computation

The curvature information of the target shape is required in SDM. As described in the previous chapter, this information is pre-computed for efficient access later. We employ the following simple method for our purpose. For a given target point R_i, its neighboring points $R_{n(i,j)}$ are identified. To do this, the neighborhood size, which depends on the sampling density of the point cloud, needs to be determined. The process of neighboring points identification is speeded up by the use of an octree for the target points. Let $R_{c,i}$ denote the centroid of the neighboring points. Then, the principal curvature directions and the normal direction are computed as the eigenvectors of the covariance matrix $\mathcal{CV}$ given by

$$\mathcal{CV} = \sum_j (R_{n(i,j)} - R_{c,i})(R_{n(i,j)} - R_{c,i})^T.$$

After that, we fit a quadratic polynomial (in the form of $z = k_1 x^2 + k_2 y^2$) to the points $R_{n(i,j)}$ in the local coordinate system formed with the principal curvature directions and the normal direction at $R_{c,i}$. With the coefficients k_1 and k_2 determined, the principal curvatures are simply $2k_1$ and $2k_2$. Besides this simple approach, there exist other methods. For example, Goldfeather et al. [61] proposed for estimating curvatures and principal directions from point clouds using a cubic-order algorithm.

4.2 Distance Field

When the equation for minimizing the goal function is set up, foot point projection is required. To improve the efficiency at run time, we compute an adaptive distance field of the target shape in preprocessing using the idea proposed in [62]. During the optimization process, the distance for a sample point $v_{k,0}$ is computed by trilinear interpolation from the stored values in the smallest node where the sample point is located. Similar pre-computation technique of the distance field has been used in [50]. Figure 4.1 shows an adaptive distance field for the ball joint.

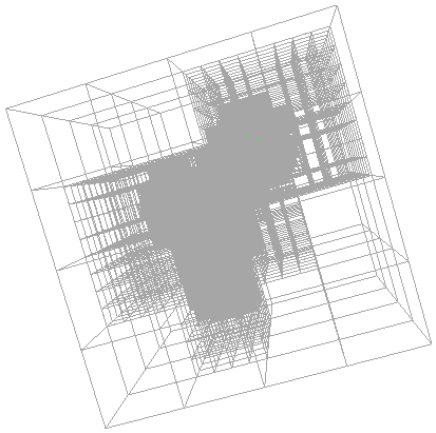


Figure 4.1: An adaptive distance field for a ball joint.

4.3 Initial Mesh Generation

To start the optimization process, an initial control mesh is required. One approach is to construct an octree partition of the point cloud with uniform cell size. Then, a mesh is obtained using the Marching Cubes algorithm [63]. The cell size of the octree is small enough so that the resulting mesh has the same topology as the target point cloud. To capture small features of a target shape, the Marching Cubes algorithm can be applied with a sufficiently small cell size to obtain a dense initial mesh before simplifying the mesh adaptively to reduce the total number of triangles [64].

Schaefer et al. proposed Dual Marching Cubes approach [52]. In our implementation, we follow this approach since it is an adaptive approach and also it does not require extra cracks-filling process. Initially, an octree is built. During the generation of the

octree, whether a cell needs further subdivision depends on the residual value of a QEM-like quadratic error function [64]. (The function is described below shortly.) If the residual value of a node exceeds a threshold value, the node is further subdivided into eight children. In this way, the sizes of the leaf nodes are not of uniform size. Next, one feature point is determined in each octree cell via optimizing a QEM-like quadratic function $QD(\cdot)$. Specifically, inside each cell, the function to be optimized has the form:

$$QD(w,p) = \sum_i \frac{(w - G_i(p))^2}{1 + \| \bigtriangledown d(p_i)\|^2}, \tag{4.1}$$

where $d(\cdot)$ is the cost function in the cell (in our case, the distance function), p is the feature point to be found, w is the value of $d(\cdot)$ at p and $G_i(\cdot)$ is a function defined at p_i:

$$G_i(p) = \bigtriangledown d(p_i)^T (p - p_i), \tag{4.2}$$

where p_i are sample points used for forming the quadratic function for optimization. In our implementation, $d(\cdot)$ is set to be the signed distance function and the corners of the octree cells are taken to be p_i.

Since the feature point of a cell should be inside the cell or on the boundary of the cell, constrained optimization is carried out. We call the OptSolve++ library for the optimizations. After that, a dual grid is constructed using the feature points in the octree cells. Finally, the initial mesh is generated by referencing the lookup table (used in the ordinary Marching Cubes algorithm) for each cell in the dual grid. Figure 4.2 illustrates the procedure for the dual marching cubes algorithm. Figure 4.3 shows the octree and the dual grid for a horse.

Another approach is to model small details by adding a displacement map over a smooth surface [65]. The visual output from this approach is impressive but it does not meet our goal of computing a complete surface representation for a point cloud.

4.4 Efficient Point Sampling on Subdivision Surface

From the formulation of the goal function, sample points on the subdivision surface are required. Given the control points P_i, the task is to evaluate sample points $P(u_{k,0}, v_{k,0})$. According to the essence of subdivision surface, sample points on the subdivision surface can be obtained by applying the subdivision rule repeatedly on the control mesh which is defined initially by the control points P_i. However, it is not clear how

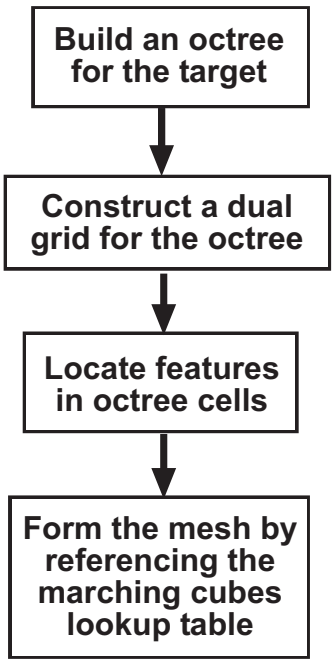


Figure 4.2: Procedure for the Dual Marching Cubes Method.

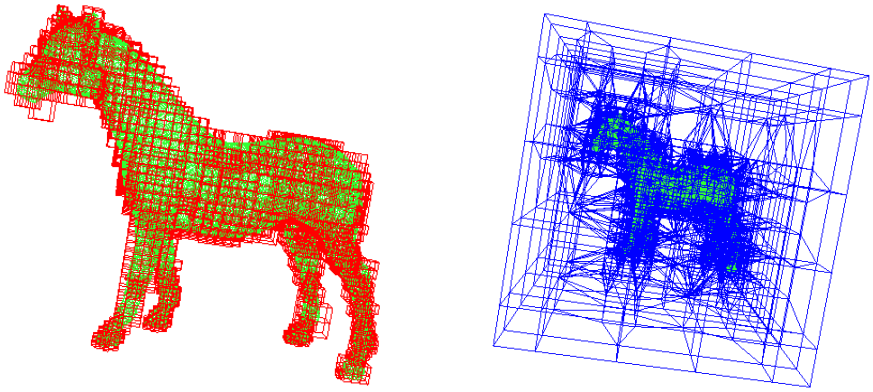


Figure 4.3: An octree and a dual grid for a horse.

to evaluate the particular point $P(u_{k,0}, v_{k,0})$ with the parameters $u_{k,0}$ and $v_{k,0}$ before Stam devised his approach in [53, 54]. In his approach, given the subdivision matrix (determined by the subdivision scheme), the eigenstructure of the subdivision matrix is analyzed. With the pre-computed coefficients of the eigenbasis functions (these need to be computed once for a subdivision scheme), the sample point $P(u_{k,0}, v_{k,0})$ can be computed efficiently by:

$$P(u_{k,0}, v_{k,0}) = \mathcal{P}^T \hat{b}(u_{k,0}, v_{k,0}), \tag{4.3}$$

where $\mathcal{P} = \{P_i\}$ is the control points of the subdivision surface and $\hat{b}(\cdot)$ stores the eigenbasis functions.

In our implementation, we follow Stam's approach for sample points evaluation on the subdivision surface.

4.5 Indexed Storage for Sparse Matrices

Since each sample point on the subdivision surface is just a linear combination of a few control points, the matrix for solving the linear system of equations is sparse. In order to reduce the storage for zero elements and the computation time for the matrix operations, an indexed storage, instead of the whole matrix, is used. We apply the indexed storage described in [55]. It is a row-indexed scheme, in which two one-dimensional arrays are used. All the diagonal elements, including zero elements, are stored. But, for off-diagonal elements, only non-zero elements are stored. Therefore, the storage size required by this row-indexed scheme is roughly two times the number of non-zero elements in the matrix. This scheme contributes a large reduction from the full matrix representation for a sparse matrix.

This sparse representation is simple for a matrix to multiply itself or its transpose by a vector to its right.

4.6 Equation Solving using the Conjugate Gradient Method

Once the linear system of equations for minimizing the goal function has been setup, it can be solved by applying direct methods such as Gaussian elimination or Cholesky decomposition. However, these methods are expensive especially for problems of large

dimensions. For example, both Gaussian elimination and Cholesky decomposition cost $O(n^3)$ if the number of equations is n.

The conjugate gradient method (CG) is a well-known iterative method [45, 56–60] that can be used to solve linear systems of which the coefficient matrices are symmetric and positive definite. Specifically, the function to which the CG method applies should have the form:

$$f(x) = \frac{1}{2}x^T Ax - b^T x, \tag{4.4}$$

where A is symmetric and positive definite.

Given an initial guess x_0, iterates x_i are generated according to the following formulae:

$$x_{i+1} = x_i + \sigma_i p_i, \tag{4.5}$$

$$r_{i+1} = r_i - \sigma_i A p_i, \tag{4.6}$$

$$\sigma_i = \frac{r_i^T r_i}{p_i^T A p_i}, \tag{4.7}$$

$$p_{i+1} = r_{i+1} + \varsigma_i p_i, \tag{4.8}$$

and

$$\varsigma_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}. \tag{4.9}$$

In the above formulae, p_i is the search direction for the i^{th} iteration and it can be verified easily that r_i is the residue $(b - Ax_i)$ for the i^{th} iteration. Initially, $r_0 = b - Ax_0$ and $p_0 = r_0$.

Given the direction p_i, the new iterate is generated by moving the current iterate along this direction by a step size σ_i. σ_i is obtained by a one-dimensional minimization.

$$\frac{df(x_i + \sigma_i p_i)}{d\sigma_i} = p_i^T A x_i + \sigma_i p_i^T A p_i - p_i^T b. \tag{4.10}$$

After setting $\frac{df(x_i+\sigma_i p_i)}{d\sigma_i}$ to zero (a necessary condition for $f(x_i + \sigma_i p_i)$ to have a minimum), we have,

$$\sigma_i = \frac{p_i^T r_i}{p_i^T A p_i}. \tag{4.11}$$

And also, ς_i is determined such that p_{i+1} is orthogonal to all Ap_j and r_j, where $j < i$.

To summarize, the vectors in the CG method have the following properties:

$$p_i^T A p_j = 0, \forall i \neq j;$$

$$r_i^T r_j = 0, \forall i \neq j;$$

and

$$r_i^T A p_i = p_i^T A p_i.$$

From the above properties and for the reason of computational efficiency, σ is computed using Equation 4.20 rather than Equation 4.11 ($r_i^T r_i$ needs to be computed anyway). It can be observed that only matrix-vector multiplications and inner-products are required in the CG method. Matrix-vector multiplications are carried out efficiently since the matrix involved is a sparse matrix and is stored in an indexed structure [55] (described in the previous subsection).

Theoretically, the maximum number of iterations needed is bounded by the number of distinct eigenvalues of the matrix A [57, 59, 60]. More specifically, the number of iterations depends on the distribution of the eigenvalues of the coefficient matrix. In general, the performance for the CG method is better for coefficient matrices that have clustered eigenvalues distribution than coefficient matrices that have uniform eigenvalues distribution.

The coefficient matrix for the linear system is determined by the problem. But, in order to improve the convergence rate of the CG method, a technique, called preconditioning, can be applied. The idea is to pre-multiply another matrix to the original matrix so that the modified problem, of which the coefficient matrix has a better structure from the viewpoint of the CG method, can be solved in a more efficient way. A simple way to construct a matrix for preconditioning, also known as Jacobi preconditioning,

is to use a diagonal matrix of which the diagonal elements are the corresponding elements of the original coefficient matrix [56]. For preconditioning, the substitution for $\hat{x} = M^{1/2}x$ is made and Equation 4.4 becomes:

$$f(\hat{x}) = \frac{1}{2}\hat{x}^T(M^{-T/2}AM^{-1/2})\hat{x} - (M^{-T/2}b)^T\hat{x}. \tag{4.12}$$

From the implementation point of view, the modification for incorporating preconditioning is minimal. An additional solving of the following equation is needed:

$$Mw_i = r_i, \tag{4.13}$$

where M is the matrix for preconditioning.
Then, some of the original formulae are modified accordingly as follows:

$$\sigma_i = \frac{w_i^T r_i}{p_i^T A p_i}, \tag{4.14}$$

$$p_{i+1} = w_{i+1} + \varsigma_i p_i, \tag{4.15}$$

and

$$\varsigma_i = \frac{w_{i+1}^T r_{i+1}}{w_i^T r_i}. \tag{4.16}$$

The CG method works for the problems that have symmetric positive definite coefficient matrices. For problems that do have non-symmetric matrices, one approach is to solve the normal equation instead of the original problem (i.e. to solve $A^TAx = A^Tb$ instead of solving $Ax = b$). This approach is sometimes known as the CGNR method (Conjugate Gradient Method on the Normal Equations) [56]. Although the new coefficient matrix A^TA is guaranteed to be symmetric, its condition number doubles that of the original matrix A. This has negative impact on the convergence rate of the optimization process and the accuracy of the solution.

Alternatively, in order to solve the problems that have non-symmetric coefficient matrices, the bi-conjugate gradient method (biCG) [56] can be applied. The iterative formulae are similar to those in the CG method, with some additional vectors. Initially, $p_0 = \tilde{p}_0 = r_0$.

$$x_{i+1} = x_i + \sigma_i p_i, \tag{4.17}$$

$$r_{i+1} = r_i - \sigma_i A p_i, \tag{4.18}$$

$$\tilde{r}_{i+1} = \tilde{r}_i - \sigma_i A^T \tilde{p}_i, \tag{4.19}$$

$$\sigma_i = \frac{r_i^T \tilde{r}_i}{\tilde{p}_i^T A p_i}, \tag{4.20}$$

$$p_{i+1} = r_{i+1} + \varsigma_i p_i, \tag{4.21}$$

$$\tilde{p}_{i+1} = \tilde{r}_{i+1} + \varsigma_i \tilde{p}_i, \tag{4.22}$$

and

$$\varsigma_i = \frac{r_{i+1}^T \tilde{r}_{i+1}}{r_i^T \tilde{r}_i}. \tag{4.23}$$

Regarding the doubling of the condition number of the coefficient matrix in the CGNR method, the biCG method is preferable to the CGNR method. Similar to the case in the CG method, preconditioning can be applied to the biCG method to improve its performance. Practically, the biCG method works well in most cases. However, there does not exist much information about the convergence rate of the biCG method in the literature.

In our fitting procedure, one of the critical steps is to solve the linear system of equations for minimizing the goal function. With the use of the sparse data structure (described in the previous section) and the conjugate gradient method, the linear system is solved efficiently. When implementing the conjugate gradient solver, we take [55] as reference. The conjugate gradient solver is terminated when the relative error, $\frac{\|b-Ax\|}{\|b\|}$, is less than 10^{-6}. In most cases, the number of iterations by the CG solver is far less than the number of variables. In order to avoid the CG solver from running infinitely, the maximum number of iterations is set to the maximum of 200 and the double of the number of variables. We choose the numbers based on the experience gained from experiments. It is a balance between efficiency and accuracy for solving the equations.

4.7 Smoothing Term

In this section, the smoothing term F_s in the goal function is described. The objective of the smoothing term is to increase the smoothness of the surface and discourage self-intersection. Following [20], the smoothing term in Equation 3.1 is defined by

$$F_s = \frac{1}{n}\sum_{i=1}^{n} V(P_i)^T V(P_i), \tag{4.24}$$

where P_i, $i = 1, 2, \ldots, n$, are the control points and $V(\cdot)$ is a discretized version of Laplacian. $V(\cdot)$ is defined as:

$$V(P_i) = \frac{1}{deg(P_i)}\sum_{j} U(P_{n(i,j)}) - U(P_i), \tag{4.25}$$

and $U(\cdot)$ is defined as:

$$U(P_i) = \frac{1}{deg(P_i)}\sum_{j} P_{n(i,j)} - P_i, \tag{4.26}$$

where $deg(P_i)$ is the degree of P_i and $P_{n(i,j)}$ is the j^{th} neighbor of P_i.

It is not trivial to choose an appropriate value for the coefficient λ for F_s. If λ is too small, the term will have little influence and self-intersection may occur. On the other hand, if λ is too large, the fitting result may not be acceptable since the fitting surface will be too rigid to give small fitting errors. In our experiments, the initial value for λ is set to be 0.001. As the optimization proceeds, λ is reduced gradually at different rates for different target shapes.

4.8 Local Subdivision

When the result of the approximation is not as good as expected due to the lack of the degree of freedom provided by the current control points, new control points need to be inserted. Instead of applying the subdivision rule to all the triangles, we perform subdivisions only to the triangles that have large errors. This is referred to as *local subdivision*.

During the fitting process, the error between the subdivision surface and the target shape is measured so that regions with relatively large fitting errors are identified. The faces in these regions are then subdivided in a 1-to-4 manner (see Figure 4.4). To avoid undesirable T-vertices, the neighboring triangles are split, following the Red-Green Splitting scheme [66].

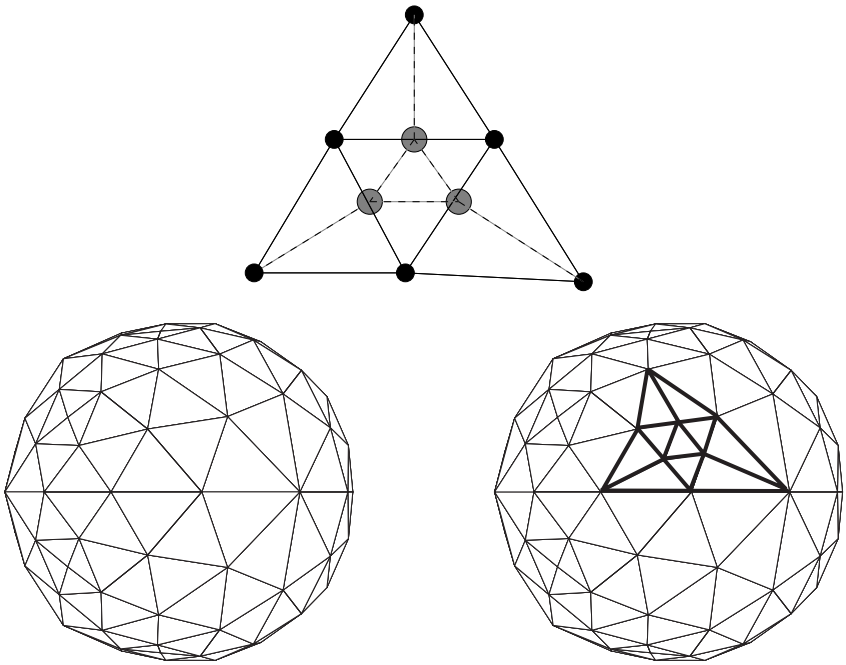


Figure 4.4: One triangle is split into four triangles. Neighboring triangles are also split.

After local subdivision, if the fitting errors in most other regions are already acceptable, only the newly added points and their neighboring vertices are treated as variables and optimized. This saves computation time by avoiding solving a much larger linear system of equations. Since the optimized problem has been altered when new control points are inserted via local subdivisions, our fitting process is indeed a multi-staged optimization.

Chapter 5

Optimization: Surface Fitting

In this chapter, we first describe some fundamentals in the field of optimization [45–47, 56, 58, 59, 67–70]. Then, we will relate them with different distance optimization methods used in the problem of fitting subdivision surfaces to unorganized points.

5.1 Optimization Basics

In an optimization problem, values of the valuables in the goal function are being modified during the optimization process such that the goal function is minimized. In this section, several approaches for tackling optimization problems are described. Here, the goal function $f(x)$ is assumed to be twice-differentiable.

5.1.1 Necessary and Sufficient Conditions for a Local Minimum

In this section, the necessary and sufficient conditions for a local minimum are discussed. We are dealing with unconstrained optimizations, meaning that no additional constraints are made on the variables.

Necessary Condition: *Given a goal function $f(x)$, $\bigtriangledown f(x^*)$ must be zero if $f(x^*)$ is a local minimum of $f(x)$.*

Using Taylor's expansion, we can expand $f(x)$ around x^*:

$$f(x^* + \delta) = f(x^*) + \delta^T \bigtriangledown f(x^* + \theta\delta), \tag{5.1}$$

where δ is a modification vector and $\theta \in (0, 1)$.

If $\bigtriangledown f(x^*) \neq 0$, by continuity, there exists $\theta \neq 0$ such that $\bigtriangledown f(x^* + \theta\delta) \neq 0$. Then, there exists a modification vector δ such that $\delta^T \bigtriangledown f(x^* + \theta\delta) < 0$. In that case, $f(x^* + \delta) - f(x^*) < 0$ and it means that $f(x^*)$ is not a local minimum.
However, it cannot be concluded that $f(x^*)$ is a local minimum of $f(x)$ if it is just known that $\bigtriangledown f(x^*) = 0$. $\bigtriangledown f(x^*) = 0$ is also true when $f(x^*)$ is a local maximum or a saddle point of $f(x)$.

Sufficient Condition: *Given a goal function $f(x)$, if $\bigtriangledown f(x^*) = 0$ and $\bigtriangledown^2 f(x^*)$ is positive definite, then $f(x^*)$ is a local minimum of $f(x)$.*

Using Taylor's expansion again, and one more term is expanded:

$$f(x^* + \delta) = f(x^*) + \delta^T \bigtriangledown f(x^*) + \frac{1}{2}\delta^T \bigtriangledown^2 f(x^* + \theta\delta)\delta, \tag{5.2}$$

where δ is a modification vector and $\theta \in (0, 1)$.

Since $\bigtriangledown f(x^*) = 0$,

$$f(x^* + \delta) - f(x^*) = \frac{1}{2}\delta^T \bigtriangledown^2 f(x^* + \theta\delta)\delta. \tag{5.3}$$

Since $\bigtriangledown^2 f(x^*)$ is positive definite, and by continuity, $\bigtriangledown^2 f(x^* + \theta\delta)$ is positive definite. Then, $\delta^T \bigtriangledown^2 f(x^* + \theta\delta)\delta > 0, \forall\delta$. Consequently, $f(x^* + \delta) - f(x^*) > 0, \forall\delta$. In other words, $f(x^*)$ is a local minimum of $f(x)$.

5.1.2 The Newton Method

Given a goal function $f(x)$ and the current value x_c, which is assumed to be near a local minimum of $f(x)$, the idea of the Newton method is to compute the next iterate, x_+, as a minimizer of the local quadratic model $m_c(x)$ (an approximant obtained by Taylor's expansion up to the 2^{nd} order term) of $f(x)$ about x_c.

$$m_c(x) = f(x_c) + \bigtriangledown f(x_c)^T(x - x_c) + \frac{1}{2}(x - x_c)^T \bigtriangledown^2 f(x_c)(x - x_c). \tag{5.4}$$

Computing the gradient of Equation 5.4, we have:

$$\bigtriangledown m_c(x) = \bigtriangledown f(x_c) + \bigtriangledown^2 f(x_c)(x - x_c). \tag{5.5}$$

Then, x_+ is computed as the solution of $\bigtriangledown m_c(x) = 0$. So,

$$x_+ = x_c - (\bigtriangledown^2 f(x_c))^{-1} \bigtriangledown f(x_c). \tag{5.6}$$

In practice, the inverse of the Hessian, $(\bigtriangledown^2 f(x_c))^{-1}$, is not computed and the step s, $x_+ - x_c$, is solved using the equation:

$$\bigtriangledown^2 f(x_c)s = - \bigtriangledown f(x_c). \tag{5.7}$$

After that,

$$x_+ = x_c + s. \tag{5.8}$$

If x_c is too far away from a local minimizer, $\bigtriangledown^2 f(x_c)$ can have negative eigenvalues and is not positive definite. As described in the previous section, the sufficient condition for a local minimum is that the Hessian needs to be positive definite. Therefore, when x_c is too far away from a local minimizer, the quadratic model may not have local minima and the Newton method may not converge to a local minimum. For a value x_c that is close to a local minimum, the Newton method gives quadratic convergence.

5.1.3 The Gauss-Newton Method

In the Newton method, the Hessian, which can be expensive to compute, is required for formulating the local quadratic model. In this section, another method for tackling the optimization problem, called the Gauss-Newton method, is described.

Consider a goal function $f(x)$ of a nonlinear least squares problem that is in the form:

$$f(x) = \frac{1}{2} \sum_k r_k(x)^T r_k(x). \tag{5.9}$$

In the Newton method, the Hessian of the goal function is computed as follows:

$$\bigtriangledown^2 f(x) = \sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T + \sum_k r_k(x) \bigtriangledown^2 r_k(x) \tag{5.10}$$

In the Guass-Newton method, the second-order term, $\bigtriangledown^2 r_k(x)$, is discarded and the Hessian is approximated as follows:

$$\bigtriangledown^2 f(x) \approx \sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T. \tag{5.11}$$

The process of the Gauss-Newton method is as the same as that of the Newton method with the Hessian, $\bigtriangledown^2 f(x_c)$, replaced by $\sum_k \bigtriangledown r_k(x_c) \bigtriangledown r_k(x_c)^T$.
For zero residual problems, $r_k(x) = 0$ and therefore the second-order term in Equation 5.10 vanishes. In these cases, the Gauss-Newton method is just identical to the

Newton method. The motivation for the Gauss-Newton method is that the second-order term might be negligible for small residual problems.

The advantage of the Gauss-Newton method is that the computational cost is smaller when comparing with the Newton method since the second-order terms are discarded in the Gauss-Newton method. The Gauss-Newton method works well for zero or small residual problems. For zero residual problems, like the Newton method, the Gauss-Newton method exhibits quadratic convergence.
However, the Gauss-Newton method works poorly for large residual problems because the Hessian is poorly approximated by simply discarding the second-order terms in those situations.

5.1.4 The Steepest Descent Method

In the Newton method and the Gauss-Newton method, the Hessian of the goal function or its approximant is required. There exists a method, called the steepest descent method, in which no computation of the Hessian is needed. Only the gradient of the goal function is involved.

Given a goal function $f(x)$ and the current value x_c, the next value x_+ is being found such that $f(x_+) < f(x_c)$. A direction is called a descent direction if the goal function value $f(x)$ decreases when x is displaced along that direction for a reasonably small step size. At x_c, the steepest descent direction is $- \bigtriangledown f(x_c)$, which is the negative of the gradient at x_c. The direction $- \bigtriangledown f(x_c)$ is really a descent direction if x_c is not a stationary point (a stationary point refers to a local minimum, a local maximum or a saddle point; and x_c is a stationary point if and only if $\bigtriangledown f(x_c) = 0$).

The iterative formula for the steepest descent method is as follows:

$$x_+ = x_c - \alpha \bigtriangledown f(x_c), \tag{5.12}$$

where α, to be determined by a line search algorithm, is the step size along the steepest descent direction.

Although there is no need to compute or approximate the Hessian, the steepest descent method can be recognized as the simplest form of the Newton-type method where the

iterative formula can be obtained by replacing $\bigtriangledown^2 f(x_c)$ on the left hand side of Equation 5.6 by $\frac{1}{\alpha}I$, where I is the identity matrix. From another point of view, the Hessian is approximated by $\frac{1}{\alpha}I$.

The steepest descent method is simple for implementation but is known to give only linear convergence [71].

5.1.5 The Levenberg-Marquardt Method - a Trust Region Method

As described in the previous subsections, the three methods (the Newton method, the Gauss-Newton method and the steepest descent method) have their own advantages and shortcomings. When the current estimate is close to a local minimum, the Newton method gives quadratic convergence. However, computation the Hessian, which is often expensive, is required in the Newton method. The Gauss-Newton method reduces the computation time by discarding the second-order terms when approximating the Hessian. But, the Gauss-Newton method works satisfactorily for zero residual problems and small residual problems only. The steepest descent method does not require the computation of the Hessian and does converge for large residual problems. However, the steepest descent method only has linear convergence. In this subsection, a regularized version of the Gauss-Newton method, so called the Levenberg-Marquardt Method (the LM method) [47, 59, 67, 68, 72], which is supposed to be robust enough to ensure global convergence, is described.

The essence of the LM method is that the local model is only trusted within a neighborhood that falls within a limited range around the current point x_c. It means that the step size s for an iteration is bounded, and this is formally defined by the constraint $\|s\| \leq \triangle_k$. The value of $\triangle_k$ depends on the degree of the agreement between the local model and the actual goal function. By using the method of Lagrange multipliers, the original constrained optimization has been transformed into an unconstrained optimization. The step s can be computed by solving the following equation:

$$(\sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T + \nu_c I)s = - \bigtriangledown f(x_c), \tag{5.13}$$

where ν_c is the LM parameter which is adjusted at each iteration.
Comparing the above equation with that of the Newton method (Equation 5.6), it can

be noticed that the Hessian of the goal function is approximated as follows:

$$\bigtriangledown^2 f(x) \approx \sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T + \nu_c I \tag{5.14}$$

For all values of ν_c, the matrix $\sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T + \nu_c I$ is positive definite. When ν_c is close to zero, the matrix is dominated by $\sum_k \bigtriangledown r_k(x) \bigtriangledown r_k(x)^T$, which makes the LM method close to the Gauss-Newton method. When ν_c is a large value, the matrix is dominated by the identity matrix I, which makes the step close to the steepest descent direction.

During the optimization process, a value, called the gain ratio, is monitored. The gain ratio gr is the ratio of the decrease in the actual goal function to the decrease predicted by the local model.

$$gr = \frac{f(x_c) - f(x_c + s)}{L(0) - L(s)} \tag{5.15}$$

where L is the local quadratic model used to approximate the goal function. After some mathematical manipulations, the gain ratio gr becomes:

$$gr = \frac{2(f(x_c) - f(x_c + s))}{s^T(\nu_c s - \bigtriangledown f(x_c))} \tag{5.16}$$

If the gain ratio is small, which means that the current model is a poor approximation to the goal function, ν_c will be increased so that the next step is closer to the steepest descent direction and the step size is reduced. If the gain ratio is high, which means that the current model is a good approximation to the goal function, ν_c will be decreased so that the next step is closer to a Gauss-Newton step, which converges much faster. A particular way to modify the value of ν_c according to gr is described in [69]. By monitoring the agreement between the local model and the actual goal function, this approach attempts to share the advantages of both the steepest descent method and the Gauss-Newton method. Comparing with the Gauss-Newton method, both the direction and the step size are modified when the LM method is applied. In the optimization field, the LM method is often implemented as a trust-region strategy. The size of the trust-region depends on the agreement between the current model and the actual goal function.

The effectiveness of the LM method in fitting subdivision surfaces to point clouds is observed in the experiments in section 6 in the next chapter.

5.1.6 The Armijo Method - a Line Search Method

For guaranteeing global convergence, line search method [47, 59, 67, 68, 72] is an alternative approach to trust region method described in the previous section.

After a moving direction has been determined by a certain optimization method such as PDM, SDM or TDM, the step size needs to be decided explicitly by performing a line search to guarantee convergence rather than just having steps of fixed length all the time. The Armijo method provides one way to perform this task.

Briefly speaking, the following condition, called the sufficient condition [47,68], needs to be satisfied:

$$f(x_c) - f(x_c + \alpha\delta) \geq -\alpha_2 \alpha \bigtriangledown f(x_c)^T \delta, \tag{5.17}$$

where α is the step size, δ is the moving direction and α_2 is a parameter smaller than 1.

The step size α can be set to 1 initially and is halved until the above sufficient condition is satisfied.

With the use of the step size control, the stability of the optimization process is better. Similar to the LM method, extra goal function and gradient evaluations are required and these increase the computational time. Different from the LM method in which both the direction and the step size are modified, only the step size is modified in the Armijo method.

The effectiveness of the Armijo method in fitting subdivision surfaces to point clouds is observed in section 6 in the next chapter.

5.2 Fitting Subdivision Surface to Point Cloud – from the Viewpoint of Optimization

In this section, we would like to relate the optimization basics described in the previous section to various methods for solving the subdivision surface fitting problem.

First, some notations are defined for ease of the discussion. A fitting subdivision surface $\mathcal{S}$ is given for fitting the target shape Γ. $\mathcal{S}$ is defined by control points $P_i, i = 1, 2, \ldots, n$ while Γ is assumed to be twice differentiable. $P(u_k, v_k), k = 1, 2, \ldots, N$, which are linear combinations of the control points P_i, are sample points on $\mathcal{S}$ and $R(s_k, t_k)$ are their corresponding foot points on Γ, where u_k, v_k and s_k, t_k are the surface parameters of the sample points and the foot points respectively. Consider that the control points P_i of $\mathcal{S}$ are modified with a displacement vector $\mathcal{D} = (D_1^T, D_2^T, \ldots, D_n^T)^T$ in each optimization step. Then the sample points on the modified surface are $P(\mathcal{D}; u_k, v_k)$ and their corresponding foot points on Γ are $R(s(k; \mathcal{D}), t(k; \mathcal{D}))$. For clarity, $P(\mathcal{D}; u_k, v_k)$ and $R(s(k; \mathcal{D}), t(k; \mathcal{D}))$ are denoted by P and R respectively. Sometimes, $P - R$ is denoted by E_k and $\frac{1}{2}E_k^T E_k$ is denoted by f_k.

In the subdivision surface fitting problem, $\mathcal{D}$ needs to be computed such that the goal function

$$f = \frac{1}{2}\sum_{k=1}^{N}(P - R)^T(P - R) \tag{5.18}$$

is minimized, and subjected to the constraints:

$$(P - R)^T \frac{\partial R}{\partial s_k} = 0 \tag{5.19}$$

and

$$(P - R)^T \frac{\partial R}{\partial t_k} = 0. \tag{5.20}$$

Note that the constraints are added since they are the necessary conditions for a minimum of f, which can easily be verified by differentiation. Geometrically, it means that the vector $P - R$ must be normal to the target shape at R.
Since the gradient of f_k is required in several methods, we include its computation here.

$$\begin{aligned} \bigtriangledown_{\mathcal{D}} f_k &= \left(\frac{\partial P}{\partial \mathcal{D}} - \bigtriangledown_{\mathcal{D}} s_k \frac{\partial R^T}{\partial s_k} - \bigtriangledown_{\mathcal{D}} t_k \frac{\partial R^T}{\partial t_k}\right) E_k \\ &= \frac{\partial P}{\partial \mathcal{D}} E_k \text{ (due to the constraints 5.19 and 5.20)} \end{aligned} \tag{5.21}$$

Depending on the context, $\bigtriangledown_{\mathcal{D}} f_k$ will also be written as $\frac{\partial P}{\partial \mathcal{D}}^T (P - R)$.

5.2.1 PDM – the Gradient Descent Method

In this subsection, we are going to show that PDM is the gradient descent method. In PDM, the PD error function is used.

The derivation starts from the quadratic local model (obtained by expanding up to the second order term using Taylor's expansion) at the current point P_c. In the derivation below, $\bigtriangledown^2_{\mathcal{D}} f(x_c)$ is replaced by the matrix $\frac{\partial P^T}{\partial \mathcal{D}} \frac{\partial P}{\partial \mathcal{D}}$.

$$
\begin{aligned}
& f_k(P_c) + \bigtriangledown_{\mathcal{D}} f_k(P_c)^T \mathcal{D} + \frac{1}{2} \mathcal{D}^T \bigtriangledown^2_{\mathcal{D}} f_k(P_c) \mathcal{D} \\
= \; & f_k(P_c) + \bigtriangledown_{\mathcal{D}} f_k(P_c)^T \mathcal{D} + \frac{1}{2} \mathcal{D}^T \frac{\partial P^T}{\partial \mathcal{D}} \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} \\
= \; & \frac{1}{2}(P_c - R_c)^T (P_c - R_c) + (P_c - R_c)^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} + \frac{1}{2} \mathcal{D}^T \frac{\partial P^T}{\partial \mathcal{D}} \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} \\
= \; & \frac{1}{2}(P_c + \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} - R_c)^T (P_c + \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} - R_c) \\
= \; & \frac{1}{2}(P(P_c; \mathcal{D}) - R_c)^T (P(P_c; \mathcal{D}) - R_c) \\
= \; & \frac{1}{2} \text{PD error term (Equation 2.2)}, \qquad (5.22)
\end{aligned}
$$

where P_c are the points for the current iteration and R_c are the foot points of P_c.

Hence PDM is the gradient descent method. This is a typical optimization approach for solving a separable nonlinear least squares problem and is known to have linear convergence [71].

It is possible to show that PDM converges linearly in another way:

Given a fitting subdivision surface $\mathcal{S}$ defined by n control points $P_i \in E^3$ and m data points $R_k \in E^3$, PDM minimizes the goal function $f(\mathcal{P}, \mathcal{U})$ with variables $\mathcal{P}$ and $\mathcal{U}$ in the Euclidean space E^{3n+2m} spanned by $\mathcal{P}$ and $\mathcal{U}$, where $\mathcal{P} = \{P_i\}_{i=1}^n$ are the control points of the fitting subdivision surface and $\mathcal{U} = \{s_k, t_k\}_{k=1}^m$ are the parameter values associated the R_k.

PDM has the following two steps that are carried out in iteration: (1) For fixed parameter values $\mathcal{U}_0 = \{s_{k,0}, t_{k,0}\}$ and control points $\mathcal{P}_0 = \{P_{i,0}\}$, the control points $\mathcal{P}_1 = \{P_{i,1}\}$ are found such that the quadratic function $f(\mathcal{P}, \mathcal{U}_0)$ is minimized. This is

done by solving a linear system of equations; (2) For the control points $\mathcal{P}_1$ produced in step 1, the parameter values $\mathcal{U}_1 = \{s_{k,1}, t_{k,1}\}$ are found such that the error function $f(\mathcal{P}_1, \mathcal{U})$ is minimized. Note that $f(\mathcal{P}, \mathcal{U})$ is not, in general, quadratic in $\mathcal{U}$; therefore one normally computes the foot points R_k on the target shape for the optimization in step (2).

The above steps of PDM can also be formulated as follows. First we need some notation. The space $E^{3n+2m} = \{W\}$, where $W = (\mathcal{P}, \mathcal{U})^T$, can be decomposed as the direct sum of subspace $E_P^{3n} = \{\mathcal{P}\}$ of dimensions $3n$ and subspace $E_U^{2m} = \{\mathcal{U}\}$ of dimensions $2m$. Let $\mathbf{e}_j \in E^{3n+2m}$ be the j-th unit basis vector, $j = 1, 2, \ldots, 3n + 2m$, i.e. all components of $\mathbf{e}_j$ are zero, except that its j-th component is 1. Let $Y_P = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{3n}\}$ be the basis vectors spanning E_P^{3n}. Let $Y_U = \{\mathbf{e}_{3n+1}, \mathbf{e}_{3n+2}, \ldots, \mathbf{e}_{3n+2m}\}$ be the basis vectors spanning E_U^{2m}. Let $\mathcal{L}(Y)$ be the linear space spanned by k linearly independent vectors $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$, where the $\mathbf{y}_\ell \in E^{3n+2m}$, $\ell = 1, 2, \ldots, k$. Let $\langle Q_0; \mathcal{L}(Y)\rangle \subset E^{3n+2m}$ denote the k-dimensional Euclidean subspace obtained by attaching the linear space $\mathcal{L}(Y)$ to a point $Q_0 \in E^{3n+2m}$, i.e. $\langle Q_0; \mathcal{L}(Y)\rangle = \{Q_0 + YX\}$, where $X = (x_1, x_2, \ldots, x_k)^T \in E^k$. From a starting point $W_0 = (\mathcal{P}_0, \mathcal{U}_0)^T \in E^{3n+2m}$, PDM first computes a minimizer $\tilde{W}_1 = (\mathcal{P}_1, \mathcal{U}_0)^T$ of $f(\mathcal{P}, \mathcal{U}_0)$ in the subspace $\langle W_0; \mathcal{L}(Y_P)\rangle$. Then PDM computes a minimizer $W_1 = (\mathcal{P}_1, \mathcal{U}_1)^T$ of $f(\mathcal{P}_1, \mathcal{U})$ in the subspace $\langle \tilde{W}_1; \mathcal{L}(Y_U)\rangle$. Then, from W_1, the above two steps are iterated to compute $W_2 = (\mathcal{P}_2, \mathcal{U}_2)^T$, and so on.

To study the convergence rate of PDM, it suffices to consider PDM in the neighborhood of a local minimizer of $f(\mathcal{P}, \mathcal{U})$; without loss of generality, suppose that the minimizer is at the origin. It is well known from optimization theory that we just need to consider the Hessian H of $f(\mathcal{P}, \mathcal{U})$ at the origin. Suppose the minimizer under consideration is strict, i.e. all eigenvalues of H are positive. We consider the application of PDM to optimizing the quadratic function

$$f_H(\mathcal{P}, \mathcal{U}) = (\mathcal{P}, \mathcal{U}) H (\mathcal{P}, \mathcal{U})^T.$$

The minimizer $\tilde{W}_1$ of $f_H(\mathcal{P}, \mathcal{U}^0)$ in the subspace $\langle W_0; \mathcal{L}(Y_P)\rangle$ can be found by minimizing the quadratic function

$$f_{H,\mathcal{P}_0} = [W_0 + Y_P X_P]^T H [W_0 + Y_P X_P]$$

with respect to $X_P \in E^{3n}$. It is easy to show that

$$\tilde{W}_1 = [I - Y_P (Y_P^T H Y_P)^{-1} Y_P H] W_0.$$

Similarly, starting from $\tilde{W}_1$, the minimizer of f_H in the subspace $\langle \tilde{W}_1; Y_U \rangle$ is given by

$$W_1 = [I - Y_T(Y_T^T H Y_T)^{-1} Y_T H]\tilde{W}_1 = MW_0$$

where $M = [I - Y_T(Y_T^T H Y_T)^{-1} Y_T H][I - Y_P(Y_P^T H Y_P)^{-1} Y_P H]$. Iterating the above two steps, we obtain

$$W_j = M^j W_0, \quad j = 0, 1, 2 \ldots,$$

where $W_j = (\mathcal{P}_j, \mathcal{U}_j)^T$ is the result produced at the j-th iteration of PDM. Hence, PDM has a linear convergence. Iteration $(\mathcal{P}_j, \mathcal{U}_j)^T = M(\mathcal{P}_{j-1}, \mathcal{U}_{j-1})^T$ does not produce $(\mathcal{P}_j, \mathcal{U}_j) = 0$ in a finite number of steps if $(\mathcal{P}_0, \mathcal{U}_0)$ is not in the null-space of M. The actual convergence speed depends on the eigen-structure of H as well as the relationship between $(\mathcal{P}_0, \mathcal{U}_0)$ and the eigenvectors of H.

5.2.2 TDM – the Gauss-Newton Method

In this section, we deal with the Gauss-Newton method. First, the goal function is expressed as follows:

$$f(x) = \frac{1}{2} \sum_k r_k^2(x), \tag{5.23}$$

where $r_k(x) = \|P - R\|$.

It is clear that $f_k(P_c) = \frac{1}{2} r_k(P_c)^2$, and

$$\begin{aligned} \bigtriangledown_{\mathcal{D}} f_k &= r_k \bigtriangledown_{\mathcal{D}} r_k \\ \bigtriangledown_{\mathcal{D}} r_k &= \frac{\bigtriangledown_{\mathcal{D}} f_k}{r_k}. \end{aligned} \tag{5.24}$$

From Equation 5.21, we have:

$$\begin{aligned} \bigtriangledown_{\mathcal{D}} r_k &= \frac{\frac{\partial P}{\partial \mathcal{D}} E_k}{r_k} \\ \bigtriangledown_{\mathcal{D}} r_k &= \frac{\partial P}{\partial \mathcal{D}} \frac{E_k}{r_k} \\ \bigtriangledown_{\mathcal{D}} r_k &= \frac{\partial P}{\partial \mathcal{D}} \frac{(P - R)}{\|P - R\|} \\ \bigtriangledown_{\mathcal{D}} r_k &= \frac{\partial P}{\partial \mathcal{D}} \mathcal{N}, \end{aligned} \tag{5.25}$$

where $\mathcal{N}$ is the normal vector at the foot point on the target shape for the k^{th} sample point.

Now we move on to show that TDM is the Gauss-Newton method. The derivation starts from the quadratic local model (obtained by expanding up to the second order term using Taylor's expansion) at the current point P_c. As discussed in the previous section, $\bigtriangledown^2_{\mathcal{D}} f_k(P_c)$ is approximated by $\bigtriangledown_{\mathcal{D}} r_k(P_c) \bigtriangledown_{\mathcal{D}} r_k(P_c)^T$, which is $\frac{\partial P}{\partial \mathcal{D}} \mathcal{N}\mathcal{N}^T \frac{\partial P}{\partial \mathcal{D}}^T$. The substitution is done at the second line in the following derivation.

$$
\begin{aligned}
& f_k(P_c) + \bigtriangledown_{\mathcal{D}} f_k(P_c)^T \mathcal{D} + \frac{1}{2} \mathcal{D}^T \bigtriangledown^2_{\mathcal{D}} f_k(P_c) \mathcal{D} \\
= \; & f_k(P_c) + \bigtriangledown_{\mathcal{D}} f_k(P_c)^T \mathcal{D} + \frac{1}{2} \mathcal{D}^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{N}\mathcal{N}^T \frac{\partial P^T}{\partial \mathcal{D}} \mathcal{D} \\
= \; & \frac{1}{2}(P_c - R_c)^T (P_c - R_c) + (P_c - R_c)^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} + \frac{1}{2} \mathcal{D}^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{N}\mathcal{N}^T \frac{\partial P^T}{\partial \mathcal{D}} \mathcal{D} \\
= \; & \frac{1}{2}(P_c - R_c)^T \mathcal{N}\mathcal{N}^T (P_c - R_c) + (P_c - R_c)^T \mathcal{N}\mathcal{N}^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{D} + \frac{1}{2} \mathcal{D}^T \frac{\partial P}{\partial \mathcal{D}} \mathcal{N}\mathcal{N}^T \frac{\partial P^T}{\partial \mathcal{D}} \mathcal{D} \\
= \; & \frac{1}{2}(P + \frac{\partial P^T}{\partial \mathcal{D}} \mathcal{D} - R_c)^T \mathcal{N}\mathcal{N}^T (P + \frac{\partial P^T}{\partial \mathcal{D}} \mathcal{D} - R_c) \\
= \; & \frac{1}{2}(P(P_c; \mathcal{D}) - R_c)^T \mathcal{N}\mathcal{N}^T (P(P_c; \mathcal{D}) - R_c) \\
= \; & \frac{1}{2}((P(P_c; \mathcal{D}) - R_c)^T \mathcal{N})^2 \\
= \; & \frac{1}{2}\text{TD error term (Equation 2.3)}, \qquad (5.26)
\end{aligned}
$$

where P_c are the points for the current iteration, R_c are the foot points of P_c and $\mathcal{N}$ are the normal vectors at R_c.

Hence TDM is the Gauss-Newton method. In surface fitting problem, TDM works well when the initial positions of the control points are close to the optimal positions and the surface is really capable of fitting the target data well. In other words, it works well for zero or small residual problems. However, TDM is not stable around large curvature regions since the discarded terms in the approximant of the Hessian of the goal function are indeed not negligible.

5.2.3 SDM – the Newton Method

In this subsection, we are going to show that SDM is the Newton method. In SDM, the SD error function is used.

We first compute the derivative of $\bigtriangledown_{\mathcal{D}} f_k$ (Equation 5.21) and get the Hessian.

$$\begin{aligned}\bigtriangledown^2_{\mathcal{D}} f_k &= \left(\frac{\partial P}{\partial \mathcal{D}}^T - \frac{\partial R}{\partial s_k}\bigtriangledown^T_{\mathcal{D}} s_k - \frac{\partial R}{\partial t_k}\bigtriangledown^T_{\mathcal{D}} t_k\right)\frac{\partial P}{\partial \mathcal{D}} + E_k^T\frac{\partial^2 P}{\partial \mathcal{D}^2} \\ &= \left(\frac{\partial P^T}{\partial \mathcal{D}} - \frac{\partial R}{\partial s_k}\bigtriangledown^T_{\mathcal{D}} s_k - \frac{\partial R}{\partial t_k}\bigtriangledown^T_{\mathcal{D}} t_k\right)\frac{\partial P}{\partial \mathcal{D}}, \end{aligned} \tag{5.27}$$

where $E_k^T\frac{\partial^2 P}{\partial \mathcal{D}^2} = 0$ since P can be expressed as a linear combination of $\mathcal{D}$.

From the constraints 5.19 and 5.20, we have the following two equations:

$$\begin{aligned} 0 &= \bigtriangledown_{\mathcal{D}}\left((P-R)^T\frac{\partial R}{\partial s_k}\right) \\ &= \left(\bigtriangledown_{\mathcal{D}} s_k\frac{\partial^2 R^T}{\partial s_k^2} + \bigtriangledown_{\mathcal{D}} t_k\frac{\partial^2 R^T}{\partial t_k \partial s_k}\right)(P-R) + \\ &\quad \left(\frac{\partial P}{\partial \mathcal{D}} - \bigtriangledown_{\mathcal{D}} s_k\frac{\partial R^T}{\partial s_k} - \bigtriangledown_{\mathcal{D}} t_k\frac{\partial R^T}{\partial t_k}\right)\frac{\partial R}{\partial s_k} \end{aligned} \tag{5.28}$$

and

$$\begin{aligned} 0 &= \bigtriangledown_{\mathcal{D}}\left((P-R)^T\frac{\partial R}{\partial t_k}\right) \\ &= \left(\bigtriangledown_{\mathcal{D}} t_k\frac{\partial^2 R^T}{\partial t_k^2} + \bigtriangledown_{\mathcal{D}} s_k\frac{\partial^2 R^T}{\partial s_k \partial t_k}\right)(P-R) + \\ &\quad \left(\frac{\partial P}{\partial \mathcal{D}} - \bigtriangledown_{\mathcal{D}} t_k\frac{\partial R^T}{\partial t_k} - \bigtriangledown_{\mathcal{D}} s_k\frac{\partial R^T}{\partial s_k}\right)\frac{\partial R}{\partial t_k} \end{aligned} \tag{5.29}$$

Without loss of generality, suppose that $R(s_k, t_k)$ is a local regular parameterization of the target surface Γ such that $\frac{\partial R}{\partial s_k}$ and $\frac{\partial R}{\partial t_k}$ are the unit vectors along the principal directions of the target surface at R. Then, we have $\frac{\partial R^T}{\partial s_k}\frac{\partial R}{\partial t_k} = \frac{\partial R^T}{\partial t_k}\frac{\partial R}{\partial s_k} = 0$. It follows that

$$\begin{aligned} 0 &= \left(\bigtriangledown_{\mathcal{D}} s_k\frac{\partial^2 R^T}{\partial s_k^2} + \bigtriangledown_{\mathcal{D}} t_k\frac{\partial^2 R^T}{\partial t_k \partial s_k}\right)(P-R) + \\ &\quad \left(\frac{\partial P}{\partial \mathcal{D}} - \bigtriangledown_{\mathcal{D}} s_k\frac{\partial R^T}{\partial s_k}\right)\frac{\partial R}{\partial s_k} \end{aligned} \tag{5.30}$$

and

$$\begin{aligned} 0 &= \left(\bigtriangledown_{\mathcal{D}} t_k\frac{\partial^2 R^T}{\partial t_k^2} + \bigtriangledown_{\mathcal{D}} s_k\frac{\partial^2 R^T}{\partial s_k \partial t_k}\right)(P-R) + \\ &\quad \left(\frac{\partial P}{\partial \mathcal{D}} - \bigtriangledown_{\mathcal{D}} t_k\frac{\partial R^T}{\partial t_k}\right)\frac{\partial R}{\partial t_k} \end{aligned} \tag{5.31}$$

From (5.30) and (5.31), we get

$$\bigtriangledown_{\mathcal{D}} s_k = -\frac{\frac{\partial P}{\partial \mathcal{D}}\frac{\partial R}{\partial s_k} + \frac{\partial^2 R^T}{\partial t_k \partial s_k}(P-R)\bigtriangledown_{\mathcal{D}} t_k}{\frac{\partial^2 R^T}{\partial s_k^2}(P-R) - \frac{\partial R^T}{\partial s_k}\frac{\partial R}{\partial s_k}} \tag{5.32}$$

and

$$\bigtriangledown_{\mathcal{D}} t_k = -\frac{\frac{\partial P}{\partial \mathcal{D}}\frac{\partial R}{\partial t_k} + \frac{\partial^2 R^T}{\partial s_k \partial t_k}(P-R)\bigtriangledown_{\mathcal{D}} s_k}{\frac{\partial^2 R^T}{\partial t_k^2}(P-R) - \frac{\partial R^T}{\partial t_k}\frac{\partial R}{\partial t_k}} \tag{5.33}$$

Now, we make the following substitutions into (5.32) and (5.33): $\frac{\partial R}{\partial s_k} = \mathcal{T}_1$, $\frac{\partial R}{\partial t_k} = \mathcal{T}_2$, $\frac{\partial^2 R}{\partial s_k^2} = \kappa_1 \mathcal{N}$, $\frac{\partial^2 R}{\partial t_k^2} = \kappa_2 \mathcal{N}$, $\|P-R\|_2 = d$, where κ_1, κ_2 are the principal curvatures at R, $\mathcal{T}_1$, $\mathcal{T}_2$ are the unit tangent vectors along the principal directions at R and $\mathcal{N}$ is the unit normal vector at R. After the substitutions, we have:

$$\bigtriangledown_{\mathcal{D}} s_k = -\frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1 + \frac{\partial^2 R^T}{\partial t_k \partial s_k}(P-R)\bigtriangledown_{\mathcal{D}} t_k}{d\kappa_1 - 1} \tag{5.34}$$

and

$$\bigtriangledown_{\mathcal{D}} t_k = -\frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2 + \frac{\partial^2 R^T}{\partial s_k \partial t_k}(P-R)\bigtriangledown_{\mathcal{D}} s_k}{d\kappa_2 - 1} \tag{5.35}$$

Since $\frac{\partial R^T}{\partial s_k}\frac{\partial R}{\partial t_k} = 0$, differentiating with respect to s_k yields $\frac{\partial^2 R^T}{\partial s_k^2}\frac{\partial R}{\partial t_k} + \frac{\partial^2 R^T}{\partial s_k \partial t_k}\frac{\partial R}{\partial s_k} = 0$. Therefore,

$$\frac{\partial^2 R^T}{\partial s_k \partial t_k}\frac{\partial R}{\partial s_k} = -\frac{\partial^2 R^T}{\partial s_k^2}\frac{\partial R}{\partial t_k} = -\kappa_1 \mathcal{N}^T \mathcal{T}_2 = 0. \tag{5.36}$$

Similarly,

$$\frac{\partial^2 R^T}{\partial s_k \partial t_k}\frac{\partial R}{\partial t_k} = -\frac{\partial^2 R^T}{\partial t_k^2}\frac{\partial R}{\partial s_k} = -\kappa_2 \mathcal{N}^T \mathcal{T}_1 = 0. \tag{5.37}$$

Substitute (5.34) and (5.35) into (5.28) and take (5.36) and (5.37) into consideration,

$$\begin{aligned}
\bigtriangledown_{\mathcal{D}}^2 f_k &= \frac{\partial P}{\partial \mathcal{D}}\frac{\partial P^T}{\partial \mathcal{D}} + \frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}}}{d\kappa_1 - 1} + \frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}}}{d\kappa_2 - 1} \\
&= \frac{\partial P}{\partial \mathcal{D}}(1 - \mathcal{T}_1\mathcal{T}_1^T - \mathcal{T}_2\mathcal{T}_2^T)\frac{\partial P^T}{\partial \mathcal{D}} + d\kappa_1\frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}}}{d\kappa_1 - 1} + d\kappa_1\frac{\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}}}{d\kappa_2 - 1} \\
&= \frac{\partial P}{\partial \mathcal{D}}\mathcal{N}\mathcal{N}^T\frac{\partial P^T}{\partial \mathcal{D}} + \frac{d}{d-\rho_1}\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}} + \\
&\quad \frac{d}{d-\rho_2}\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}},
\end{aligned} \tag{5.38}$$

where $\rho_1 = \frac{1}{\kappa_1}, \rho_2 = \frac{1}{\kappa_2}$ are the radii of curvature along the principal directions at R. Note that the fact that $\mathcal{T}_1\mathcal{T}_1^T + \mathcal{T}_2\mathcal{T}_2^T + \mathcal{N}\mathcal{N}^T = I$ has been used.

After computing $\bigtriangledown^2 f_k$, we move on to prove that SDM is the Newton method. The derivation starts from the quadratic local model (obtained by expanding up to the second order term using Taylor's expansion) at the current point P_c. We have used the fact that $(P_c - R_c)^T\mathcal{T}_1 = (P_c - R_c)^T\mathcal{T}_2 = 0$.

$$
\begin{aligned}
& f_k(P_c) + \bigtriangledown_{\mathcal{D}} f_k(P_c)^T\mathcal{D} + \frac{1}{2}\mathcal{D}^T(\bigtriangledown^2_{\mathcal{D}} f_k(P_c))\mathcal{D} \\
= \;& \frac{1}{2}(P_c - R_c)^T(P_c - R_c) + (P_c - R_c)^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} + \\
& \frac{1}{2}\mathcal{D}^T(\frac{\partial P}{\partial \mathcal{D}}\mathcal{N}\mathcal{N}^T\frac{\partial P^T}{\partial \mathcal{D}} + \frac{d}{d-\rho_1}\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}} + \\
& \frac{d}{d-\rho_2}\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}})\mathcal{D} \\
= \;& \frac{1}{2}(P_c - R_c)^T\mathcal{N}\mathcal{N}^T(P_c - R_c) + (P_c - R_c)^T\mathcal{N}\mathcal{N}^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} + \\
& \frac{1}{2}\mathcal{D}^T\frac{\partial P}{\partial \mathcal{D}}\mathcal{N}\mathcal{N}^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} + \frac{1}{2}\frac{d}{d-\rho_1}\mathcal{D}^T\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} + \\
& \frac{1}{2}\frac{d}{d-\rho_2}\mathcal{D}^T\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} \\
= \;& \frac{1}{2}((P_c + \frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} - R_c)^T\mathcal{N}\mathcal{N}^T(P_c + \frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} - R_c) + \\
& \frac{d}{d-\rho_1}\mathcal{D}^T\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_1\mathcal{T}_1^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} + \frac{d}{d-\rho_2}\mathcal{D}^T\frac{\partial P}{\partial \mathcal{D}}\mathcal{T}_2\mathcal{T}_2^T\frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D}) \\
= \;& \frac{1}{2}((P_c + \frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} - R_c)^T\mathcal{N}\mathcal{N}^T(P_c + \frac{\partial P^T}{\partial \mathcal{D}}\mathcal{D} - R_c) + \\
& \frac{d}{d-\rho_1}(P^T(P_c;\mathcal{D}) - R_c^T - P_c^T + R_c^T)\mathcal{T}_1\mathcal{T}_1^T(P(P_c;\mathcal{D}) - R_c - P_c + R_c) + \\
& \frac{d}{d-\rho_2}(P^T(P_c;\mathcal{D}) - R_c^T - P_c^T + R_c^T)\mathcal{T}_2\mathcal{T}_2^T(P(P_c;\mathcal{D}) - R_c - P_c + R_c)) \\
= \;& \frac{1}{2}((P(P_c;\mathcal{D}) - R_c)^T\mathcal{N}\mathcal{N}^T(P(P_c;\mathcal{D}) - R_c) + \\
& \frac{d}{d-\rho_1}(P(P_c;\mathcal{D}) - R_c)^T\mathcal{T}_1\mathcal{T}_1^T(P(P_c;\mathcal{D}) - R_c) + \\
& \frac{d}{d-\rho_2}(P(P_c;\mathcal{D}) - R_c)^T\mathcal{T}_2\mathcal{T}_2^T(P(P_c;\mathcal{D}) - R_c)) \\
= \;& \frac{1}{2}\left[(P(P_c;\mathcal{D}) - R_c)^T\mathcal{N}\right]^2 + \frac{1}{2}\frac{d}{d-\rho_1}\left[(P(P_c;\mathcal{D}) - R_c)^T\mathcal{T}_1\right]^2 +
\end{aligned}
$$

$$\frac{1}{2}\frac{d}{d-\rho_2}\left[(P(P_c;\mathcal{D})-R_c)^T\mathcal{T}_2\right]^2$$

$$= \frac{1}{2}\text{SD error term (Equation 2.5)}, \tag{5.39}$$

where P_c are the points for the current iteration and R_c are the foot points of P_c.

From the above derivation, we can note that the SDM formula is a quadratic approximant of (5.18). SDM is indeed the Newton method with constraints.

5.3 Trajectories of the Iterates for PDM, TDM and SDM

In this section, we study the difference in the trajectories of the iterates during the fitting processes in PDM, TDM and SDM. As described in the previous chapter, all PDM, TDM and SDM follow the idea of separation of variables in solving the nonlinear least squares problem. Foot point projection is involved in all the three methods. But, the iterates for the three methods do not have the same type of trajectory.

The commonly used method, PDM, which is also called the alternating method, is known to have linear convergence rate [33]. Figure 5.1 shows the trajectory of the iterates for PDM (in 2D case). It is a zigzag path. The minimizer of the function must be on the bold curve (which represents the feasible solution set). Vertical movements, in which only the surface parameters s are modified and the control points are kept unchanged, correspond to foot point projections. Horizontal movements, in which the surface parameters s are kept unchanged and the control points are optimized, correspond to the linear least squares problem. Vertical movements and horizontal movements are repeated alternatively until the minimizer is found.

Although foot point projection is the common step in PDM, TDM and SDM, both TDM and SDM have much faster convergence when compared with PDM. These facts are also reflected in the trajectories of the iterates for these methods. Contrast to that of PDM, the trajectories of TDM and SDM are no longer orthogonal zigzag paths. Figure 5.2 shows the trajectory of the iterates for TDM (in 2D case). Again, the minimizer of the function must be on the bold curve. Vertical movements, in which only the surface parameters s are modified and the control points are kept unchanged, correspond to foot point projections. Then, for the step that solves the linear least squares problem, the movement is along the tangent line formed at the iterate. TDM uses the

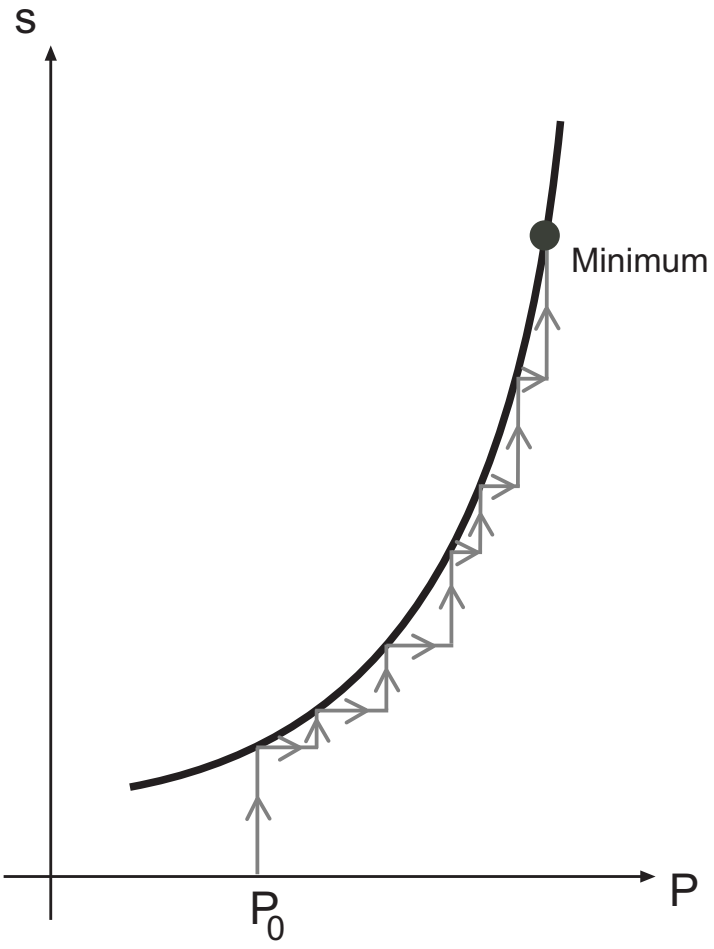


Figure 5.1: PDM: Trajectory of the iterates in the optimization space (2D case)

movement on the tangent line to approximate the movement on the solution curve. In this step, both the surface parameter s and the control points P are modified. Vertical movements and movements along the tangent lines are repeated until the minimizer is found.

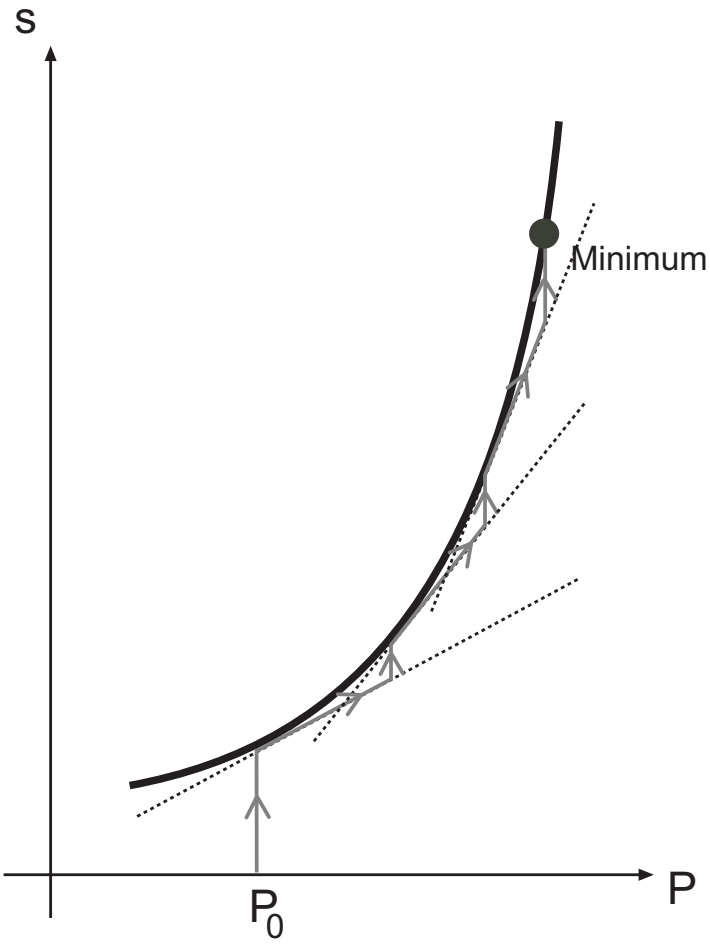


Figure 5.2: TDM: Trajectory of the iterates in the optimization space (2D case)

Similarly, in SDM, there is a step for foot point projection, which is followed by a step for solving a linear least square problem. The trajectory of the iterates for SDM in 2D case is similar to that shown in Figure 5.2. Figure 5.3 gives a 3D illustration. In the step for foot point projection, only the surface parameters s and t are modified and the control points are kept unchanged. In the figure, the yellow plane is the tangent plane formed at the foot point. Then, for the step that solves the linear least squares problem, the movement is made along the tangent plane. SDM uses the movement on

the tangent plane to approximate the movement on the solution surface. The actual movement on the tangent plane is guided by SDM's approximation to the Hessian. During this step, the variables P, s and t are modified. Like TDM, vertical movements and movements on the tangent planes are repeated until the minimizer is found.

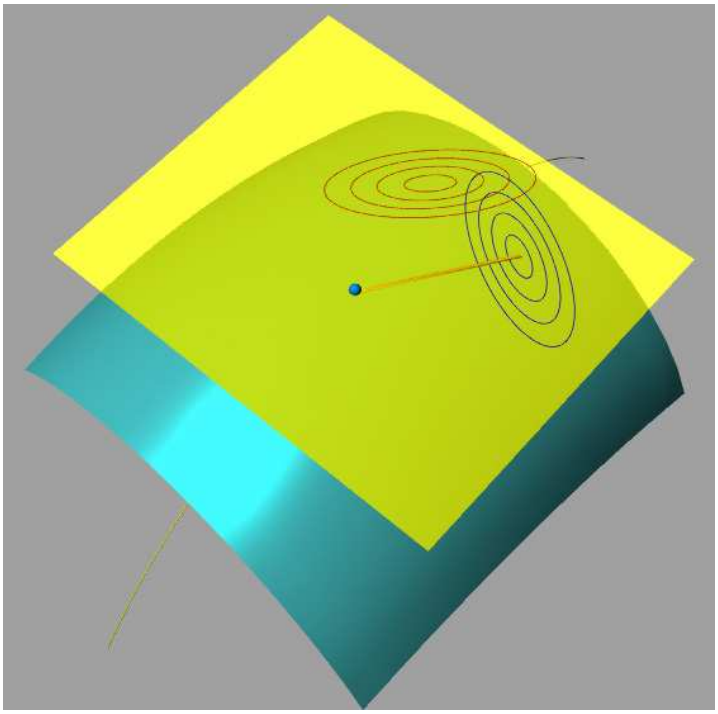


Figure 5.3: SDM: Trajectory of the iterates in the optimization space

In conclusion, although PDM, TDM and SDM share similar practical procedures, the trajectories of the iterates in the optimization space are indeed different in nature. The zigzag path for PDM and the non-zigzag paths for TDM and SDM give some insights about the difference in convergence rates of these methods.

Chapter 6

Experimental Results and Discussions

In this chapter, we present experimental results for comparing the convergence behaviors of different optimization methods. The time and error statistics for the examples are also given. All experiments are conducted on a PC with Intel Xeon 2.8 GHz CPU and 2.00 GB RAM. The machine that we use is a relatively high-end one but should be affordable and reasonable. Our program can handle complex data models such as Buddha (6.64, which consists of 543652 points) using this machine.

6.1 Behaviors of Different Types of Optimization: PDM, TDM, SDM and their Variants

In this section, some experiments are performed to compare the convergence behaviors and stabilities of different optimization methods. Besides, the effectiveness of a trust region strategy and a line search strategy is demonstrated. Although the target shapes used in the experiments in this section are relatively simple, the observations made are expected to be valid for more complex shapes.

6.1.1 Convergence Behaviors of PDM, TDM and SDM

Experiment $A1$ ***(Refer to Figures 6.1, 6.2)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM. A sphere, of which the radius is 0.5, is used as a target data in this experiment. An initial mesh, which contains 50 control points, is obtained by subdividing a $1 \times 1 \times 1$ cube using Loop's scheme. No smoothing term is used. Figure 6.1 shows the target sphere, the initial mesh, the optimized mesh and also the optimized subdivision surface. Fig-

ure 6.2 shows the error curves for PDM, SDM and TDM.

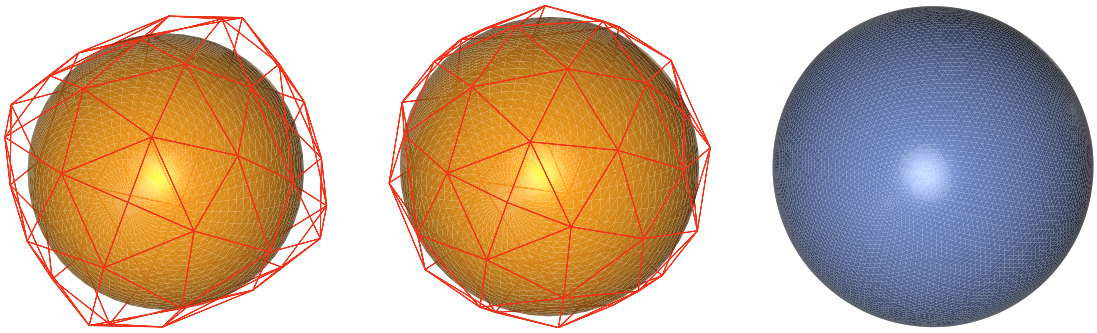


Figure 6.1: Left: Target sphere and initial mesh (red lines); Middle: Optimized mesh obtained by SDM (meshes obtained by PDM and TDM are similar and therefore are not included here); Right: Optimized subdivision surface obtained by SDM

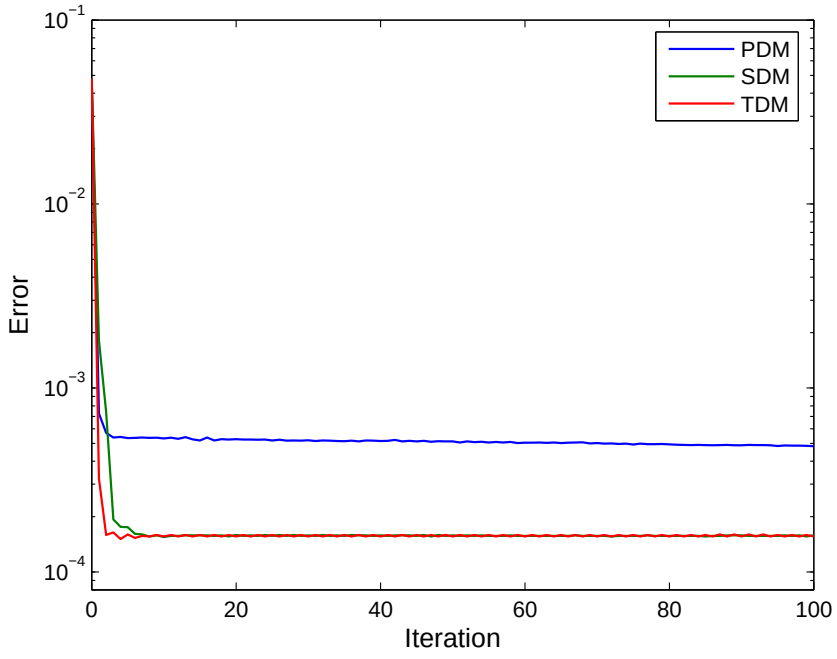


Figure 6.2: Error curves for experiment $A1$

Experiment $A2$ ***(Refer to Figures 6.3, 6.4)***

In this experiment, the target data used is an ellipsoid of which the radii are 0.25, 0.5

and 1.0. The initial mesh is $0.5 \times 1.0 \times 2.0$ and consists of 14 control points. No smoothing term is used. Compared with experiment $A1$, the target shape does not have uniform curvatures. Figure 6.3 shows the target ellipsoid, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.4 shows the error curves for PDM, SDM and TDM.

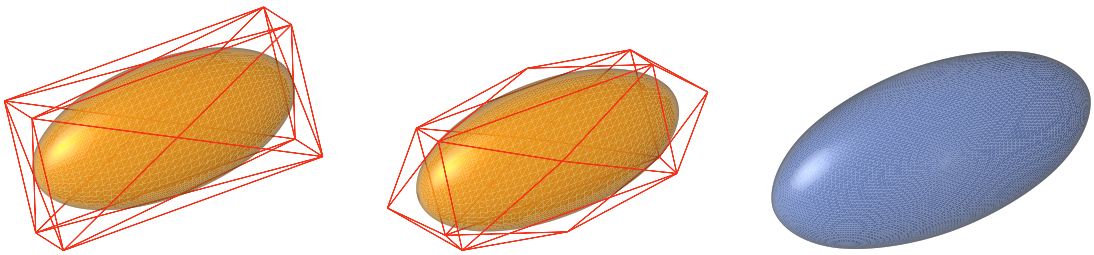


Figure 6.3: Left: Target ellipsoid and initial Mesh (red lines); Middle: Optimized mesh obtained by SDM (meshes obtained by PDM and TDM are similar and therefore are not included here); Right: Optimized subdivision surface obtained by SDM

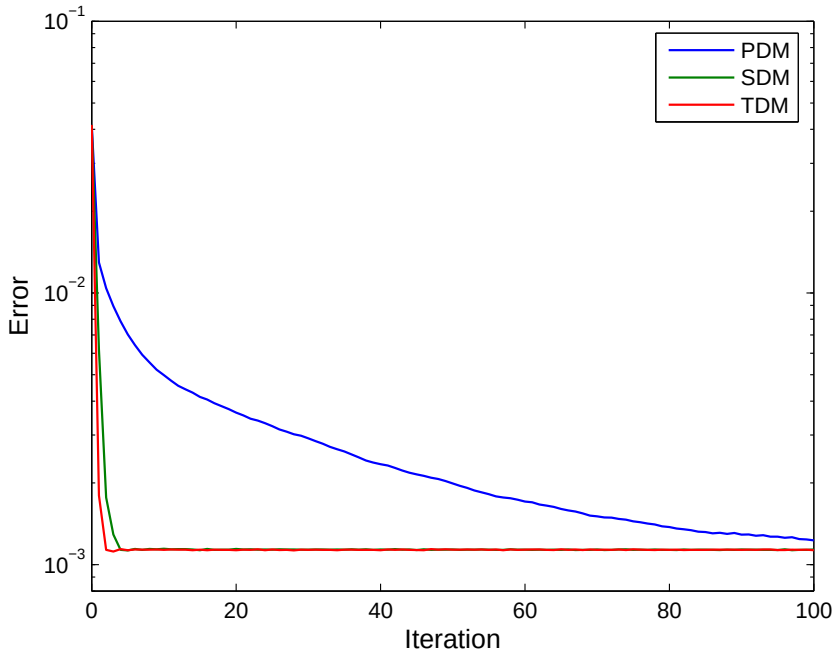


Figure 6.4: Error curves for experiment $A2$

Observation: *TDM and SDM converge faster than PDM with TDM and SDM have similar convergence rates. Furthermore, different local minimums are obtained in these methods.*

From experiments $A1$ and $A2$, it is observed that both TDM and SDM converge much faster than PDM while TDM and SDM have similar performance. In experiment $A1$, PDM takes 67 iterations (4.668 s) to obtain an E_{rms} smaller than 0.0005 while SDM and TDM just take 3 iterations (0.251 s) and 1 iteration (0.06 s) respectively. In experiment $A2$, PDM takes 50 iterations (0.791 s) to obtain an E_{rms} smaller than 0.002 while SDM and TDM just take 2 iterations (0.020 s) and 1 iteration (0.010 s) respectively. These experimental results can be predicted and explained by the theoretical facts mentioned in the previous chapter that PDM is a gradient descent method, TDM is the Gauss-Newton method and SDM is the Newton method.

6.1.2 Convergence Behaviors with Improper Initial Meshes

Experiment $B1$ ***(Refer to Figures 6.5, 6.6, 6.7, 6.8)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM when the initial mesh is not properly aligned with the target shape. A sphere, of which the radius is 0.5, is used as a target data while a $0.8 \times 0.8 \times 0.8$ cube is used as an initial mesh. The initial mesh has 14 control points. Figure 6.5 shows the target shape and the initial mesh. Figure 6.6 shows the meshes after the first iteration for PDM, SDM and TDM. Figure 6.7 shows the optimized meshes for SDM and TDM as well as the meshes for PDM after 100 and 1000 iterations. Figure 6.8 shows the error curves for PDM, SDM and TDM.

Experiment $B2$ ***(Refer to Figures 6.9, 6.10, 6.11, 6.12)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM when the initial mesh is not properly aligned with the target shape and also the control points are not evenly distributed. A sphere, of which the radius is 0.5, is used as a target data while a cone shape mesh, which consists of 34 control points, is used as an initial mesh. Figure 6.9 shows the target shape and the initial mesh. Figure 6.10 shows the meshes after the first iteration for PDM, SDM and TDM. Figure 6.11 shows the optimized meshes for SDM and TDM as well as the meshes for PDM after 100 and 1000 iterations. Figure 6.12 shows the error curves for PDM, SDM and TDM.

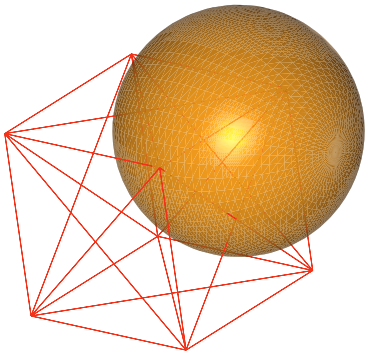


Figure 6.5: Target sphere and initial mesh (red lines)

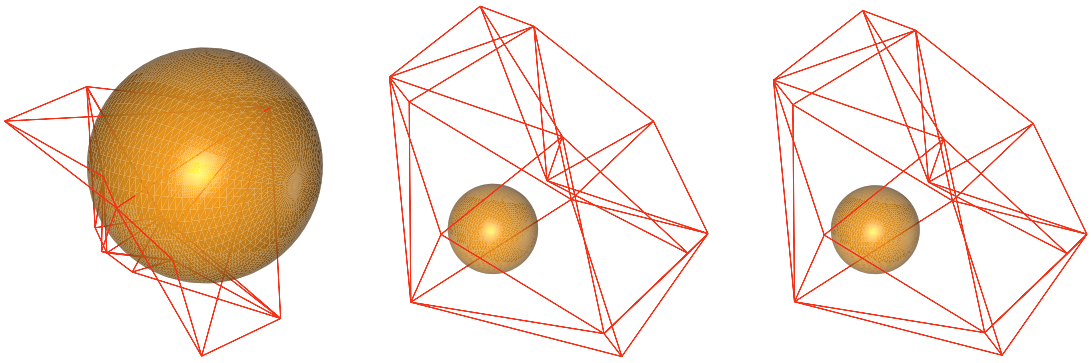


Figure 6.6: Mesh after the 1^{st} iteration. Left: PDM; Middle: SDM; Right: TDM

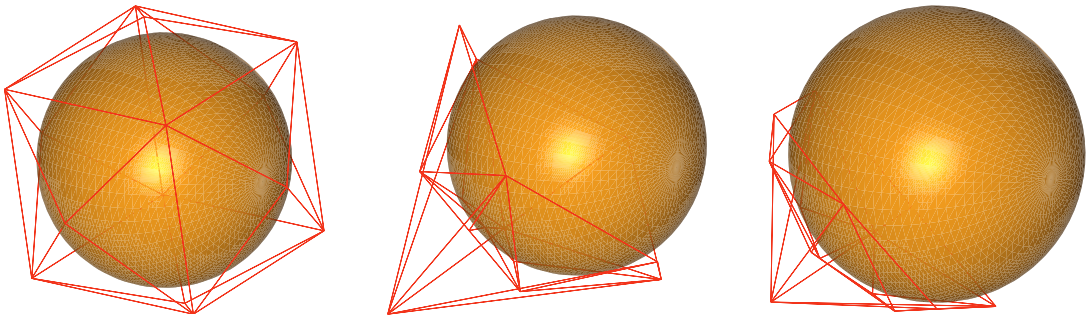


Figure 6.7: Left: Optimized mesh by SDM (the mesh obtained by TDM is similar and is therefore not included here); Middle: Mesh for PDM after 100 iterations; Right: Mesh for PDM after 1000 iterations

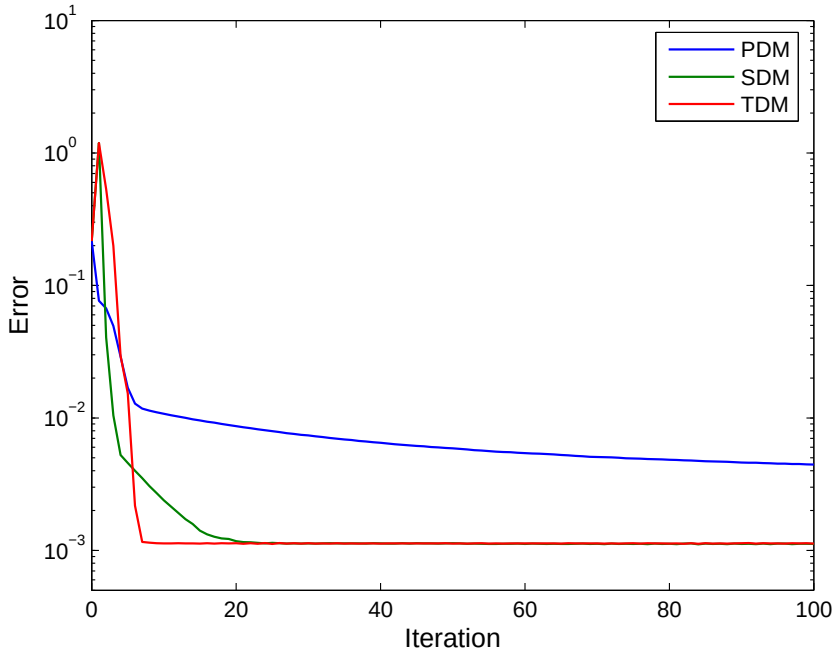


Figure 6.8: Error curves for experiment $B1$

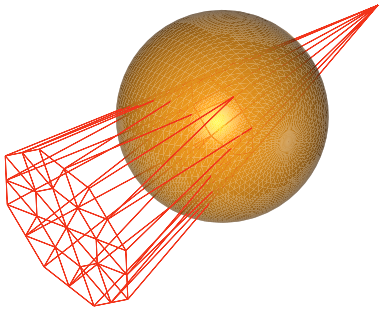


Figure 6.9: Target sphere and initial mesh (red lines)

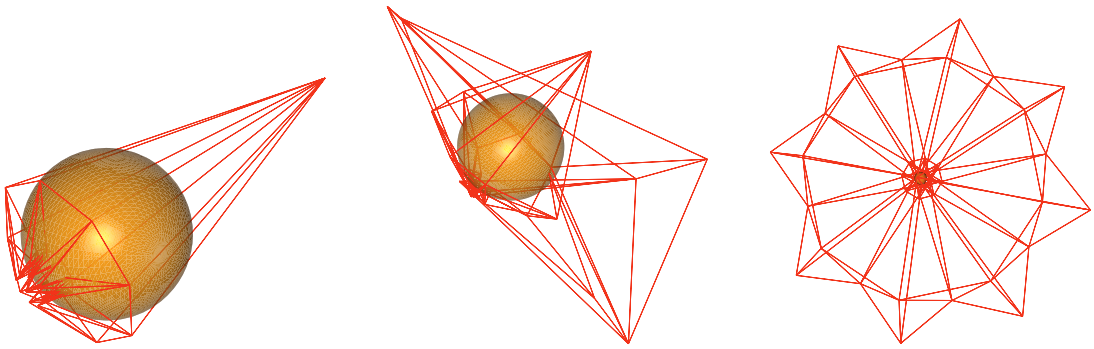


Figure 6.10: Mesh after the 1^{st} iteration. Left: PDM; Middle: SDM; Right: TDM

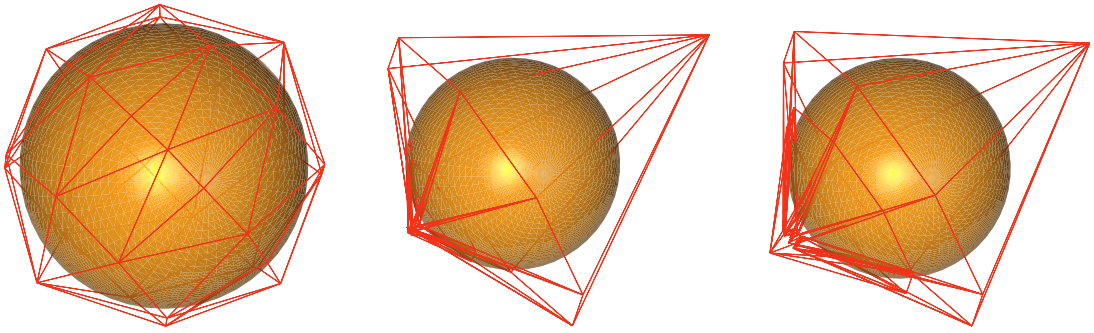


Figure 6.11: Left: Optimized mesh obtained by SDM (the mesh obtained by TDM is similar and is therefore not included here); Middle: Mesh for PDM after 100 iterations; Right: Mesh for PDM after 1000 iterations

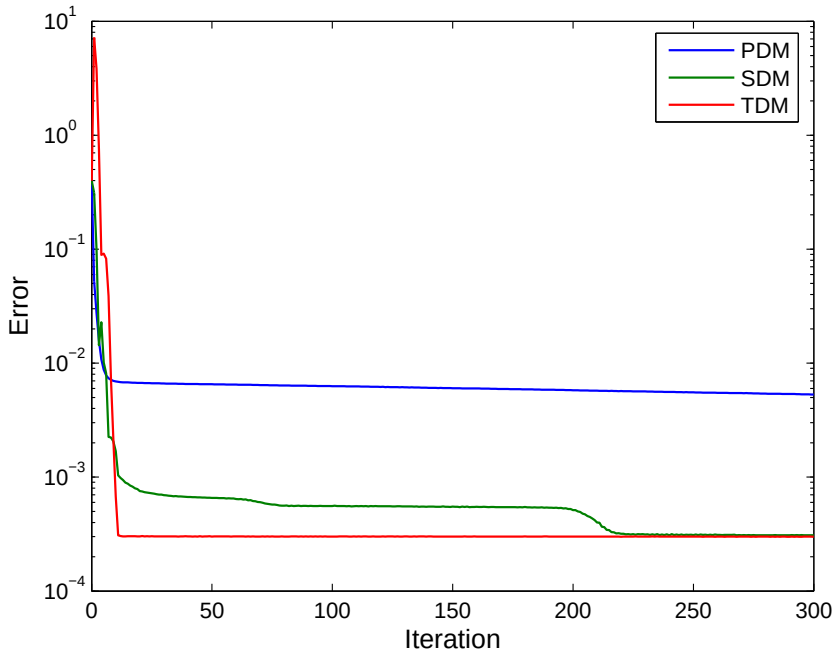


Figure 6.12: Error curves for experiment $B2$

Experiment $B3$ ***(Refer to Figures 6.13, 6.14)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM when the initial mesh is far away from the target shape and also the control points are not evenly distributed. A sphere, of which the radius is 0.5, is used as a target data while an initial mesh $1 \times 1 \times 3$, which consists of 14 control points, is used. Figure 6.13 shows the target sphere, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.14 shows the error curves for PDM, SDM and TDM.

Experiment $B4$ ***(Refer to Figures 6.15, 6.16)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM when the initial mesh is not properly aligned with the target shape and also the control points are not evenly distributed with respect to the target. A disc, of which the radii are 1, 1 and 0.1, is used as a target data while an initial mesh , which consists of 14 control points, is used. The initial mesh is placed orthogonally to the target data. Figure 6.15 shows the target ellipsoid, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.16 shows the error curves for PDM, SDM and TDM.

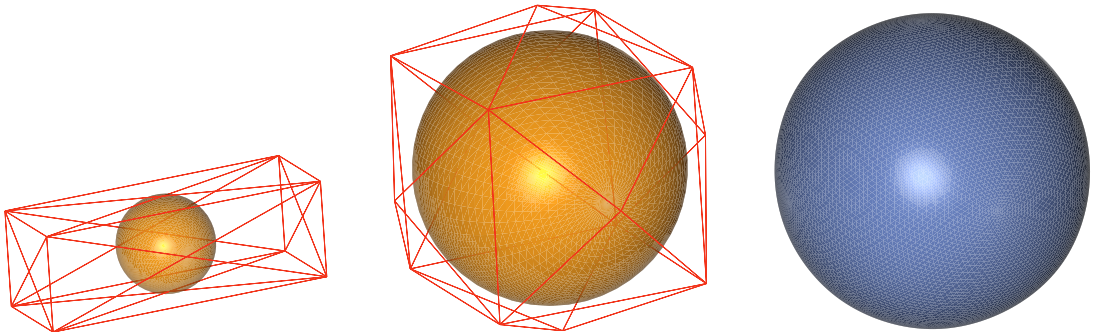


Figure 6.13: Left: Target sphere and initial mesh (red lines); Middle: Optimized mesh obtained by SDM (meshes obtained by PDM and TDM are similar and therefore are not included here); Right: Optimized subdivision surface obtained by SDM

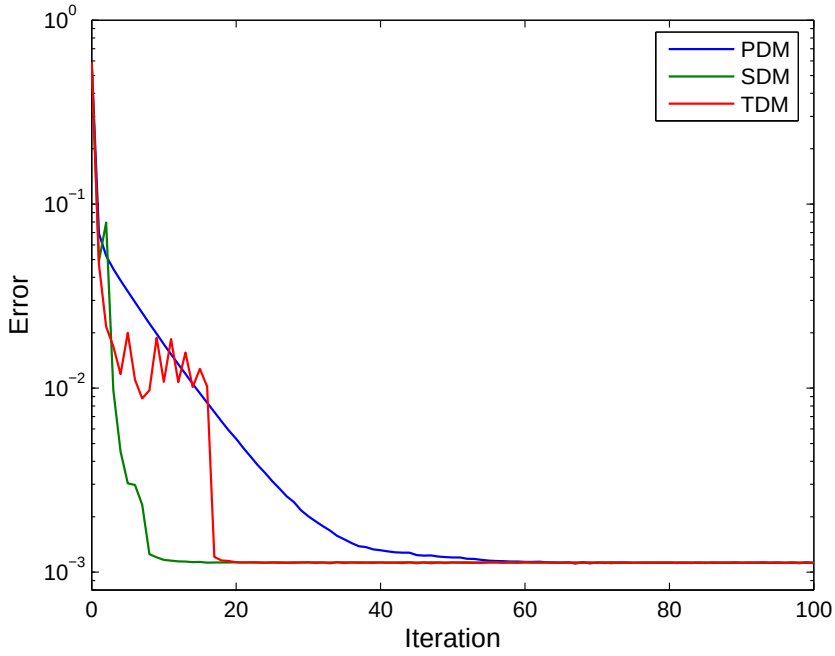


Figure 6.14: Error curves for experiment $B3$

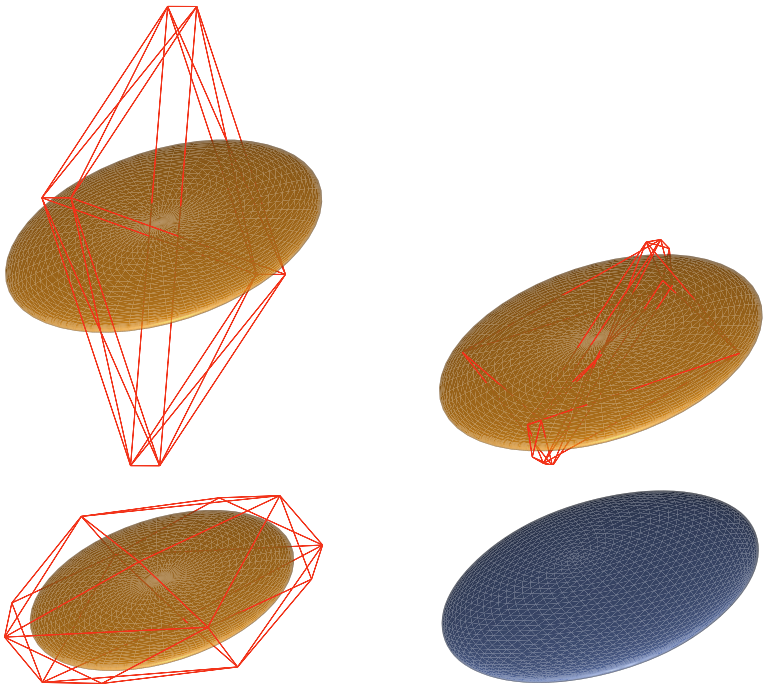


Figure 6.15: Top Left: Target disc and initial mesh (red lines); Top Right: Optimized mesh obtained by PDM; Bottom Left: Optimized mesh obtained by SDM; Bottom Right: Optimized subdivision surface obtained by SDM

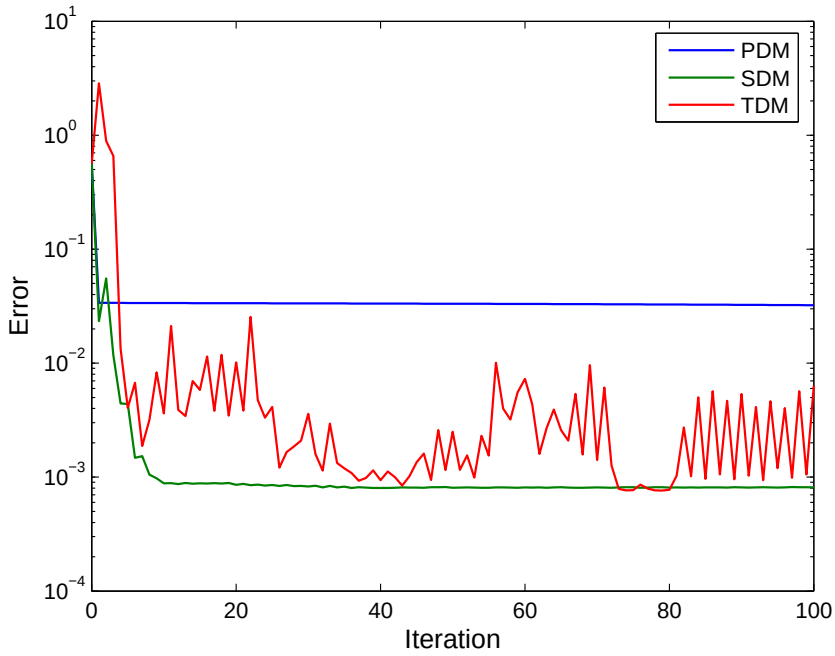


Figure 6.16: Error curves for experiment $B4$

Observation: *SDM converges successfully to the target shapes from improper initial meshes while PDM converges slowly or even fails to converge in some cases. TDM has the stability problem in some cases.*

In experiment $B1$, it can be noticed that SDM and TDM have an obvious advantage over PDM that the control points are able to have large displacements so that the poor initial meshes do not hinder the fitting subdivision surface from converging to the target shape. In experiment $B2$, it is observed that both SDM and TDM are capable of re-distributing the control points efficiently while PDM is sucked by the inappropriate initial distribution of the control points. In experiment $B3$, the initial mesh is intentionally made to be far away from the target along one direction. SDM has the best performance among the three methods while TDM is unstable at the beginning although it still outperforms PDM. PDM takes 36 iterations (0.651 s) to have an E_{rms} smaller than 0.0015 while SDM and TDM 8 iterations (0.130 s) and 17 iterations (0.270 s) to achieve that. In experiment $B4$, PDM cannot move with large enough displacements to overcome the poor initial configuration. TDM is unstable which can be explained by the fact that TDM is the Gauss-Newton method and the discarded term $r(x) \bigtriangledown^2 r(x)$ in the Hessian becomes relatively significant when $r(x)$ is large.

6.1.3 Convergence Behaviors with Far-away Initial Meshes

Experiment $C1$ ***(Refer to Figures 6.17, 6.18)***

This experiment investigates the convergence behaviors of PDM, SDM and TDM when the initial mesh is quite far away from the target shape. An ellipsoid is used as a target data in this experiment. The radii are 0.25, 0.5 and 1.0. An initial mesh of a $4.0 \times 4.0 \times 4.0$ cube, which consists of 14 control points, is used. No smoothing term is used. Figure 6.17 shows the target ellipsoid, the initial mesh and also the optimized subdivision surface. Figure 6.18 shows the error curves for PDM, SDM and TDM.

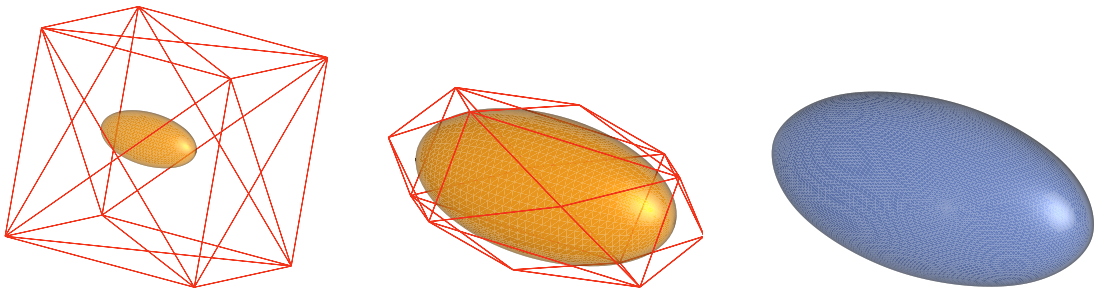


Figure 6.17: Left: Target ellipsoid and initial mesh (red lines); Middle: Optimized mesh obtained by SDM; Right: Optimized subdivision surface obtained by SDM

Observation: *TDM does not have a stable performance when the initial mesh is far away. SDM outperforms PDM.*

From experiment $C1$, it can be observed that SDM converges much faster than PDM while TDM does not work in this example. PDM has an E_{rms} larger than 0.002 after 500 iterations (9.160 s) while SDM just takes 5 iterations (0.071 s) to obtain an E_{rms} smaller than 0.002. Similar to experiment $B4$, TDM does not work well for a far-away initial mesh because of the large initial residue, which is predicted by the theory described in the previous chapter.

6.1.4 Convergence Behaviors for Targets with Large Curvature Regions

Experiment $D1$ ***(Refer to Figures 6.19, 6.20)***

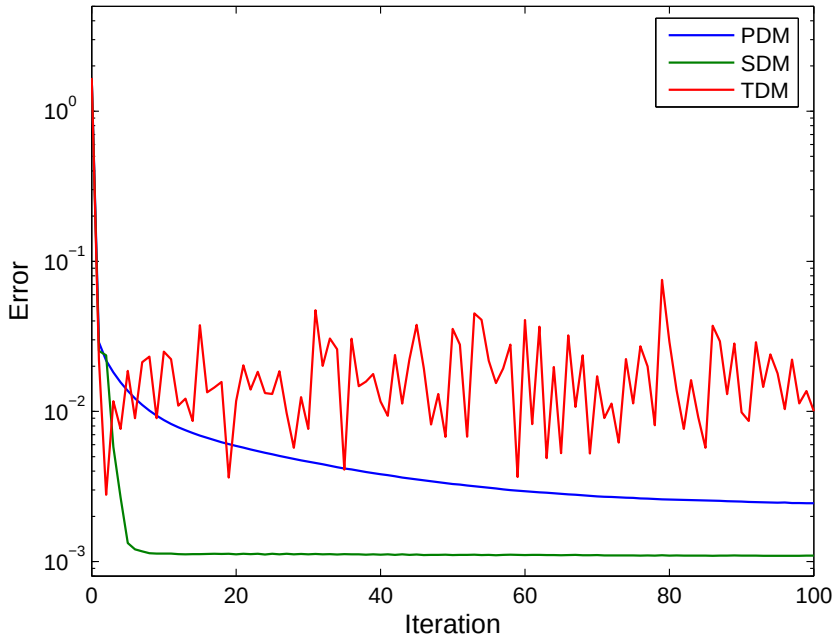


Figure 6.18: Error curves for experiment $C1$

The objective of this experiment is to observe the fitting behaviors of PDM, SDM and TDM when a target shape containing large curvature regions is to be fitted. Here, an ellipsoid having higher curvatures is used as the target data. The radii are 0.25, 0.5 and 4.0. The initial mesh is $0.5 \times 1.0 \times 8.0$. The number of control points in the initial mesh is 14. No smoothing term is used. Figure 6.19 shows the target ellipsoid, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.20 shows the error curves for PDM, SDM and TDM.

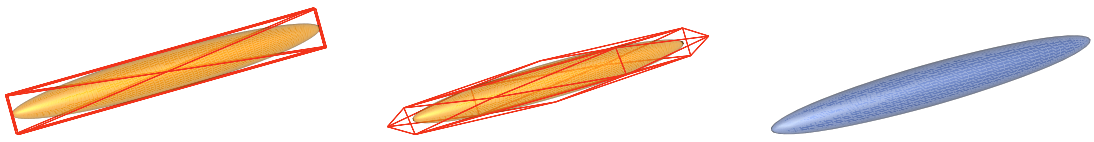


Figure 6.19: Left: Target ellipsoid and initial mesh (red lines); Middle: Optimized mesh obtained by SDM; Right: Optimized subdivision surface obtained by SDM

Experiment $D2$ ***(Refer to Figures 6.21, 6.22)***

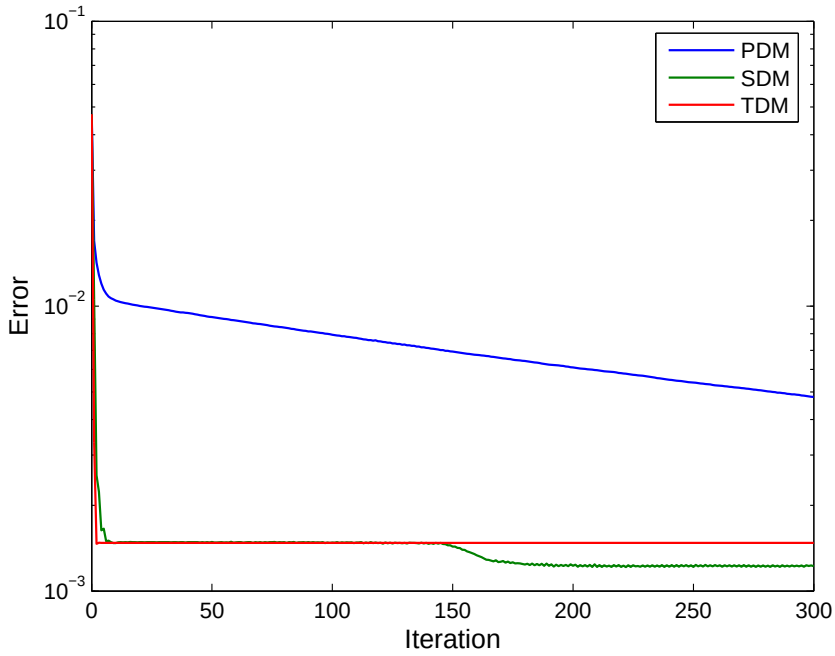


Figure 6.20: Error curves for experiment $D1$

In this experiment, another ellipsoid is used as the target shape. The radii are 0.125, 0.25 and 4.0. The initial mesh is $0.25 \times 0.5 \times 8$. The number of control points in the initial mesh is 14. No smoothing term is used. Figure 6.21 shows the target ellipsoid, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.22 shows the error curves for PDM, SDM and TDM.

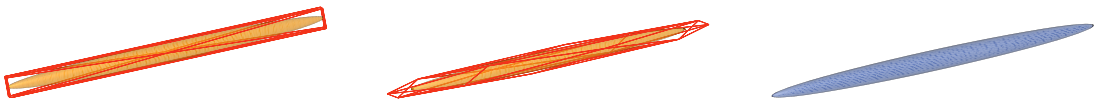


Figure 6.21: Left: Target ellipsoid and initial mesh (red lines); Middle: Optimized mesh by SDM; Right: Optimized subdivision surface by SDM

Experiment $D3$ ***(Refer to Figures 6.23, 6.24)***

In this experiment, a disc-shaped ellipsoid is used as the target shape. The radii are 1.0, 1.0 and 0.1. The initial mesh is $2.0 \times 2.0 \times 0.2$. The number of control points in the initial mesh is 14. No smoothing term is used. Figure 6.23 shows the target

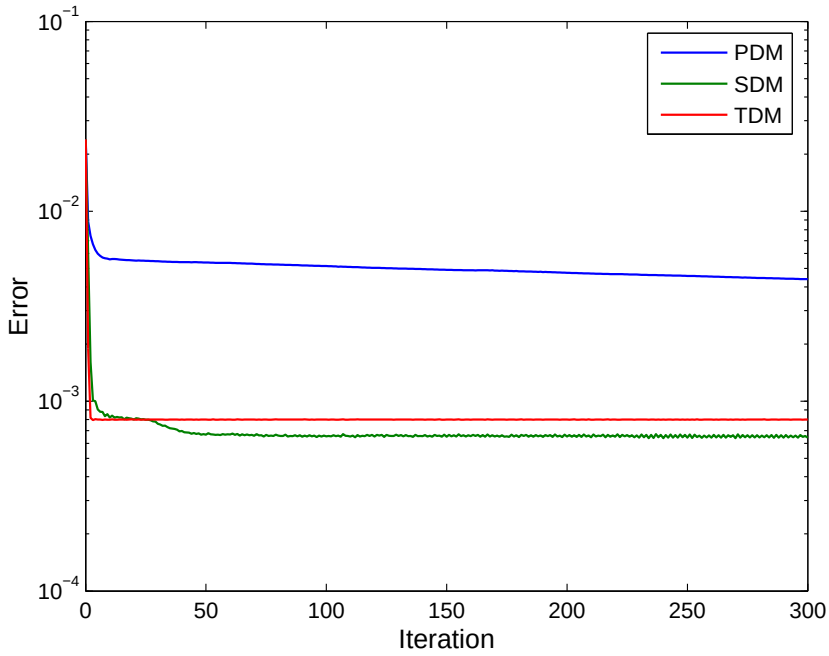


Figure 6.22: Error curves for experiment $D2$

disc, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.24 shows the error curves for PDM, SDM and TDM.

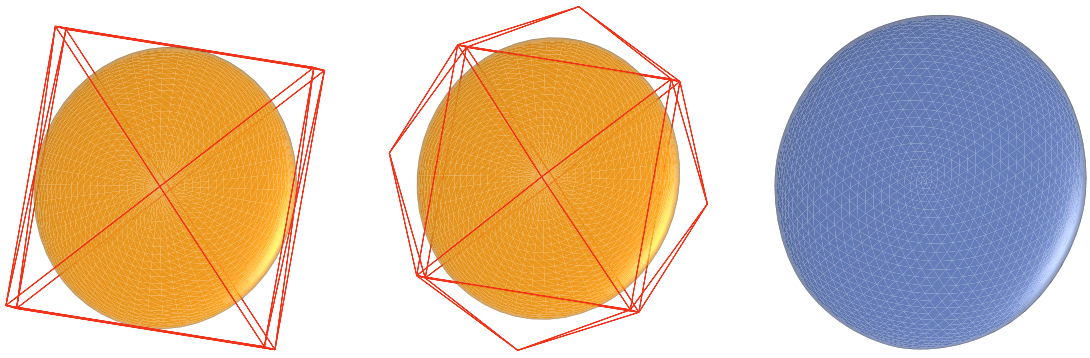


Figure 6.23: Left: Target disc and initial mesh (red lines); Middle: Optimized mesh by SDM; Right: Optimized subdivision surface by SDM

Experiment $D4$ ***(Refer to Figures 6.25, 6.26)***

In this experiment, the target shape, a disc-shaped ellipsoid, is as the same as that used

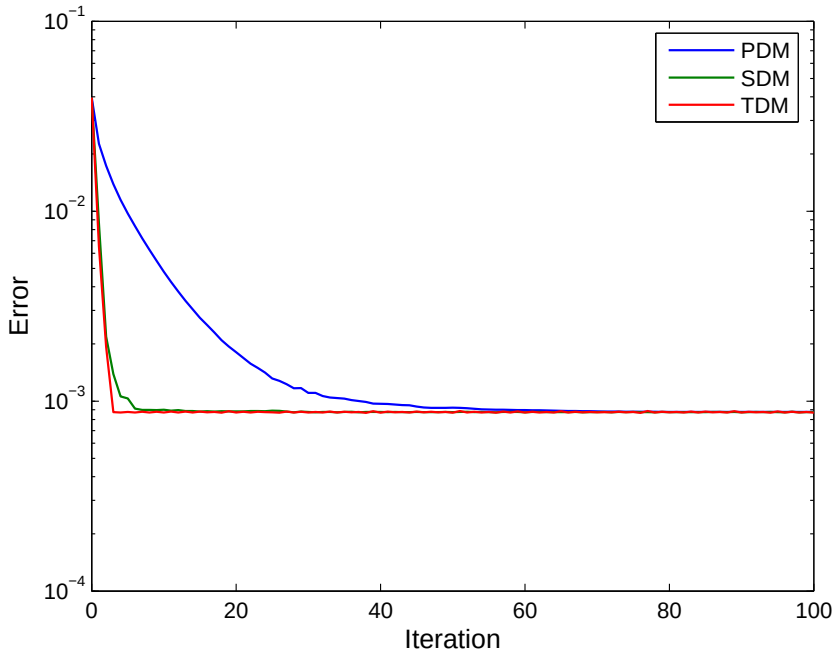


Figure 6.24: Error curves for experiment $D3$

in experiment $D3$. The radii are 1.0, 1.0 and 0.1. The initial mesh is $4.0 \times 4.0 \times 0.4$. The number of control points in the initial mesh is 14. No smoothing term is used. Figure 6.25 shows the target disc, the initial mesh, the optimized mesh and also the optimized subdivision surface. Figure 6.26 shows the error curves for PDM, SDM and TDM.

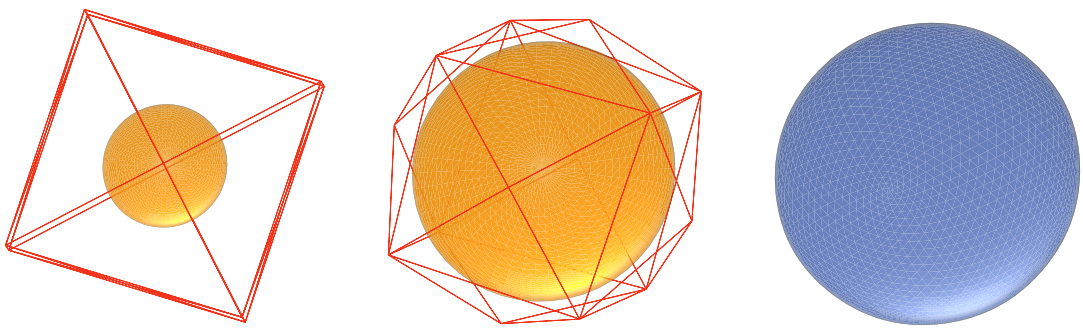


Figure 6.25: Left: Target disc and initial mesh (red lines); Middle: Optimized mesh by SDM; Right: Optimized subdivision surface by SDM

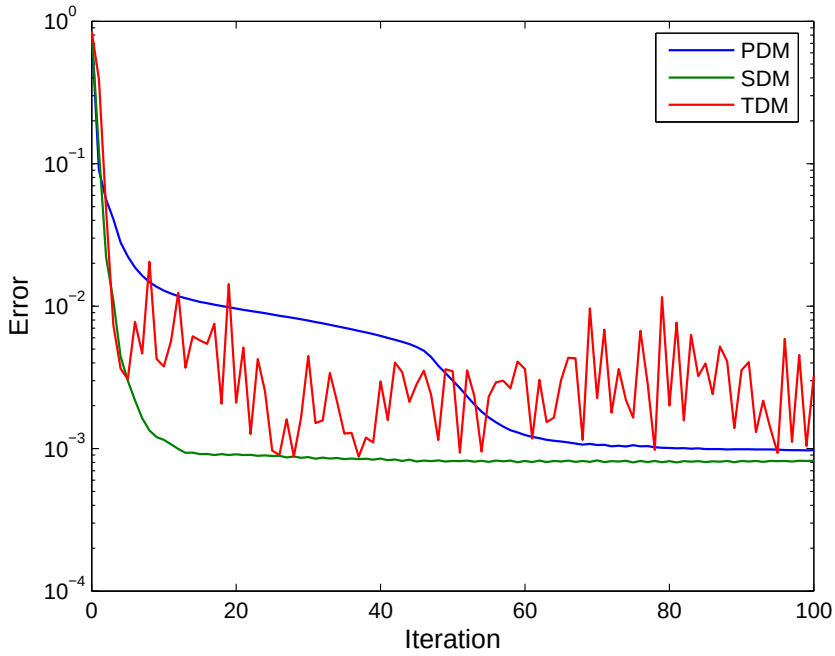


Figure 6.26: Error curves for experiment $D4$

Observation: *PDM has slow convergence. TDM is unstable when the initial mesh is far away. SDM works well in these cases.*

It is observed that both TDM and SDM converge much faster than PDM. In experiment $D1$, PDM takes 282 iterations (7.53 s) to obtain an E_{rms} smaller than 0.005 while SDM and TDM just take 2 iterations (0.05 s) and 1 iteration (0.03 s) respectively. In experiment $D2$, PDM takes 130 iterations (2.113 s) to obtain an E_{rms} smaller than 0.005 while SDM and TDM just take 2 iterations (0.04 s) and 1 iteration (0.02 s) respectively. In both experiments $D1$ and $D2$, the fitting errors obtained by TDM are around 20% larger than those obtained by SDM. This can be explained by the fact that TDM is the Gauss-Newton method and the discarded term $r(x) \bigtriangledown^2 r(x)$ in the Hessian becomes relatively significant when $\bigtriangledown^2 r(x)$ is large. In experiments $D3$ and $D4$, a disc-shaped ellipsoid is used as the target, in which large curvature regions appear long the edge of the disc. In experiment $D3$, the initial mesh is close to the target, TDM can still work well. The discarded term $r(x) \bigtriangledown^2 r(x)$ in the Hessian for the Gauss-Newton method is insignificant when $r(x)$ is small. PDM takes 38 iterations (0.552 to obtain an E_{rms} smaller than 0.001 while SDM and TDM just take 6 iterations (0.091 s) and 3 iterations (0.05 s) respectively. In experiment $D4$, an initial mesh, which is

further away from the target when compared with that used in experiment $D3$, is used. It is observed that SDM converges faster than PDM (PDM takes 83 iterations (1.362 s) to obtain an E_{rms} smaller than 0.001 while SDM takes 12 iterations (0.221 s) to achieve that) while TDM is not stable in this experiment because of the large initial residue. The behaviors in the above experiments can be explained by the fact that TDM is the Gauss-Newton method. As discussed in the previous chapter, the Gauss-Newton method works poorly in large residual problems because of the ignorance of curvature-related second order terms in the Hessian.

6.1.5 Convergence Behaviors for Optimizations with Smoothing Terms

Experiment $E1$ ***(Refer to Figure 6.27)***
In the previous experiments $A1$-$D4$, no smoothing term is included in the goal functions. In this experiment, a smoothing term is added. A sphere of radii 0.5, that has been used in experiment $A1$, is used. The coefficient for the smoothing term is 0.01. The initial mesh used, which has 14 control points, is identical to the one used in experiment $A1$. Figure 6.27 shows the error curves for PDM, SDM and TDM.

Experiment $E2$ ***(Refer to Figure 6.28)***
In this experiment, the same target and the initial mesh are used as those in the previous experiment. The coefficient for the smoothing term is 0.001. Figure 6.28 shows the error curves for PDM, SDM and TDM.

Experiment $E3$ ***(Refer to Figure 6.29)***
In this experiment, the target data used is a ball joint which is shown in Figure 6.59. If no smoothing term is included, self-intersections can occur. Here, a smoothing term is added to avoid self-intersections on the subdivision surface. The coefficient for the smoothing term is 0.01. The number of control points in the initial mesh is 128. Figure 6.29 shows the error curves for PDM, SDM and TDM.

Experiment $E4$ ***(Refer to Figure 6.30)***
In this experiment, the target data and the initial mesh are identical to those in experiment $E3$. But, a smaller coefficient, 0.001, is used for the smoothing term in this experiment. Figure 6.30 shows the error curves for PDM, SDM and TDM.

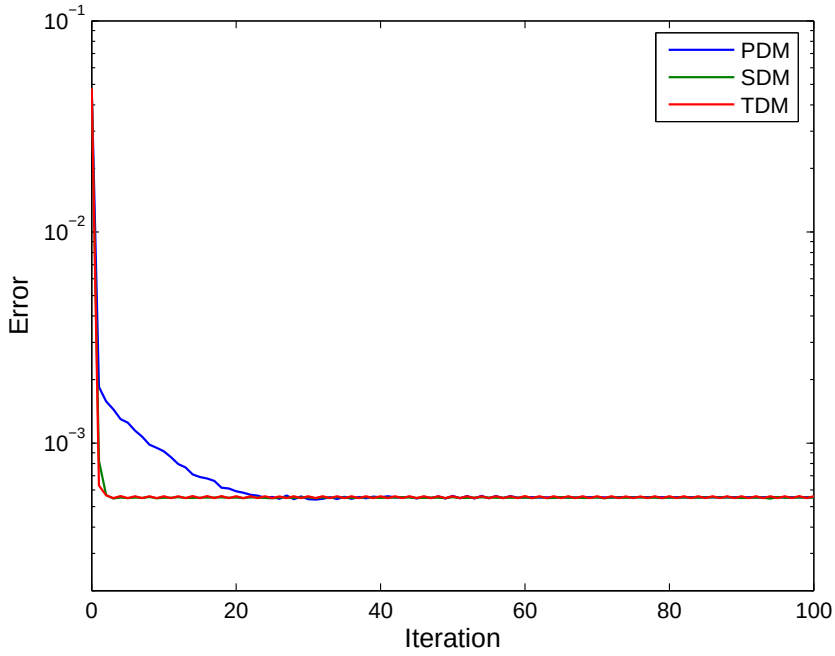


Figure 6.27: Error curves for experiment $E1$

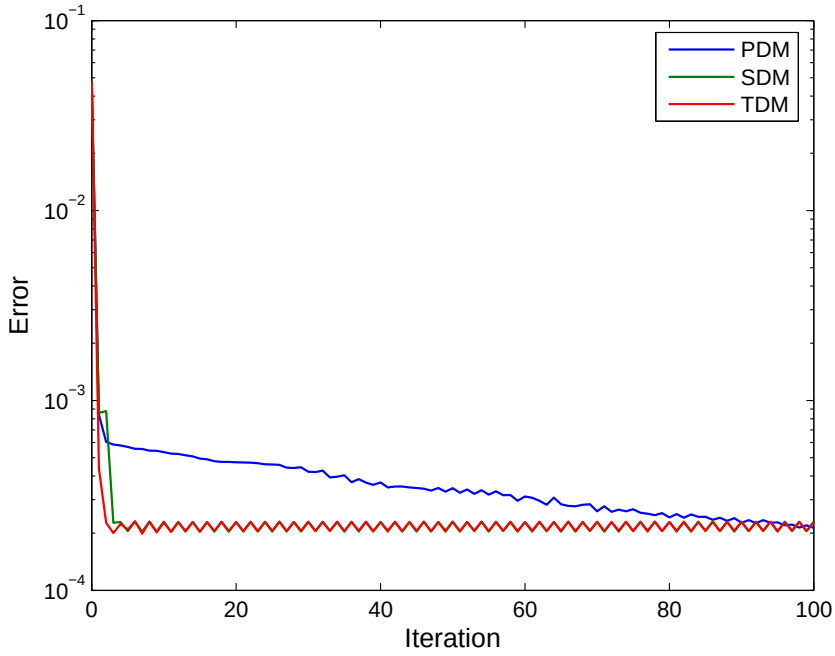


Figure 6.28: Error curves for experiment $E2$

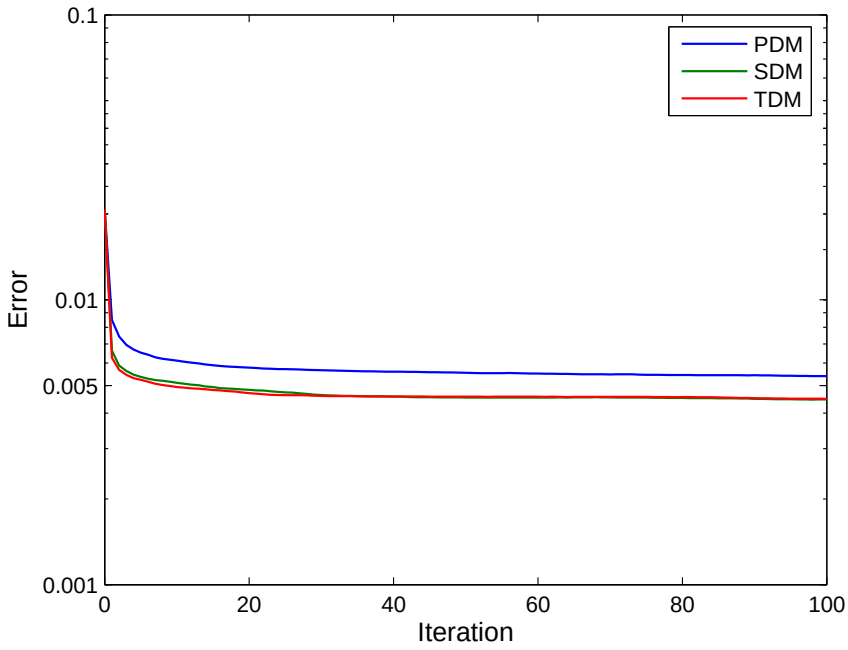


Figure 6.29: Error curves for experiment $E3$

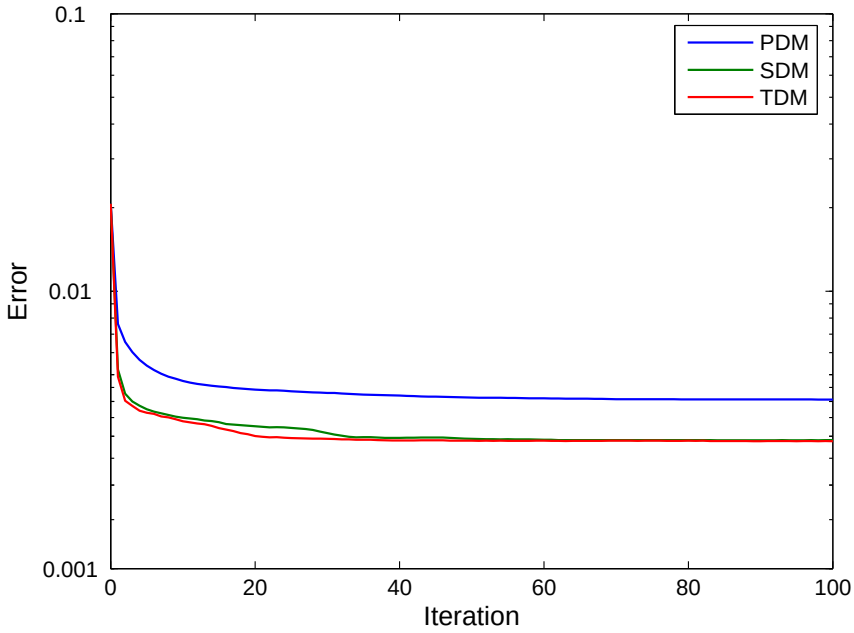


Figure 6.30: Error curves for experiment $E4$

Observation: *Like the experiments with no smoothing terms used, SDM and TDM outperform PDM when smoothing terms are added to the goal functions.*

In experiments $E1$ and $E2$, it is observed that SDM and TDM still converge faster than PDM although the fitting errors obtained are larger than the cases when no smoothing term is added in experiment $A1$. In experiment $E1$, PDM takes 20 iterations (2.013 s) to obtain an E_{rms} smaller than 0.0006 while SDM and TDM just take 2 iterations (0.190 s) and 2 iterations (0.180 s) respectively. In experiment $E2$, PDM takes 15 iterations (1.463 s) to obtain an E_{rms} smaller than 0.0005 while SDM and TDM just take 3 iterations (0.320 s) and 1 iteration (0.090 s) respectively. In experiments $E3$ and $E4$, as in the previous experiments, TDM and SDM converge much faster than PDM. In experiment $E3$, PDM takes 62 iterations (25.065 s) to obtain an E_{rms} smaller than 0.0055 while SDM and TDM just take 4 iterations (1.683 s) and 3 iterations (1.282 s) respectively. In experiment $E4$, PDM takes 17 iterations (6.739 s) to obtain an E_{rms} smaller than 0.0045 while SDM and TDM just take 2 iterations (0.880 s) and 2 iterations (0.881 s) respectively. It is observed that small errors are produced when a smaller coefficient for the smoothing term is used for PDM, TDM and SDM. The reason is that the distance functions play a more important role when a smaller coefficient for the smoothing term is used. These experiments show that the inclusion of the smoothing term will not change the general convergence behavior of PDM, TDM and SDM. The insight from these experiments is important since the smoothing term is often needed for avoiding self-intersections, especially during the early stages of the fitting process.

6.1.6 Convergence Behaviors for Multi-Staged Optimizations

Experiment $F1$ ***(Refer to Figure 6.31)***

In experiments $A1$ - $E4$, the optimizations are recognized as single-staged optimizations. The numbers of control points (in effect the numbers of variables in the optimization problems) as well as the coefficients for the smoothing terms are kept constant throughout the whole process. In this experiment, a multi-staged optimization is demonstrated. The target data used is a ball joint which is shown in Figure 6.59. The coefficient for the smoothing term is 0.01. The initial number of control points in the initial mesh is 128 and new control points are inserted at large error regions throughout the process. The final numbers of control points for PDM, SDM and TDM are

217, 172 and 184 respectively. Figure 6.31 shows the error curves for PDM, SDM and TDM.

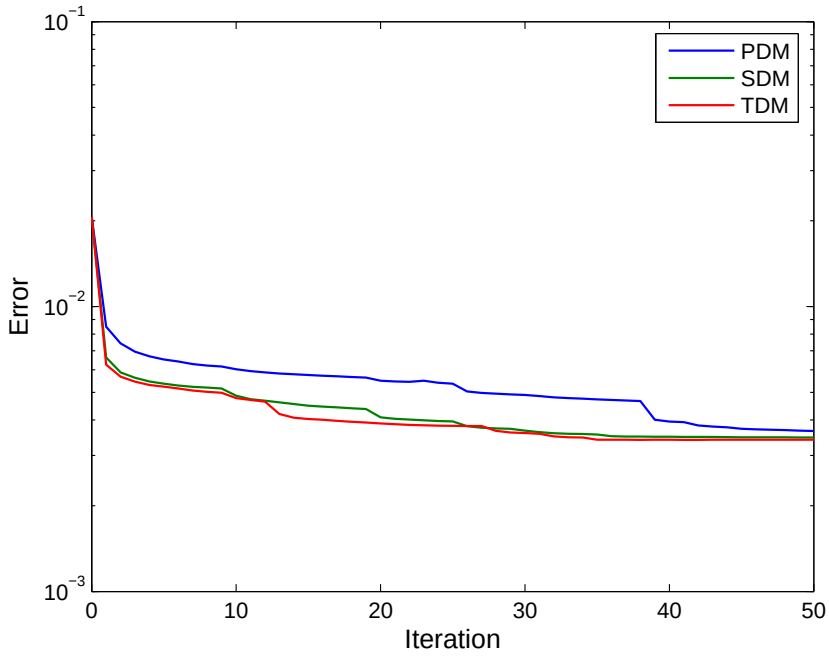


Figure 6.31: Error curves for experiment $F1$

Experiment $F2$ ***(Refer to Figure 6.32)***

In this experiment, the target data used is as the same as the one that used in experiment $F1$. It is a ball joint which is shown in Figure 6.59. The number of control points in the initial mesh is 128. Different from experiment $F1$, the number of control points is fixed throughout this experiment while the coefficient for the smoothing term is decreased gradually. The coefficient for the smoothing term has an initial value of 0.01, and it is set to 0.001 and 0.0001 at the 20^{th} and the 40^{th} iterations respectively. Figure 6.32 shows the error curves for PDM, SDM and TDM.

Experiment $F3$ ***(Refer to Figure 6.33)***

In this experiment, the target data used is as the same as the one that used in experiments $F1$ and $F2$. It is a ball joint which is shown in Figure 6.59. The initial number of control points in the initial mesh is 128 and the initial value of the coefficient for the smoothing term is 0.01. Throughout this experiment, new control points are inserted around the large error regions and the coefficient for the smoothing term is decreased.

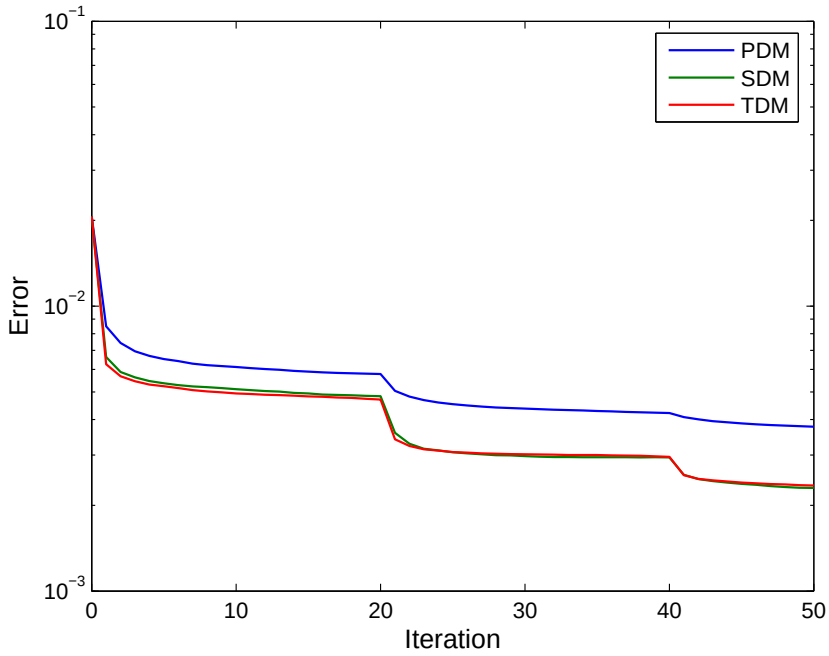


Figure 6.32: Error curves for experiment $F2$

The final numbers of control points for PDM, SDM and TDM are 202, 199 and 151 respectively while the coefficient for the smoothing term is set to 0.001 and 0.0001 at the 20^{th} and the 40^{th} iterations respectively. Figure 6.33 shows the error curves for PDM, SDM and TDM.

Observation: *Compared with those of the single-staged optimizations, smaller fitting errors are obtained using the multi-staged optimizations with the increasing degree of freedom (due to the insertion of control points) or the decreasing coefficient of the smoothing term. Within each stage of the multi-staged optimizations, same convergence behaviors of PDM, TDM and SDM are observed as those in the singled-staged cases.*

From the experiments, it is observed that SDM and TDM converge faster than PDM. SDM and TDM also yield smaller fitting error than PDM. These results are significant and mean that SDM and TDM can work properly in practical settings since the process of fitting subdivision surfaces to point clouds of complex shapes often involves multi-staged optimizations rather than single-staged optimizations.

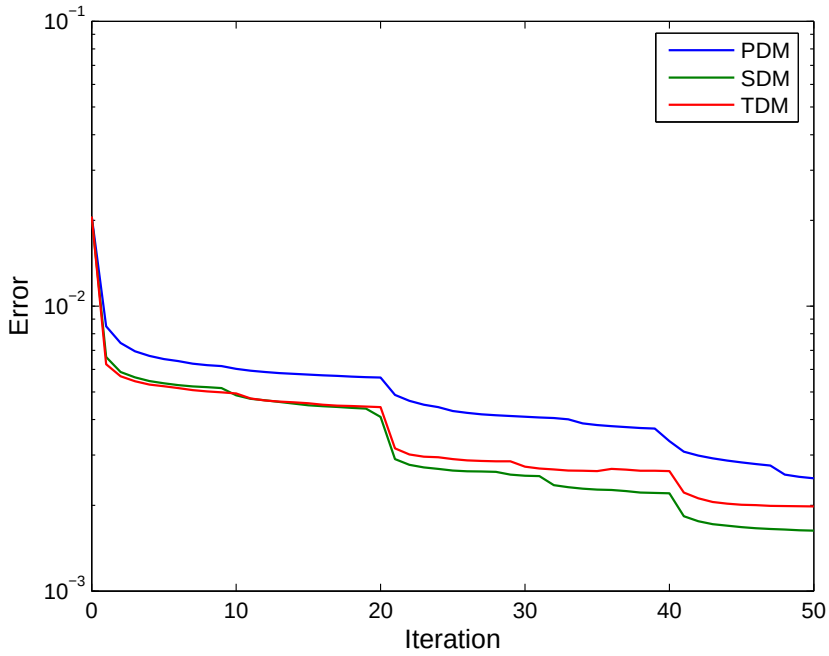


Figure 6.33: Error curves for experiment $F3$

6.1.7 Convergence Behaviors for TDM with Different Smoothing Term Coefficients

Experiment $G1$ ***(Refer to Figure 6.34)***

From the previous experiments, we know that TDM is not stable in some cases. In this experiment, we would like to observe the effect of the smoothing term of different coefficients on TDM. Figure 6.34 shows the error curves for different coefficients for the smoothing term for TDM in experiment $C1$.

Experiment $G2$ ***(Refer to Figure 6.35)***

In this experiment, we would like to observe the effect of the smoothing term of different coefficients on TDM. Figure 6.35 shows the error curves for different coefficients for the smoothing term for TDM in experiment $B4$.

Experiment $G3$ ***(Refer to Figure 6.36)***

In this experiment, we would like to observe the effect of the smoothing term of different coefficients on TDM. Figure 6.36 shows the error curves for different coefficients for the smoothing term for TDM in experiment $D4$.

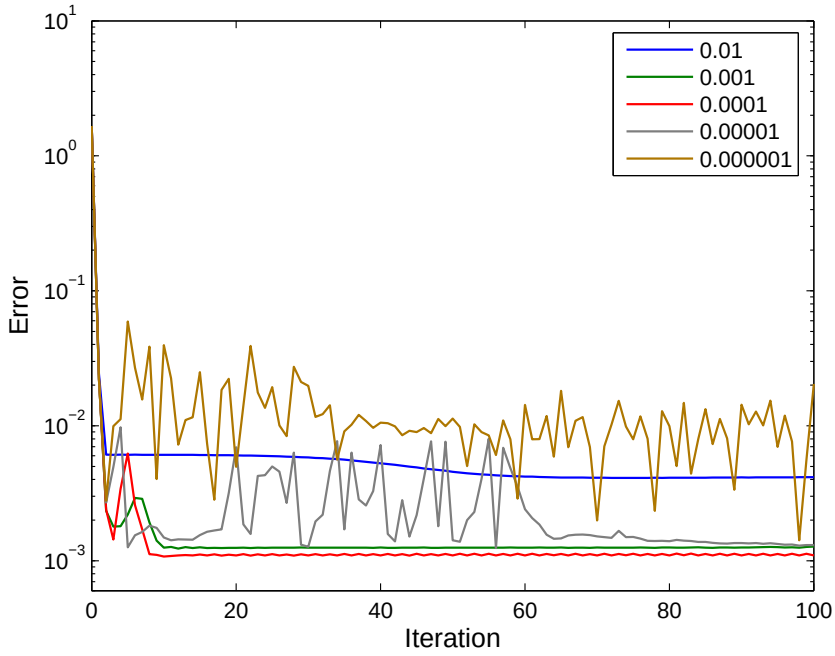


Figure 6.34: Error curves for TDM in experiment $G1$

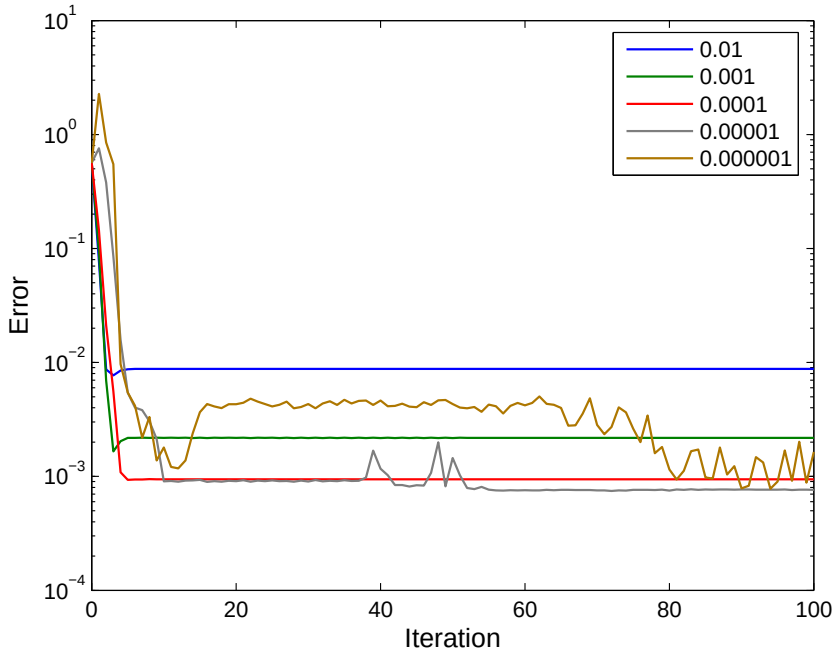


Figure 6.35: Error curves for TDM in experiment $G2$

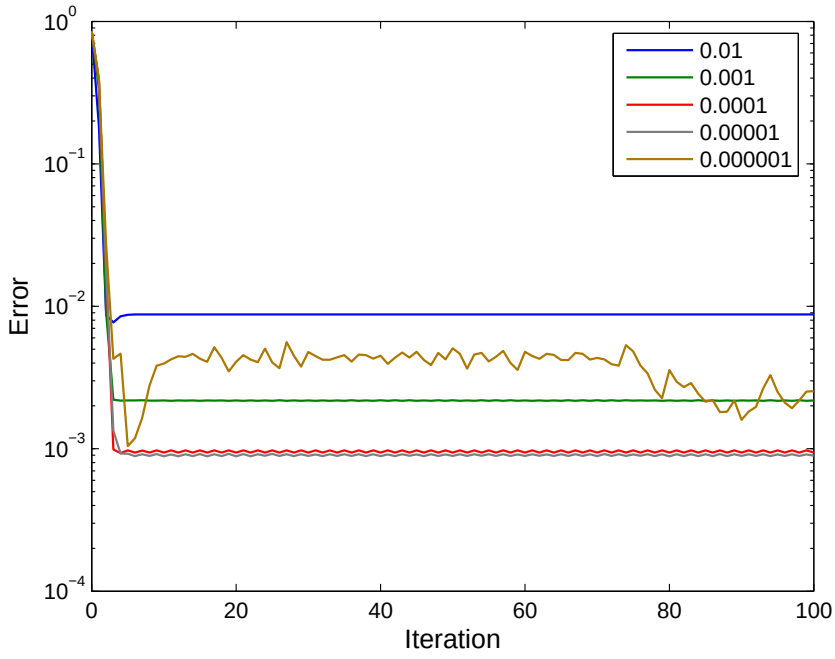


Figure 6.36: Error curves for TDM in experiment $G3$

From Figures 6.34, 6.35 and 6.36, it can be observed that certain values of smoothing term coefficient can stabilize TDM. However, it is not trivial to determine the appropriate value. If the coefficient is too large, the error obtained will be too large. If the coefficient is too small, TDM cannot be stabilized. In the next section, another method for stabilizing TDM is examined.

6.1.8 Convergence Behaviors for TDM with the LM Method

In this section, experiments are carried out to investigate the effect of a trust region method, the LM method, on the convergence behaviors of TDM. Our implementation of the LM method is based on the description in [69]. In this implementation, three parameters, τ, ϵ_1 and ϵ_2, need to be determined. The parameter τ is related to ν_c (a parameter that is described in the section describing the LM method in the previous chapter) by the fact that ν_c having an initial value $\tau \times max\{a_{ii}\}$, where a_{ii} is the diagonal elements in the coefficient matrix. ϵ_1 is related to a stopping criterion, according to which the inner iterations will be terminated when the L_∞ norm for the gradient

vector is smaller than ϵ_1. ϵ_2 is related to another stopping criterion, according to which the change of the solution is small. In particular, $\|h\| \leq \epsilon_2(\|x\| + \epsilon_2)$, where x is the current solution and h is the change of the solution. It is not a trivial task to determine the values of the parameters that can perform well for all cases. Learnt from experience, we set the parameters τ, ϵ_1 and ϵ_2 to 10^{-8}, 10^{-6} and 10^{-8} respectively.
Inside one inner iteration, several tasks need to be done. They include: (i) obtaining a step by solving a linear system using the current value ν_c; (ii) checking the gain ratio; (iii) checking stopping criteria and (iv) updating the parameters according to the gain ratio.

Experiment $H1$ ***(Refer to Figure 6.37)***
In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0) and the initial mesh (has the dimensions $0.5 \times 1.0 \times 2.0$ and consists of 14 control points) are identical to those in experiment $A2$. (See Figure 6.3.) No smoothing term is used. Figure 6.37 shows the error curves for TDM and TDM with the LM method applied. The zoom-in version (along the y-axis) in Figure 6.38 gives a better illustration of the effect of the LM method. Figure 6.44(a) shows the error curve of the inner iterations for the first iteration of the LM method. TDMLM takes 1 iteration (0.551 s) to have an E_{rms} smaller than 0.002.

Experiment $H2$ ***(Refer to Figure 6.39)***
In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0) and the initial mesh (a $4.0 \times 4.0 \times 4.0$ cube, which consists of 14 control points) are identical to those in experiment $C1$. (See Figure 6.17.) No smoothing term is used. Figure 6.39 shows the error curves for TDM and TDM with the LM method applied. Figure 6.44(b) shows the error curve of the inner iterations for the first iteration of the LM method. While TDM does not work well for this setup (as shown in experiment $C1$), TDMLM takes 1 iteration (1.593 s) to have an E_{rms} smaller than 0.002.

Experiment $H3$ ***(Refer to Figure 6.40)***
In this experiment, an ellipsoid containing large curvatures (having the radii 0.25, 0.5 and 4), which has been used in experiment $D1$, is used as the target. The initial mesh used has the dimensions $0.5 \times 1.0 \times 8.0$ and consists of 14 control points. (See Figure 6.19.) No smoothing term is used. Figure 6.40 shows the error curves for TDM and TDM with the LM method applied. Figure 6.44(c) shows the error curve of the inner iterations for the first iteration of the LM method. TDMLM takes 1 iteration

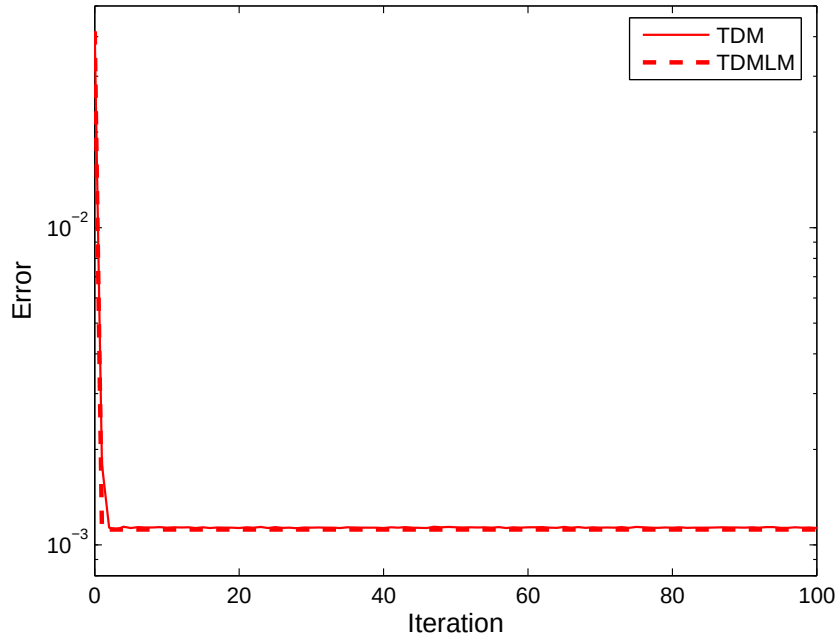


Figure 6.37: Error curves for experiment $H1$

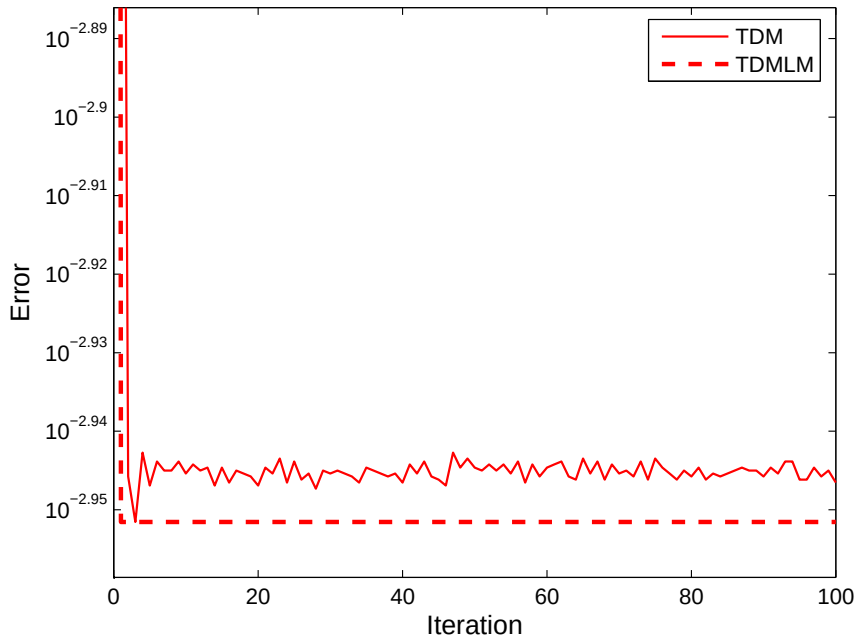


Figure 6.38: Error curves for experiment $H1$ (the zoom-in (along the y-axis) version)

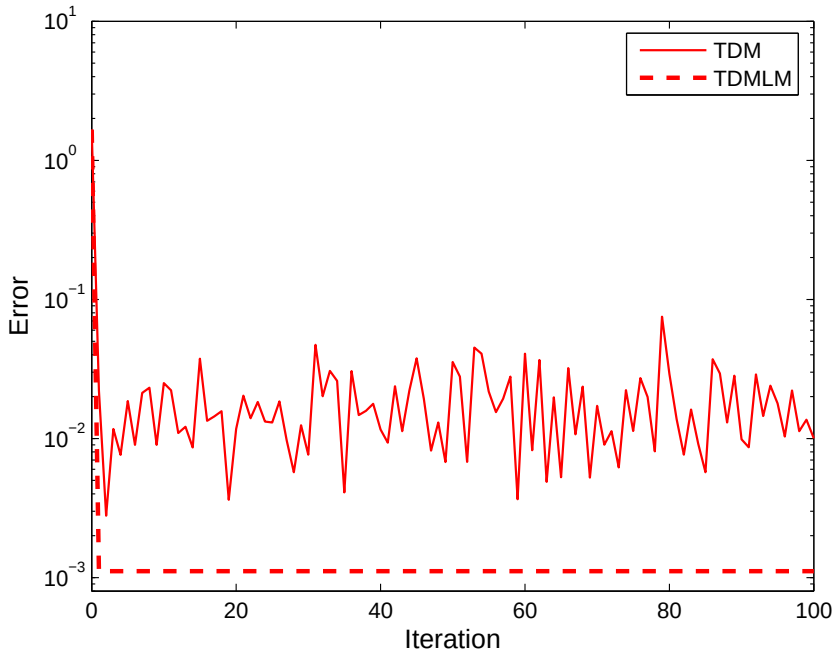


Figure 6.39: Error curves for experiment $H2$

(1.221 s) to have an E_{rms} smaller than 0.005.

Experiment $H4$ ***(Refer to Figure 6.41)***

In this experiment, another ellipsoid containing large curvatures (having the radii 0.125, 0.25 and 4), which has been used in experiment $D2$, is used as the target. The initial mesh used has the dimensions $0.25 \times 0.5 \times 8$ and consists of 14 control points. (See Figure 6.21.) No smoothing term is used. Figure 6.41 shows the error curves for TDM and TDM with the LM method applied. Figure 6.44(d) shows the error curve of the inner iterations for the first iteration of the LM method. TDMLM takes 1 iteration (0.511 s) to have an E_{rms} smaller than 0.005.

Experiment $H5$ ***(Refer to Figure 6.42)***

In this experiment, a disc-shaped ellipsoid (having the radii 1.0, 1.0 and 0.1), which has been used in experiment $B4$, is used as the target. The initial mesh used has the dimensions $2.0 \times 2.0 \times 0.2$ and consists of 14 control points. The initial mesh is put in an orientation that is orthogonal to the target disc. (See Figure 6.15.) No smoothing term is used. Figure 6.42 shows the error curves for TDM and TDM with the LM method applied. Figure 6.44(e) shows the error curve of the inner iterations for the

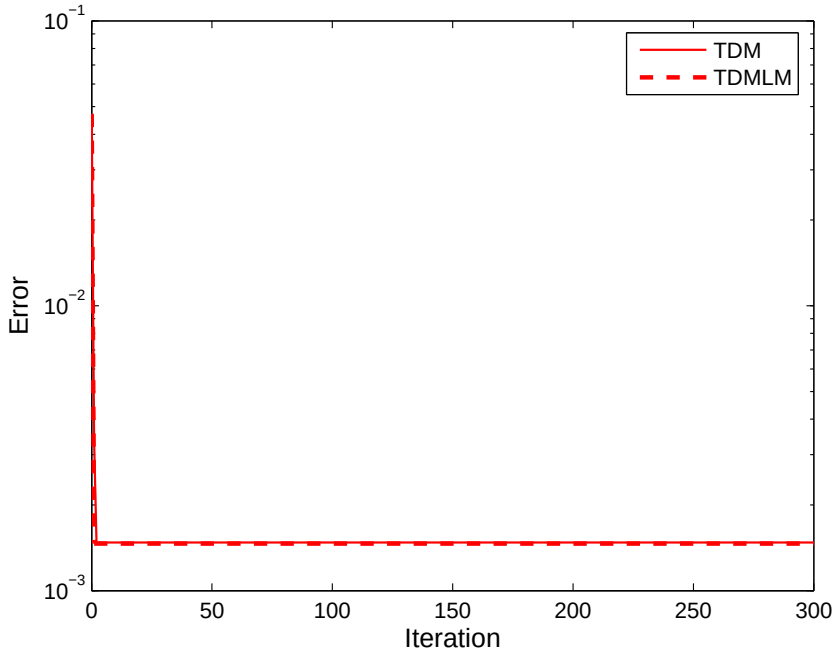


Figure 6.40: Error curves for experiment $H3$

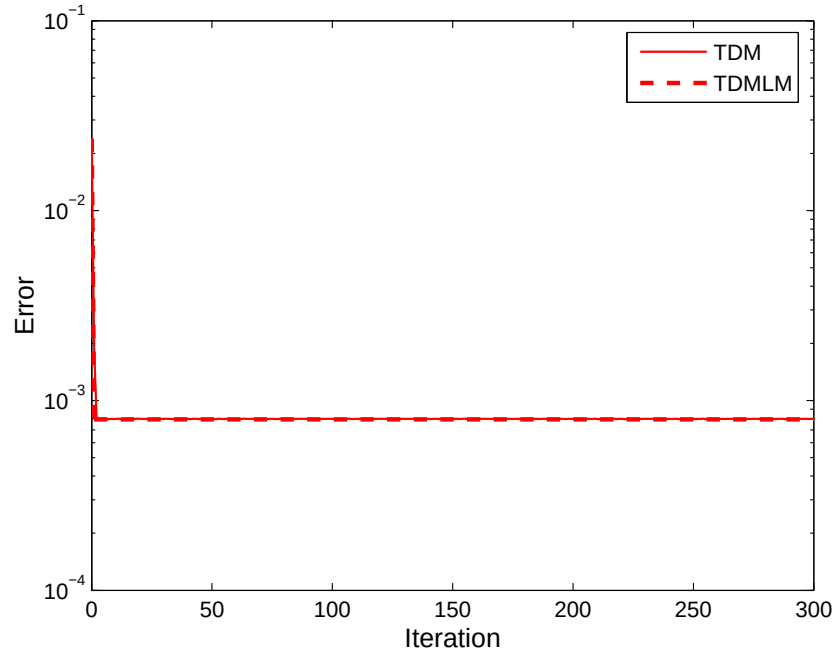


Figure 6.41: Error curves for experiment $H4$

first iteration of the LM method. TDMLM takes 1 iteration (1.662 s) to have an E_{rms} smaller than 0.001.

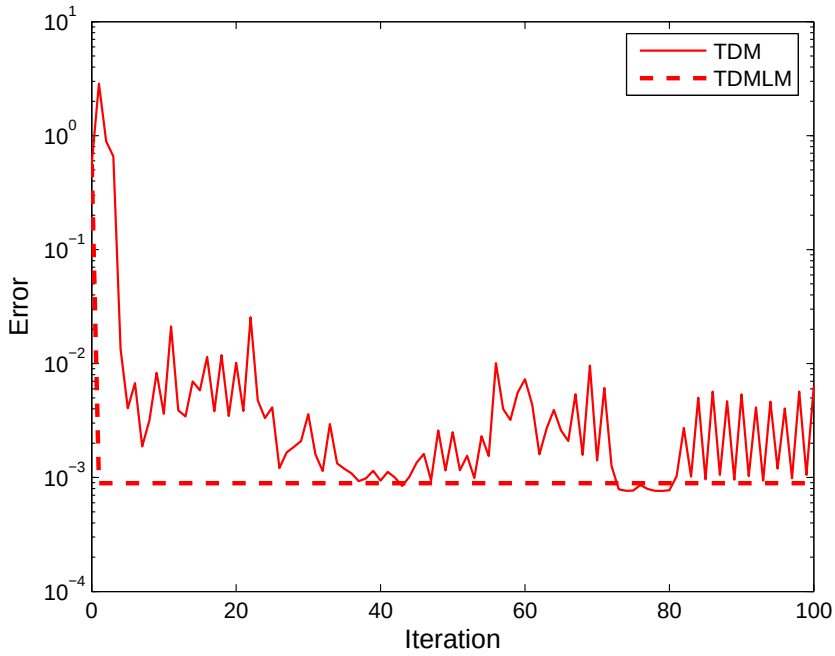


Figure 6.42: Error curves for experiment $H5$

Experiment $H6$ ***(Refer to Figure 6.43)***
In this experiment, a disc-shaped ellipsoid (having the radii 1.0, 1.0 and 0.1), which has been used in experiment $D4$, is used as the target. The initial mesh used has the dimensions $4.0 \times 4.0 \times 0.4$ and consists of 14 control points. (See Figure 6.25.) No smoothing term is used. Figure 6.43 shows the error curves for TDM and TDM with the LM method applied. Figure 6.44(f) shows the error curve of the inner iterations for the first iteration of the LM method. TDMLM takes 1 iteration (0.962 s) to have an E_{rms} smaller than 0.001.

Observation: *In the occasions which TDM works poorly, the LM method can reduce the fitting error and improve the stability.*

In experiment $H1$, it can be observed that the LM method improves the stability of TDM. In experiments $H2$, $H5$ and $H6$, the fitting errors fluctuate in a large range and fail to converge when the pure TDM is applied. But, when the LM method is applied,

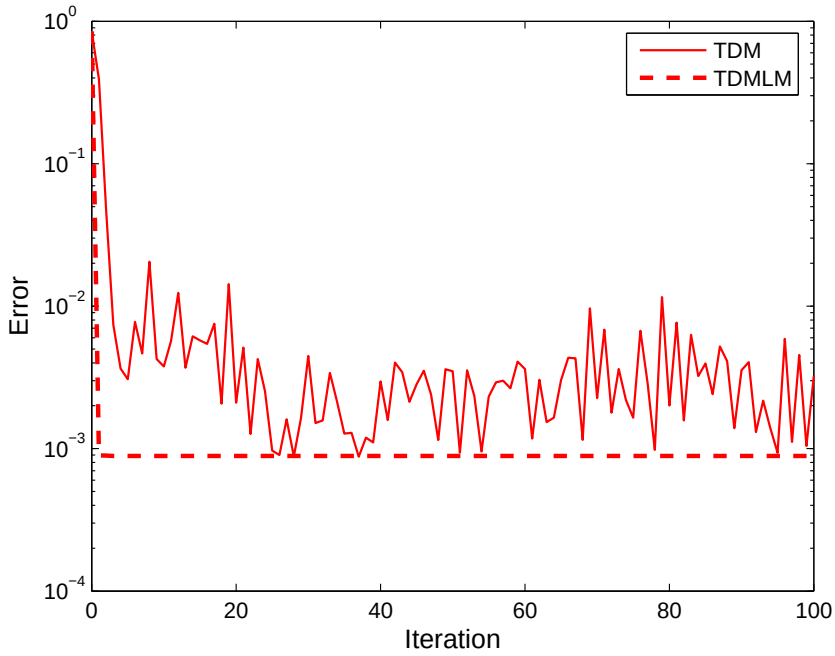


Figure 6.43: Error curves for experiment $H6$

a smaller and stable fitting error is obtained. In other words, with the use of the LM method, TDM can tackle the cases which cannot be handled well using the pure TDM. The reason is that the LM method avoids using the poor approximation (in TDM, the Gauss-Newton method) of the Hessian. More details about the LM method can be found in the previous chapter. In experiments $H3$ and $H4$, like the pure TDM, the TDM with the LM method also works well.

Concerning the computational efficiency, longer time is inevitably needed when the LM method is applied. But, the amount of time required by the LM method (with TDM) is usually less than that required by PDM method, with the former method giving smaller fitting errors.

6.1.9 Convergence Behaviors for SDM with the LM Method

In this section, experiments are carried out to investigate the impact of a trust region method on the convergence behaviors of SDM. The trust region method used is similar to the LM method that is applied to the Gauss-Newton method in the previous section.

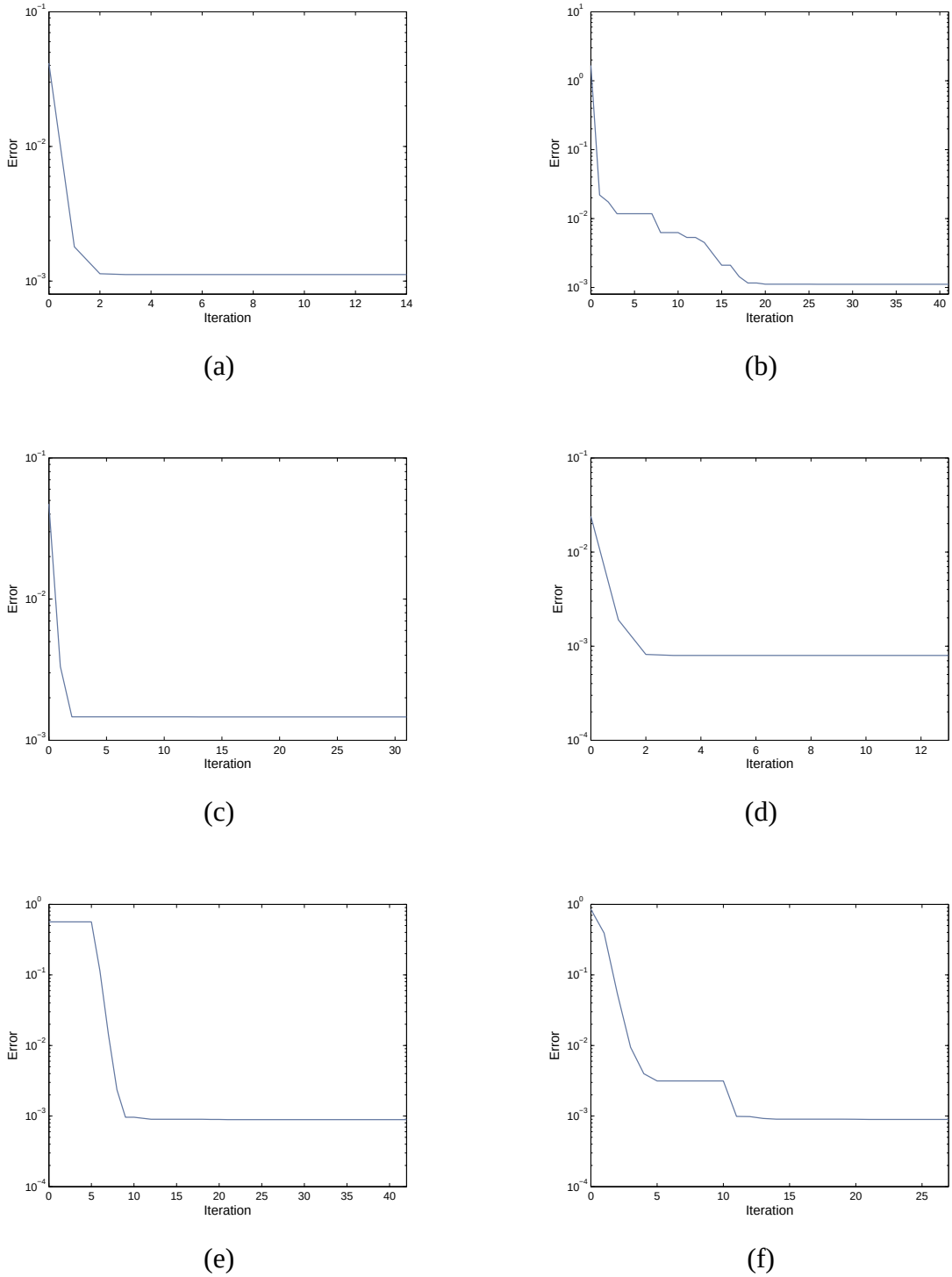


Figure 6.44: Error curves for the inner iterations of the first iteration in the LM method. (a): Experiment $H1$ (14 inner iterations); (b): Experiment $H2$ (41 inner iterations); (c): Experiment $H3$ (31 inner iterations); (d): Experiment $H4$ (13 inner iterations); (e) Experiment $H5$ (42 inner iterations); (f) Experiment $H6$ (27 inner iterations)

Specifically, a term $\nu_c I$ is added to the approximation of the Hessian. From now on, we refer this method as SDM with the use of the LM method although the classical LM method is originally designed to work with the Gauss-Newton method. The parameters for the LM method are set to the same values that have been used in the previous section.

Experiment $I1$ ***(Refer to Figure 6.45)***
In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0) and the initial mesh (has the dimensions $0.5 \times 1.0 \times 2.0$ and consists of 14 control points) are identical to those in experiment $A2$. (See Figure 6.3.) No smoothing term is used. Figure 6.45 shows the error curves for SDM and SDM with the LM method applied. The zoom-in version (along the y-axis) in Figure 6.46 gives a better illustration of the effect of the LM method. Figure 6.52(a) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (1.011 s) to have an E_{rms} smaller than 0.002.

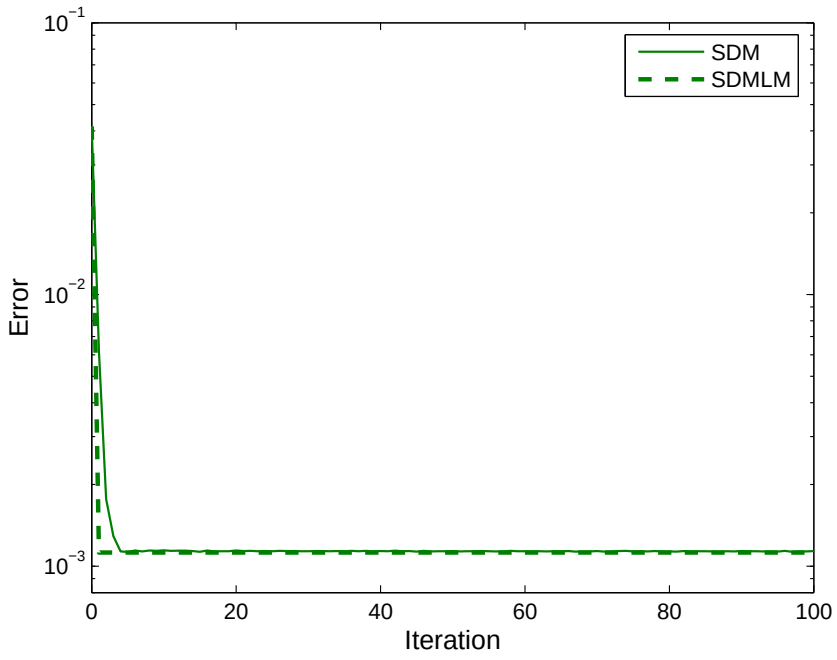


Figure 6.45: Error curves for experiment $I1$

Experiment $I2$ ***(Refer to Figure 6.47)***
In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0)

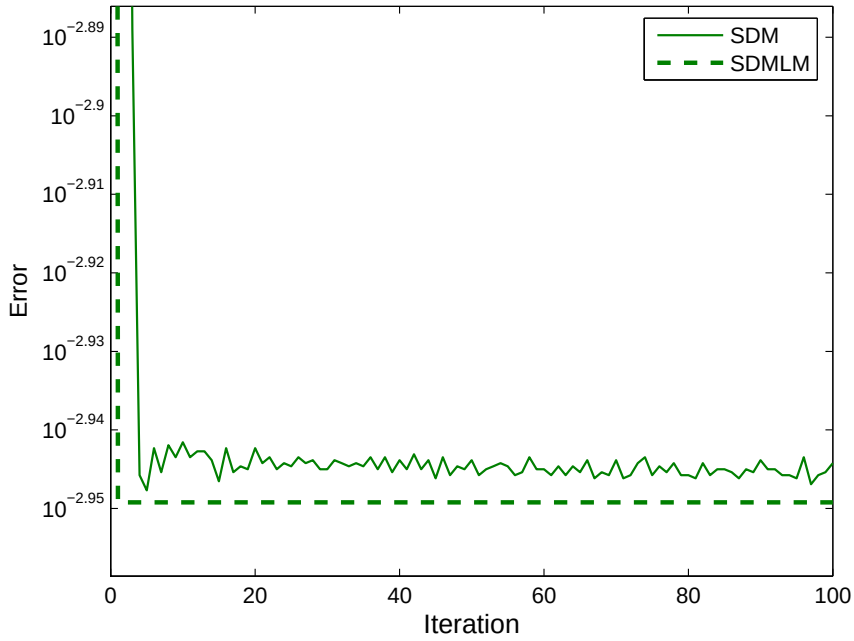


Figure 6.46: Error curves for experiment $I1$ (the zoom-in (along the y-axis) version)

and the initial mesh (a $4.0 \times 4.0 \times 4.0$ cube, which consists of 14 control points) are identical to those in experiment $C1$. (See Figure 6.17.) No smoothing term is used. Figure 6.47 shows the error curves for SDM and SDM with the LM method applied. Figure 6.52(b) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (1.042 s) to have an E_{rms} smaller than 0.002.

Experiment $I3$ ***(Refer to Figure 6.48)***

In this experiment, an ellipsoid containing large curvatures (having the radii 0.25, 0.5 and 4), which has been used in experiment $D1$, is used as the target. The initial mesh used has the dimensions $0.5 \times 1.0 \times 8.0$ and consists of 14 control points. (See Figure 6.19.) No smoothing term is used. Figure 6.48 shows the error curves for SDM and SDM with the LM method applied. Figure 6.52(c) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (1.321 s) to have an E_{rms} smaller than 0.005.

Experiment $I4$ ***(Refer to Figure 6.49)***

In this experiment, another ellipsoid containing large curvatures (having the radii 0.125, 0.25 and 4), which has been used in experiment $D2$, is used as the target. The

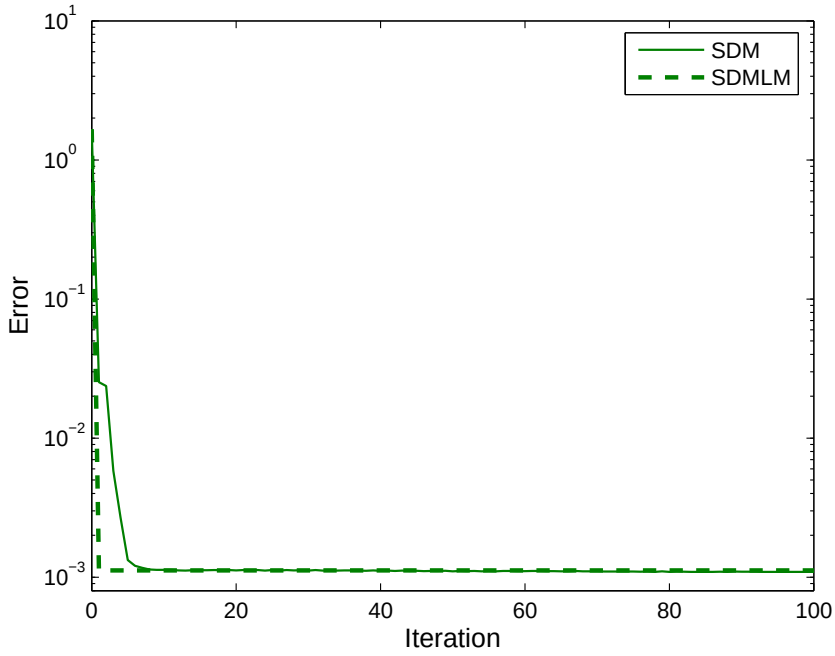


Figure 6.47: Error curves for experiment $I2$

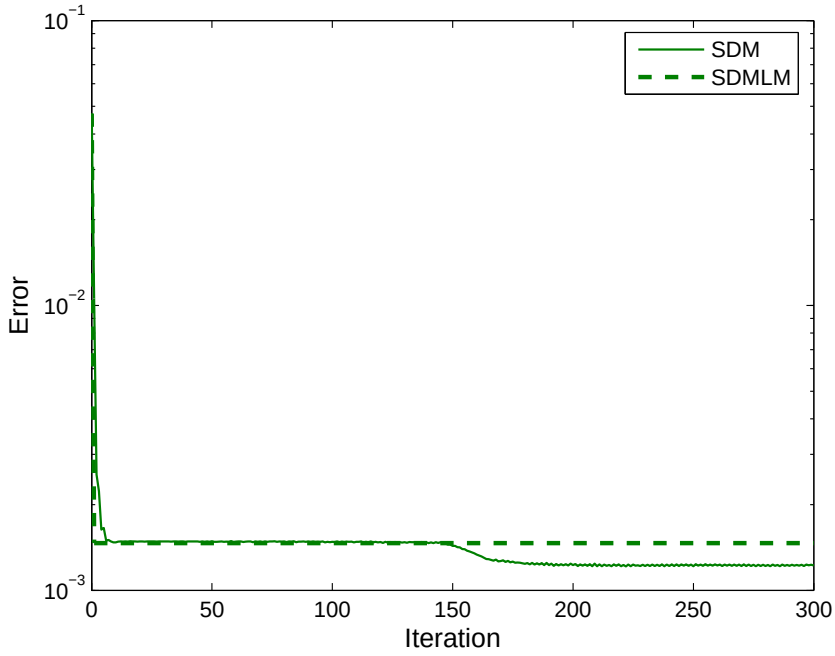


Figure 6.48: Error curves for experiment $I3$

initial mesh used has the dimensions $0.25 \times 0.5 \times 8$ and consists of 14 control points. (See Figure 6.21.) No smoothing term is used. Figure 6.49 shows the error curves for SDM and SDM with the LM method applied. Figure 6.52(d) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (0.711 s) to have an E_{rms} smaller than 0.005.

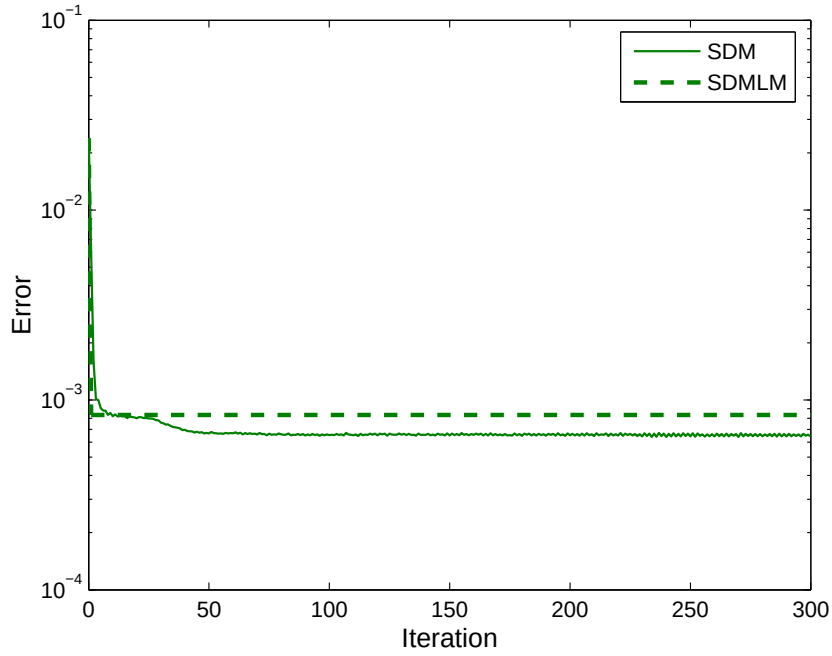


Figure 6.49: Error curves for experiment $I4$

Experiment $I5$ ***(Refer to Figure 6.50)***

In this experiment, a disc-shaped ellipsoid (having the radii 1.0, 1.0 and 0.1), which has been used in experiment $B4$, is used as the target. The initial mesh used has the dimensions $2.0 \times 2.0 \times 0.2$ and consists of 14 control points. (See Figure 6.15.) The initial mesh is placed in an orientation that is orthogonal to the target shape. No smoothing term is used. Figure 6.50 shows the error curves for SDM and SDM with the LM method applied. Figure 6.52(e) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (2.231 s) to have an E_{rms} smaller than 0.001.

Experiment $I6$ ***(Refer to Figure 6.51)***

In this experiment, a disc-shaped ellipsoid (having the radii 1.0, 1.0 and 0.1), which

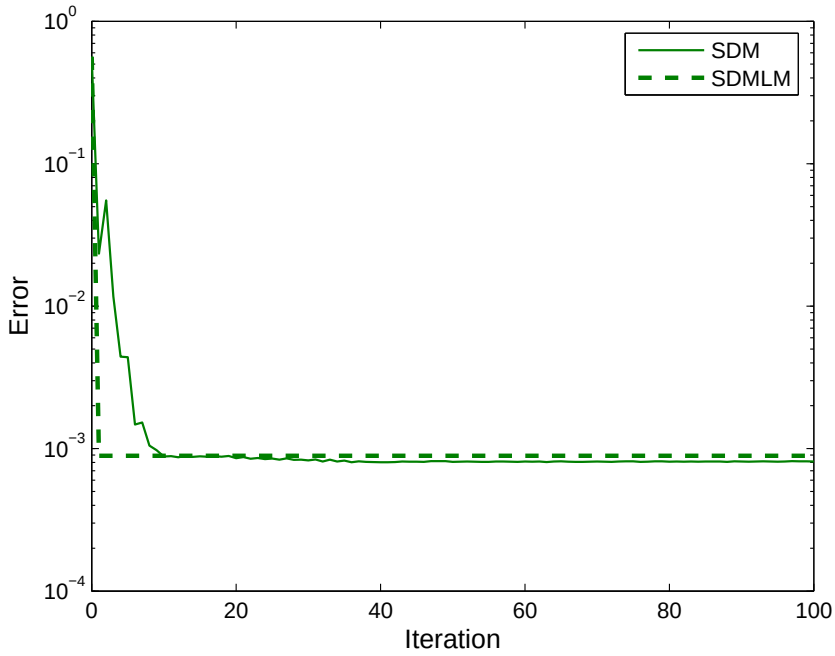


Figure 6.50: Error curves for experiment $I5$

has been used in experiment $D4$, is used as the target. The initial mesh used has the dimensions $4.0 \times 4.0 \times 0.4$ and consists of 14 control points. (See Figure 6.25.) No smoothing term is used. Figure 6.51 shows the error curves for SDM and SDM with the LM method applied. Figure 6.52(f) shows the error curve of the inner iterations for the first iteration of the LM method. SDMLM takes 1 iteration (1.743 s) to have an E_{rms} smaller than 0.001.

Observation: *The LM method improves the stability of SDM, but the effect is not as obvious as that in TDM.*

From the experiment $I1$, it can be observed that the LM method improves the stability of SDM. With the use of the LM method, different local minima can be obtained. For example, in experiments $I3$, $I4$, $I5$ and $I6$, the fitting errors obtained by pure SDM are smaller although the stability has been slightly improved with the use of the LM method.

Similar to the cases in TDM, longer computational time is required when the LM method is applied. The time required by SDMLM is comparable to the time required

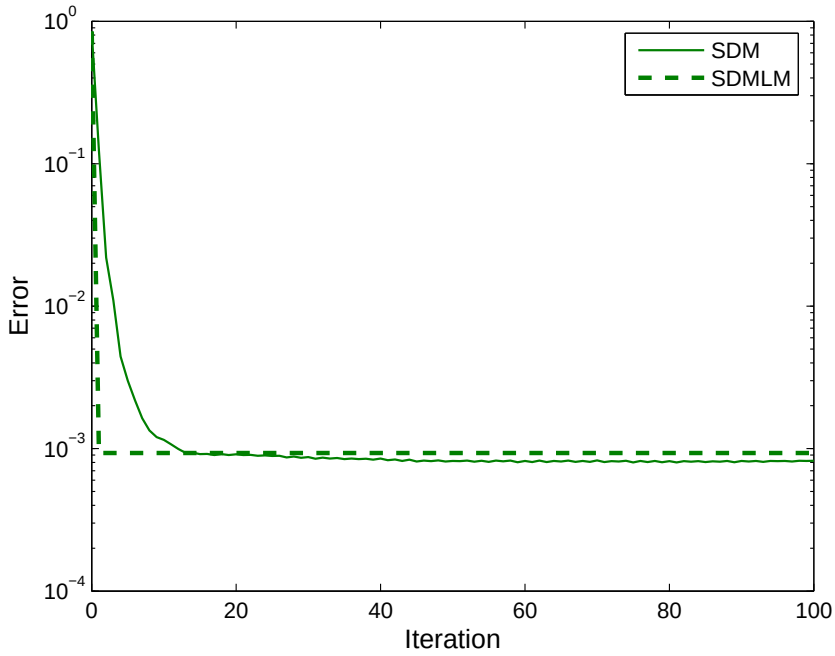
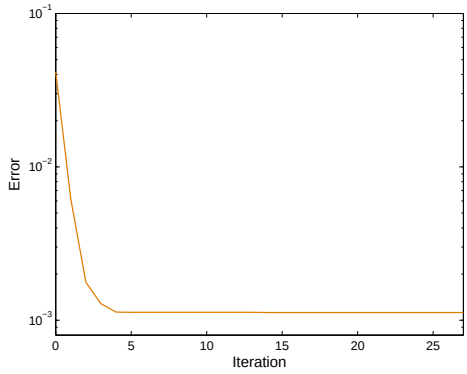


Figure 6.51: Error curves for experiment $I6$

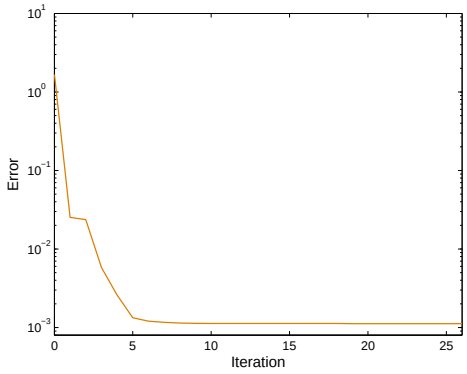
by TDMLM. Since SDM is more stable than TDM, the effect of the LM method is less obvious.

6.1.10 Convergence Behaviors for Optimizations with the Armijo Method

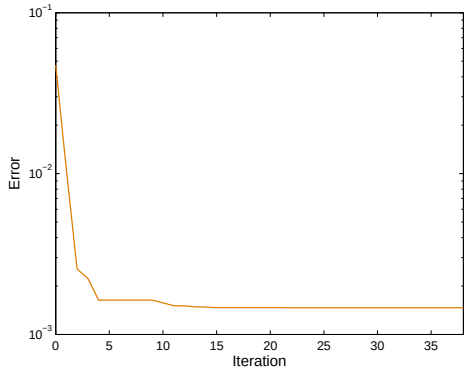
Trust region method, which has been studied in the previous sections, is one way for ensuring the global convergence of an optimization method. Step size control method is another approach for achieving global convergence [47, 59, 67, 68, 72]. In this section, the effectiveness of a step size control strategy, the Armijo method, is investigated. Specifically, when a current point x_c is given, the next position $x_+ = x_c - \alpha p_c$, needs to be found. p_c is the proposed step returned from PDM, TDM and SDM while α is the step size for the movement determined by the Armijo method. In the implementation, α is set to 1 initially. The sufficient condition (described in the previous chapter) is checked. If the sufficient condition is satisfied, the current value of α will be accepted. Otherwise, α will be halved. The process is repeated until a value of α to satisfy the sufficient condition has been found or α has been halved for more than



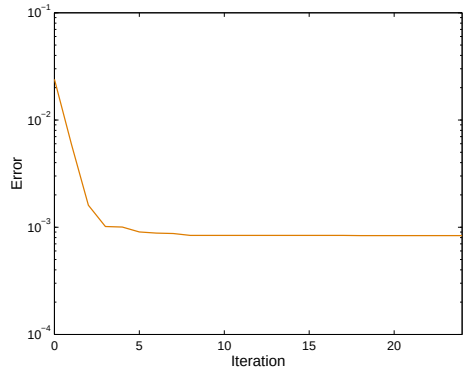
(a)



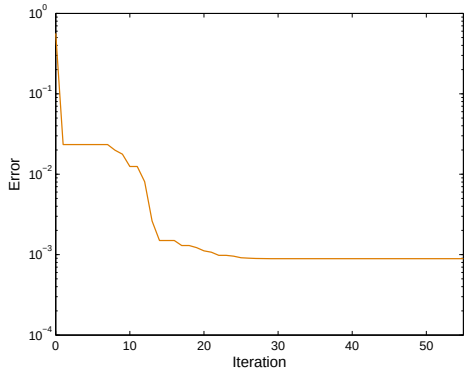
(b)



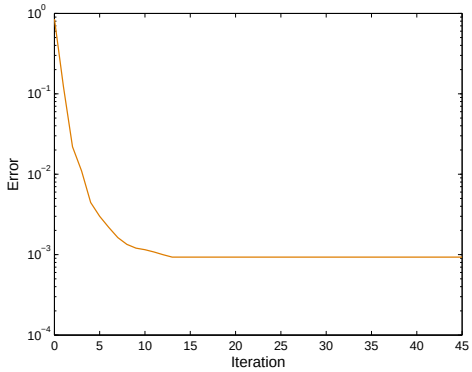
(c)



(d)



(e)



(f)

Figure 6.52: Error curves for the inner iterations of the first iteration in the LM method. (a): Experiment $I1$ (27 inner iterations); (b): Experiment $I2$ (26 inner iterations); (c): Experiment $I3$ (38 inner iterations); (d): Experiment $I4$ (24 inner iterations); (e) Experiment $I5$ (55 inner iterations); (f) Experiment $I6$ (45 inner iterations)

the pre-determined maximum number of time (in our implementation, it is set to 20) to avoid infinite looping.

Experiment $J1$ ***(Refer to Figure 6.53)***

In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0) and the initial mesh (has the dimensions $0.5 \times 1.0 \times 2.0$ and consists of 14 control points) are identical to those in experiment $A2$. (See Figure 6.3.) No smoothing term is used. Figure 6.53 shows the error curves. PDMSC takes 50 iterations (2.004 s) to obtain an E_{rms} larger than 0.002 while SDMSC and TDMSC take 2 iterations (0.080 s) and 1 iteration (0.040 s) respectively.

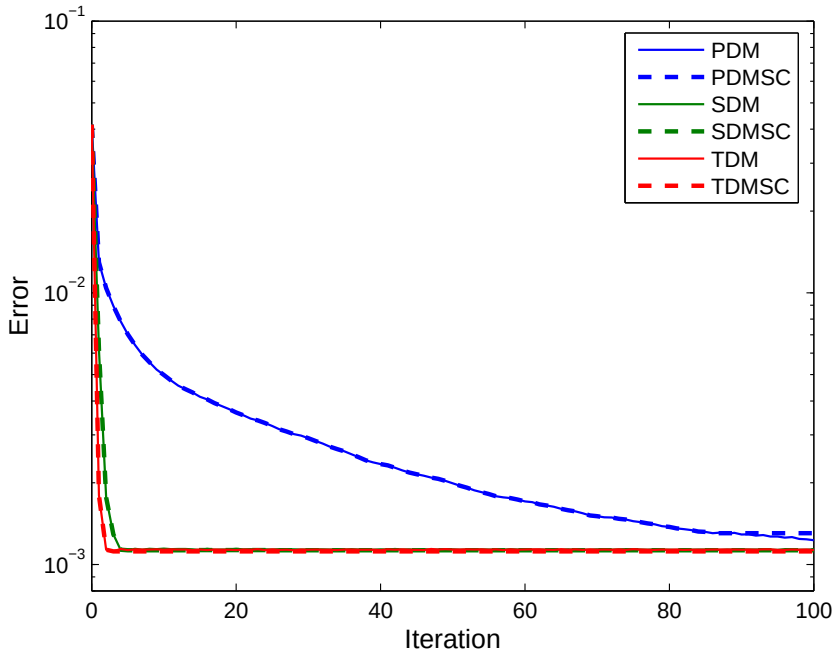


Figure 6.53: Error curves for experiment $J1$

Experiment $J2$ ***(Refer to Figure 6.54)***

In this experiment, the target data (an ellipsoid of which the radii are 0.25, 0.5 and 1.0) and the initial mesh (a $4.0 \times 4.0 \times 4.0$ cube, which consists of 14 control points) are identical to those in experiment $C1$. (See Figure 6.17.) No smoothing term is used. Figure 6.54 shows the error curves. PDMSC has an E_{rms} larger than 0.002 after 100 iterations (4.138 s) while SDMSC and TDMSC just take 5 iterations (0.240 s) and 12 iterations (0.812 s) to obtain E_{rms} smaller than 0.002 respectively.

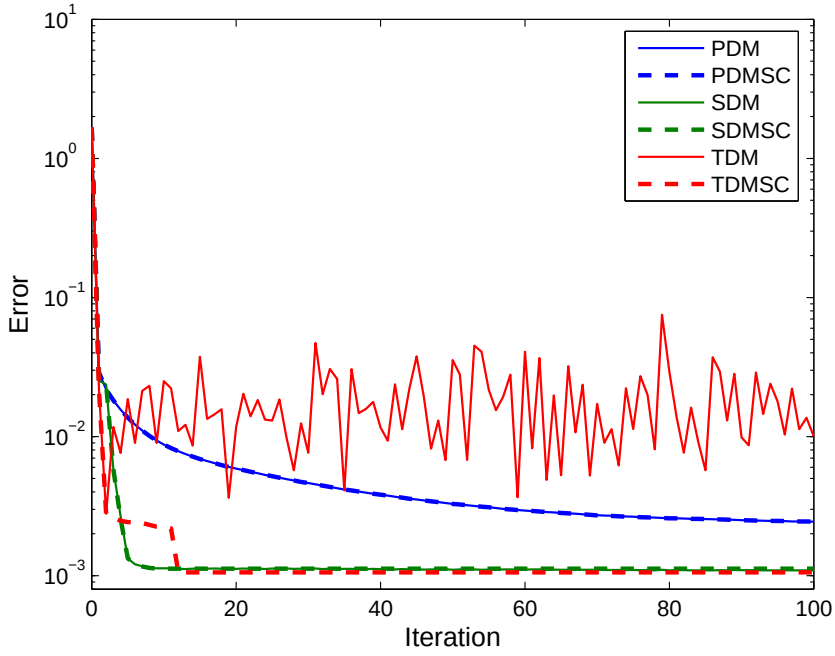


Figure 6.54: Error curves for experiment $J2$

Experiment $J3$ ***(Refer to Figure 6.55)***

In this experiment, an ellipsoid containing large curvatures (having the radii 0.25, 0.5 and 4.0), which has been used in experiment $D1$, is used as the target. The initial mesh used has the dimensions $0.5 \times 1.0 \times 8.0$ and consists of 14 control points. (See Figure 6.19.) No smoothing term is used. Figure 6.55 shows the error curves. PDMSC has an E_{rms} larger than 0.005 after 300 iterations (52.862 s) while SDMSC and TDMSC just take 2 iterations (0.080 s) and 1 iteration (0.030 s) to obtain an E_{rms} smaller than 0.005 respectively.

Experiment $J4$ ***(Refer to Figure 6.56)***

In this experiment, another ellipsoid containing large curvatures (having the radii 0.125, 0.25 and 4), which has been used in experiment $D2$, is used as the target. The initial mesh used has the dimensions $0.25 \times 0.5 \times 8$ and consists of 14 control points. (See Figure 6.21.) No smoothing term is used. Figure 6.56 shows the error curves. PDMSC has an E_{rms} larger than 0.005 after 300 iterations (71.320 s) while SDMSC and TDMSC just take 2 iterations (0.070 s) and 1 iteration (0.030 s) to obtain an E_{rms} smaller than 0.005 respectively.

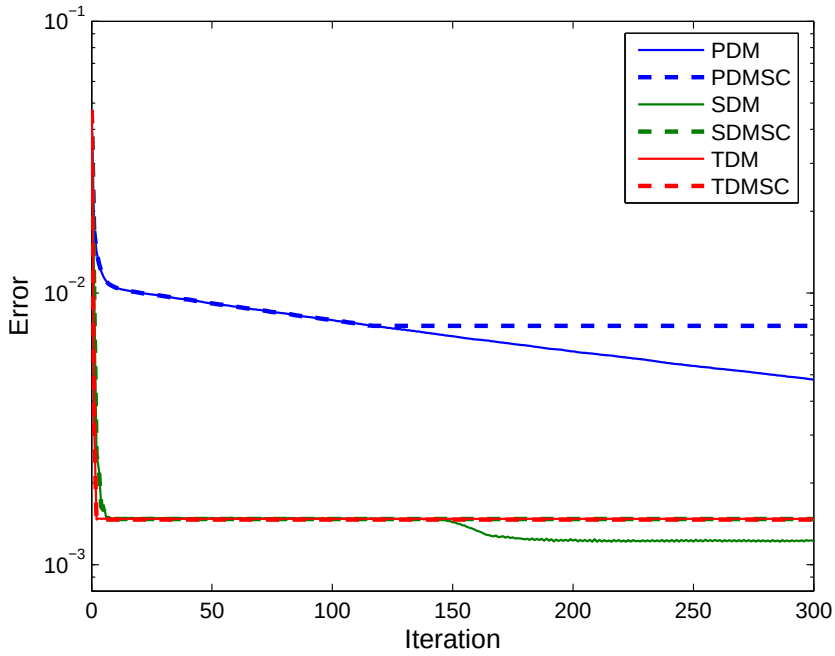


Figure 6.55: Error curves for experiment $J3$

Experiment $J5$ ***(Refer to Figure 6.57)***

In this experiment, the target data (a disc-shaped ellipsoid of which the radii are 1.0, 1.0 and 0.1) and the initial mesh ($2.0 \times 2.0 \times 0.2$, which consists of 14 control points) are identical to those in experiment $B4$. (See Figure 6.15.) The initial mesh is placed in an orientation which is orthogonal to the target shape. No smoothing term is used. Figure 6.57 shows the error curves. PDMSC has an E_{rms} larger than 0.001 after 100 iterations (26.429 s) while SDMSC and TDMSC just take 16 iterations (0.741 s) and 10 iterations (0.501 s) to obtain E_{rms} smaller than 0.001 respectively.

Experiment $J6$ ***(Refer to Figure 6.58)***

In this experiment, the target data (a disc-shaped ellipsoid of which the radii are 1.0, 1.0 and 0.1) and the initial mesh ($4.0 \times 4.0 \times 0.4$, which consists of 14 control points) are identical to those in experiment $D4$. (See Figure 6.25.) No smoothing term is used. Figure 6.58 shows the error curves. PDMSC has an E_{rms} larger than 0.001 after 100 iterations (6.426 s) while SDMSC and TDMSC just take 12 iterations (0.480 s) and 9 iterations (0.410 s) to obtain E_{rms} smaller than 0.001 respectively.

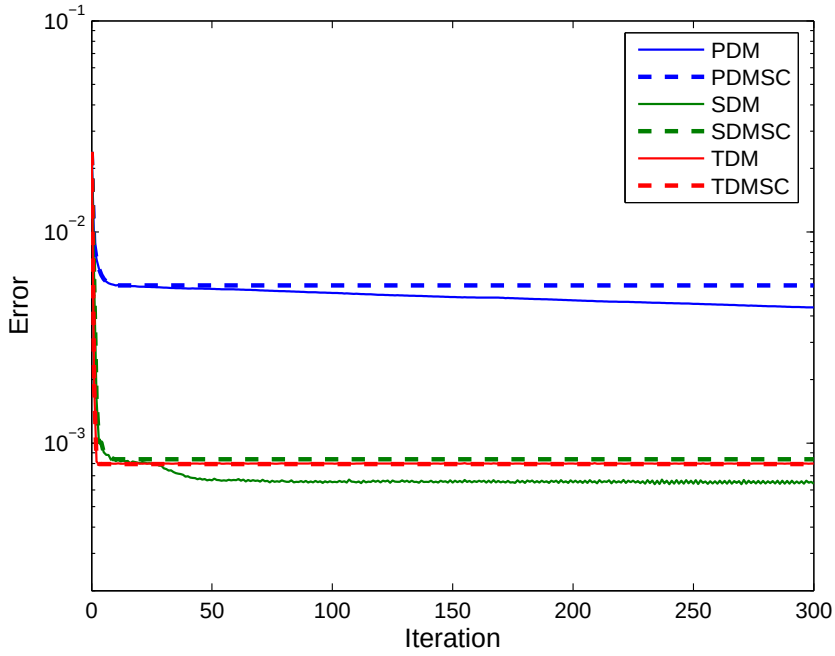


Figure 6.56: Error curves for experiment $J4$

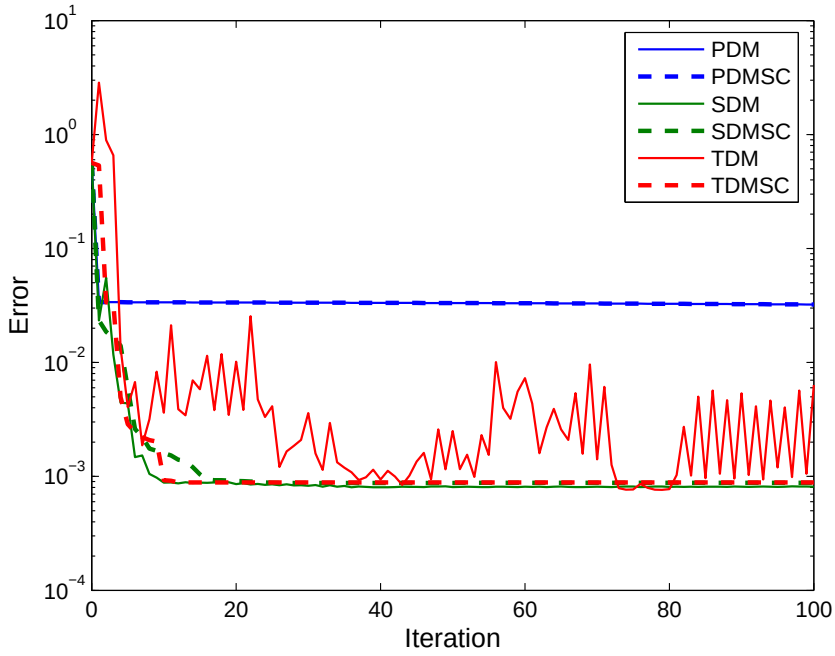


Figure 6.57: Error curves for experiment $J5$

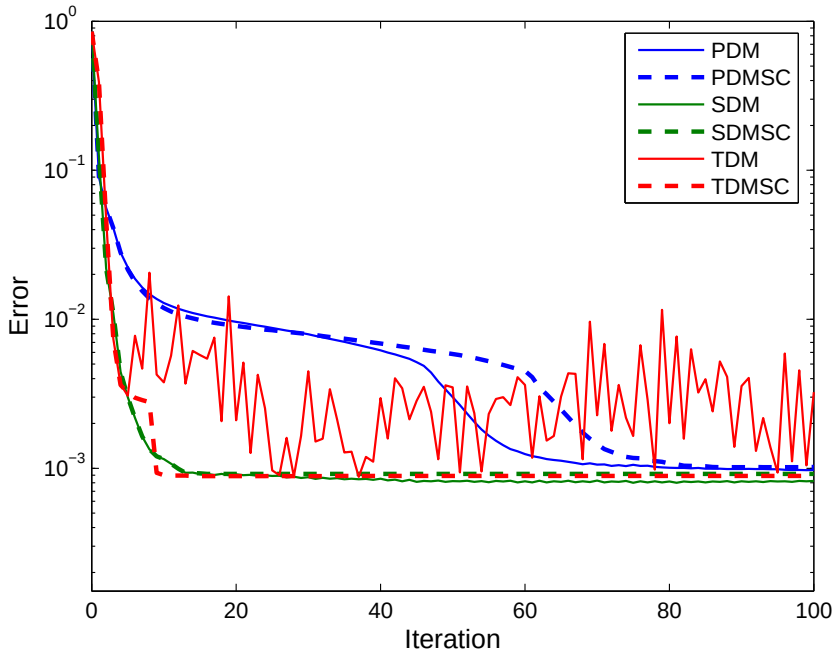


Figure 6.58: Error curves for experiment $J6$

Observation: *The Armijo method improves the stability of PDM, TDM and SDM.*

From the experiments, with the use of the step size control method, all PDM, TDM and SDM are stable. In experiments $J2$, $J5$ and $J6$, in which TDM performs poorly because of the far-away initial mesh, the step size control method improves the fitting error dramatically. The price paid for the step size control method is the much longer computational time. For details statistics, see the table for the time statistics of these experiments in section 6.1.13. When compared with the LM method, it is observed that the cost required by the Armijo method is smaller.

6.1.11 Conclusions from Experiments

From the above experiments, it is observed that TDM and SDM converge much faster than PDM. These agree with the facts that PDM is the gradient descent method which has linear convergence rate while TDM and SDM are the Gauss-Newton method and the Newton method respectively.

SDM and TDM have better capabilities in overcoming the poorly aligned initial meshes. Experiments $B1$-$B4$ demonstrate this. This is due to the fact that tangential movements of control points are hindered in PDM. For more complex target data, if local subdivision is allowed (i.e. multi-staged optimizations), more control points are required for PDM than when compared with SDM and TDM, in additional to the slower convergence rate. Experiments $F2$ and $F3$ show this.

In cases which the initial meshes are far away from the target shapes, such as the data used in experiments $B4$, $C1$ and $D4$, TDM fails to converge. These experimental results can be explained theoretically by the fact that TDM is the Gauss-Newton method and second-order terms $r(x) \bigtriangledown^2 r(x)$ are discarded in the Hessian. The ignored term becomes significant when either $r(x)$ or $\bigtriangledown^2 r(x)$ is larger. With the use of the LM method, TDM can handle these cases in a better way by shifting the local model closer to the steepest descent method if the current local model is detected to be a poor approximation to the goal function. The local model is shifted closer to the Gauss-Newton method if the current local model is detected to be a good approximation to the goal function for faster convergence. Experimental results agree with the theoretical analysis. Besides the LM method, the Armijo method also shows the stabilizing powers in the experiments. In the Armijo method, line search is performed after the original PDM, SDM or TDM is carried out. One difference between the Armijo method and the LM method is that only the step size is modified in the Armijo method while both the search direction and the step size are modified in the LM method.

6.1.12 Stability of TDM: from Computational Point of View

From the experiments, it is observed that pure TDM can be unstable in some situations. Theoretically, we know that it is due to the poor approximations of the Hessian in some cases. From the computational point of view, it can also be noticed that the coefficient matrices formed from TDM have largest condition numbers than those from PDM and SDM. It is well known that large condition numbers can lead to unstable computations; and this fact can also account for the observed unstable behaviors for TDM in some of the experiments. Table 6.1 gives the condition numbers of the coefficient matrices in one particular iteration of PDM, SDM and TDM in experiments $A1$ and $E3$.

	PDM cond. no.	SDM cond. no.	TDM cond. no.
Experiment A1	$64.78, 89.95$	$1.776 \times 10^5, 6.68 \times 10^5$	$1.06 \times 10^6, 3.07 \times 10^6$
Experiment E3	$7.21, 14.29$	$128.24, 356.87$	$153.85, 410.72$

Table 6.1: Condition numbers of coefficient matrices in one particular iteration of PDM, SDM and TDM in experiments $A1$ and $E3$. In each condition number entry, the first number is the 2-norm condition number and the second number is the infinity-norm condition number.

6.1.13 Computational Issues

The codes of PDM and TDM are adapted from that of SDM by replacing the SD error function (2.5) with the PD error function (2.2) and the TD error function (2.3) respectively.

Note that distance field pre-computation is needed for all PDM, TDM and SDM. Normal vector information is needed for TDM and SDM. Curvature pre-computation is only required by SDM. Tables 6.2, 6.3, 6.4 and 6.5 show some time statistics for the experiments.

Regarding the computational efficiency, SDM needs much longer time than PDM and TDM on pre-computation since curvature information is not required in PDM and TDM. In each iteration, SDM takes more time since SDM needs extra time to set up the more complex distance function.

6.2 More Examples

In this section, we present more examples to show that our fitting method works well in fitting subdivision surfaces to more complex target point clouds. The models are scaled such that the longest dimension of the model is 1.0.

Figures 6.59, 6.60, 6.61, 6.62, 6.63 and 6.64 show the data sets for a head (Igea), a ball joint, a rocker arm, an armadillo, a bunny and a buddha (*Igea, the ball joint and the rocker arm come from http://www.cyberware.com. The armadillo, the bunny and the buddha come from http://www-graphics.stanford.edu/data/3Dscanrep/.*). The figures show the initial meshes, the optimized control meshes by SDM, the initial and the optimized subdivision surfaces with color error coding and the shaded optimized sub-

	PDM	SDM	TDM
Experiment A1 (100 iters.)	7.148s	8.281s	7.482s
Experiment A2(100 iters.)	1.602s	1.644s	1.783s
Experiment B1 (300 iters.)	1.733s	1.784s	1.703s
Experiment B2 (300 iters.)	15.230s	15.564s	14.606s
Experiment B3 (100 iters.)	1.862s	2.643s	2.615s
Experiment B4(100 iters.)	1.854s	1.844s	1.854s
Experiment C1 (100 iters.)	1.762s	1.920s	1.833s
Experiment D1 (300 iters.)	4.728s	5.075s	5.125s
Experiment D2 (300 iters.)	5.006s	5.389s	5.238s
Experiment D3 (100 iters.)	1.745s	1.701s	1.602s
Experiment D4 (100 iters.)	1.692s	1.723s	1.543s
Experiment E1 (100 iters.)	10.355s	10.358s	10.210s
Experiment E2 (100 iters)	10.596s	10.675s	10.475s
Experiment E3 (100 iters.)	36.402s	38.006s	38.139s
Experiment E4 (100 iters.)	36.301s	38.469s	38.665s
Experiment F1 (50 iters.)	23.365s	23.635s	24.775s
Experiment F2 (50 iters.)	18.345s	19.450s	19.413s
Experiment F3 (50 iters.)	21.028s	25.169s	20.874s

Table 6.2: Time statistics for experiments $A1$-$F3$. Pre-processing time is not included.

division surfaces. Blue, green, yellow and red represent errors in the ranges $[0, 0.005)$, $[0.005, 0.01)$, $[0.01, 0.015)$ and $[0.015, \infty)$, respectively.
Table 6.6 gives the timing data for the preprocessing steps. At a data point, nearby points that lie within its neighborhood of 0.03 are used for computing the curvatures. Table 6.7 shows error statistics. Table 6.8 shows the breakdown of the time used in different tasks in optimization. From Table 6.8, we observe that the time for generating entries of the matrix of the linear equations is substantial when compared with other parts. Note that the number of data points affects only the time used for the preprocessing steps, but does not affect the time used in the optimization step, which is mainly determined by the number of control points.

We would like to compare the fitting result of the Igea model in Figure 6.60 with that in [35]. In that paper, the projection direction is different from that in our approach. In [35], target data points are projected on the fitting subdivision surface. In our ap-

	E_{rms} Threshold	Time(without LM)	Time (with LM)
Experiment H1	0.002	0.010s	0.551s
Experiment H2	0.002	N.A. (very unstable)	1.593s
Experiment H3	0.005	0.030s	1.221s
Experiment H4	0.005	0.020s	0.511s
Experiment H5	0.001	N.A. (very unstable)	1.662s
Experiment H6	0.001	N.A. (very unstable)	0.962s

Table 6.3: Time statistics for experiments $H1$-$H6$. Time required for E_{rms} to fall below the threshold is presented. Pre-processing time is not included.

	E_{rms}	Time(without LM)	Time (with LM)
Experiment I1	0.002	0.020s	1.011s
Experiment I2	0.002	0.071s	1.042s
Experiment I3	0.005	0.050s	1.321s
Experiment I4	0.005	0.040s	0.711s
Experiment I5	0.001	0.151s	2.231s
Experiment I6	0.001	0.221s	1.743s

Table 6.4: Time statistics for experiments $I1$-$I6$. Time required for E_{rms} to fall below the threshold is presented. Pre-processing time is not included.

proach, sample points on the subdivision surface are projected on the target shape. Projecting target data points on the fitting subdivision surface has an advantage that the details of the target shape will not be missed by the fitting subdivision surface easily. However, projecting target data points on the fitting subdivision surface also implies that the foot points of the target data points on the fitting subdivision surface cannot be precomputed efficiently.

From Table 6.9, we can see that our approach obtains a smaller E_{rms} within a shorter time although using a larger number of control points (The PC on which we run the experiment has the same specification as that used in [35]). It is noticed that our approach gives a slightly higher E_m error. But, since we compute the error by finding the closest target data point for each sample point on the subdivision surface instead of computing the shortest distance between the sample point to the target shape, we have over-estimated the error. Indeed, the average spacing between the target data point in this point cloud is 0.004837, which is in the same order magnitude of our E_m error (0.004495) (The Igea model that we use has the dimensions $0.696 \times 0.997 \times 1.0$).

	E_{rms}	Time(without step size control)	Time (with step size control)
Experiment J1 (SDM)	0.002	0.020s	0.080s
Experiment J1 (TDM)	0.002	0.010s	0.040s
Experiment J2 (SDM)	0.002	0.071s	0.240s
Experiment J2 (TDM)	0.002	N.A. (very unstable)	0.812s
Experiment J3 (SDM)	0.005	0.050s	0.080s
Experiment J3 (TDM)	0.005	0.030s	0.040s
Experiment J4 (SDM)	0.005	0.040s	0.070s
Experiment J4 (TDM)	0.005	0.020s	0.030s
Experiment J5 (SDM)	0.001	0.151s	0.741s
Experiment J5 (TDM)	0.001	N.A. (very unstable)	0.501s
Experiment J6 (SDM)	0.001	0.221s	0.480s
Experiment J6 (TDM)	0.001	N.A. (very unstable)	0.410s

Table 6.5: Time statistics for experiments $J1$-$J6$. Time required for E_{rms} to fall below the threshold is presented. Pre-processing time is not included.

	No. of data points	Curvatures	Distance fields
Ball joint	137062	95.83s	52.51s
Igea	134345	36.07s	215.69s
RockerArm	40177	14.91s	70.48s
Armadillo	172974	4.44s	191.47s
Bunny	35201	212.74s	148.26s
Buddha	543652	4837.67s	200.66s

Table 6.6: Time statistics for pre-computing curvatures and distance fields. Time is measured in seconds.

	No. of iterations	No. of control points	Smoothing term coefficient	E_m	E_{rms}
Ball joint	29	416, 551	0.01, 0.0001	0.0035	0.0008
Igea	14	526, 2464	0.01, 0.00001	0.0029	0.0005
RockerArm	15	870, 950	0.01, 0.00001	0.0018	0.0003
Armadillo	12	602, 9602	0.01, 0.00001	0.0212	0.0020
Bunny	14	919, 996	0.01, 0.00001	0.0037	0.0009
Buddha	7	4668, 4773	0.01, 0.00001	0.0032	0.0004

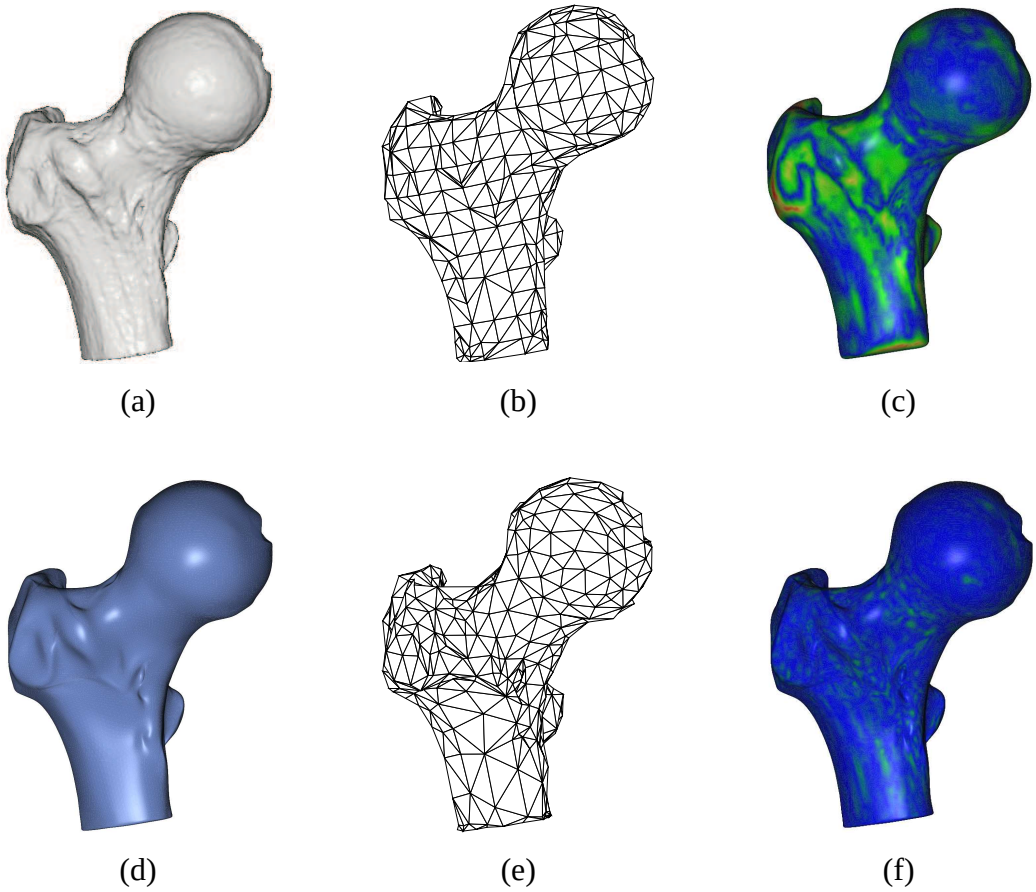
Table 6.7: Statistics for the examples. The numbers in *No. of control points* field refer to the number of control points in the initial control meshes and the final optimized control meshes. The numbers in *Smoothing term coefficient* refer to the initial and the final values for the smoothing term coefficient. The total time does not include the time on pre-computation.

	Equations setup	Equations solving	Error evaluation	Total time
Ball joint	38.68s	31.17s	3.61s	73.47s
Igea	60.04s	51.79s	6.85s	118.69s
RockerArm	41.04s	30.52s	6.86s	78.43s
Armadillo	506.56s	51.75s	65.25s	623.56s
Bunny	39.63s	32.57s	5.13s	77.35s
Buddha	394.09s	112.18s	20.58s	526.86s

Table 6.8: Time statistics. Time is measured in seconds.

	Final no. of control points	E_m (%)	E_{rms} (%)	Time taken (min:sec)
Result in [35]	1553	0.247	0.05755	8:29
Our result	2464	0.185	0.03251	1:58

Table 6.9: Comparison with the approach in [35] for the Igea model. The errors E_m and E_{rms} are expressed in percentage of the diagonal of the model.



(a) (b) (c)

(d) (e) (f)

Figure 6.59: Ball Joint: (a) Point cloud. (137062 points; dimensions: $0.87 \times 0.50 \times 1$) (b) Initial mesh. ($416$ control points) (c) Initial subdivsion surface. (d) Shaded subdivision surface. (e) Optimized mesh. (551 control points) (f) Optimized subdivision surface. Max. Err.: 0.0035; RMS. Err.: 0.0008.

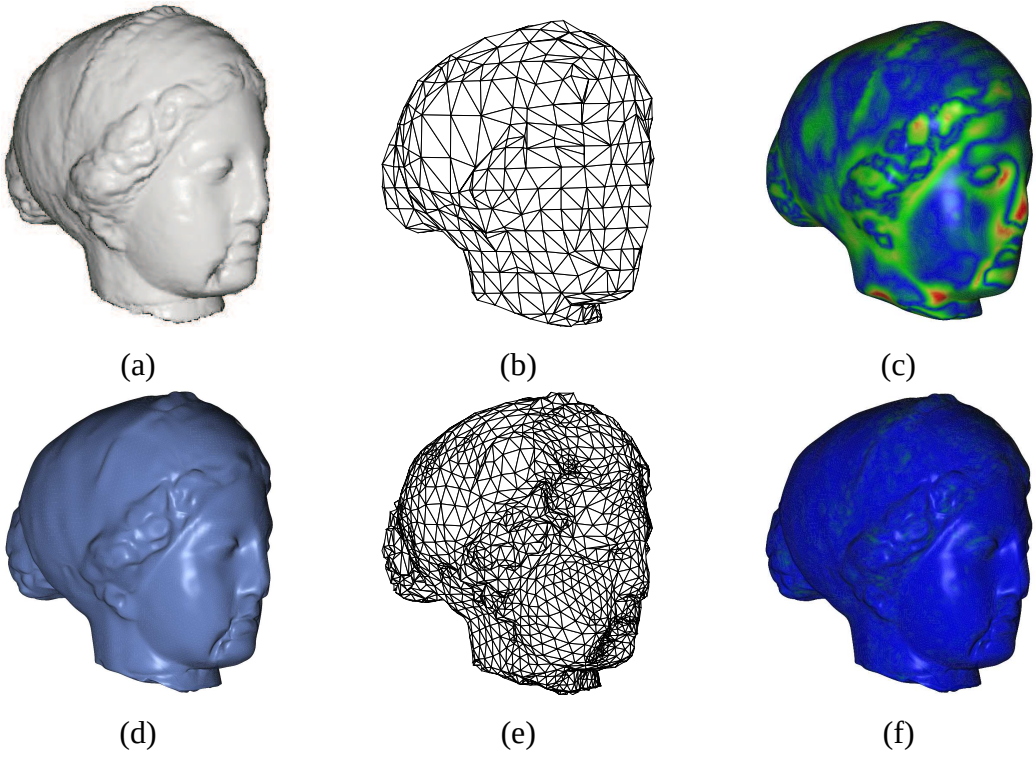
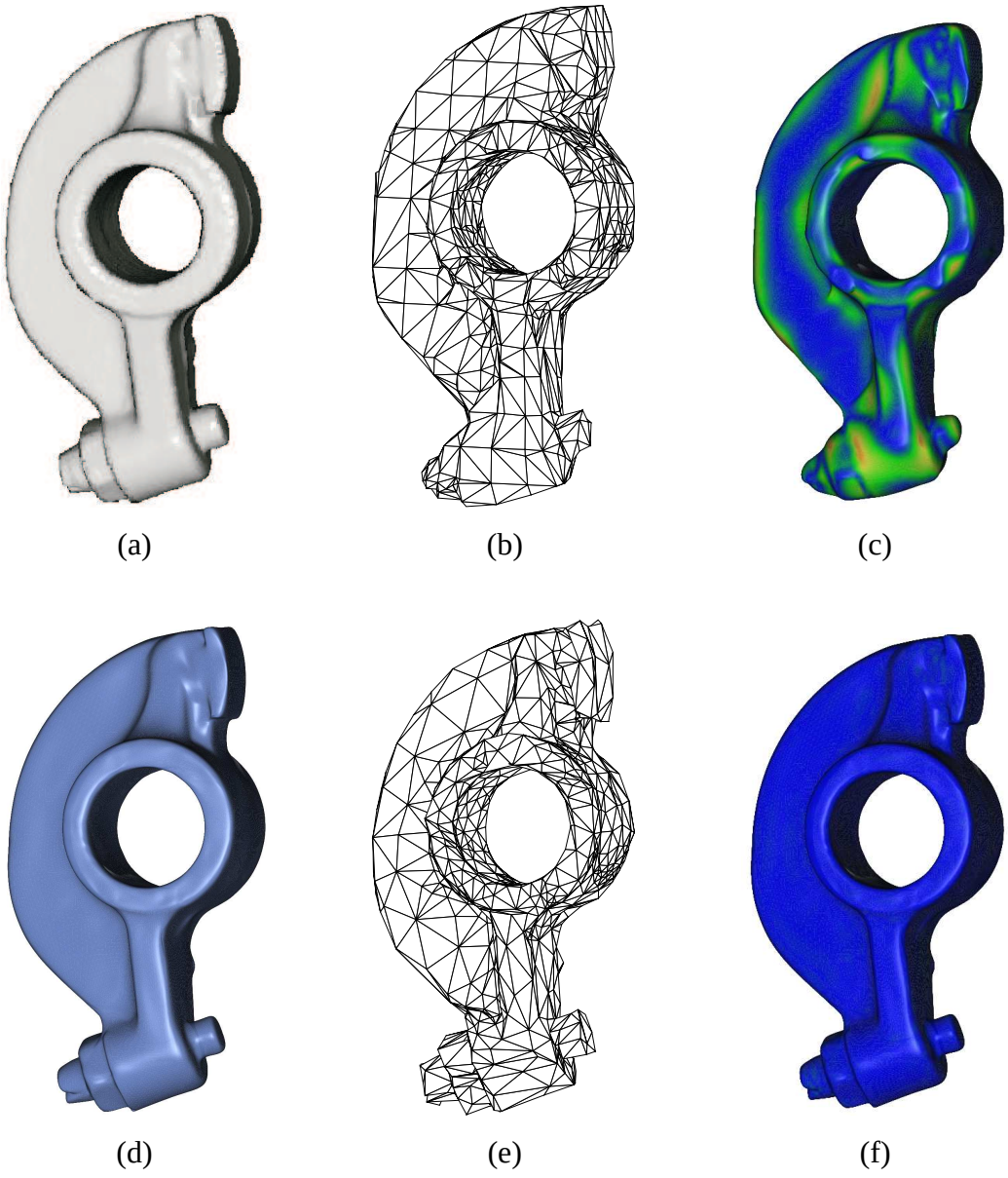


Figure 6.60: Igea: (a) Point cloud. (134345 points; dimensions: $0.70 \times 1 \times 1$) (b) Initial mesh. (526 control points) (c) Initial subdivision surface. (d) Shaded subdivision surface. (e) Optimized mesh. (2464 control points) (f) Optimized subdivision surface. Max. Err: 0.0029; RMS. Err.: 0.0005.



(a) (b) (c)

(d) (e) (f)

Figure 6.61: RockerArm: (a) Point cloud. (40177 points; dimensions: $0.51 \times 1 \times 0.30$) (b) Initial mesh. ($870$ control points) (c) Initial subdivision surface. (d) Optimized subdivision surface. (e) Optimized mesh. (950 control points) (f) Optimized subdivision surface. Max. Err.: 0.0018; RMS. Err.: 0.0003.

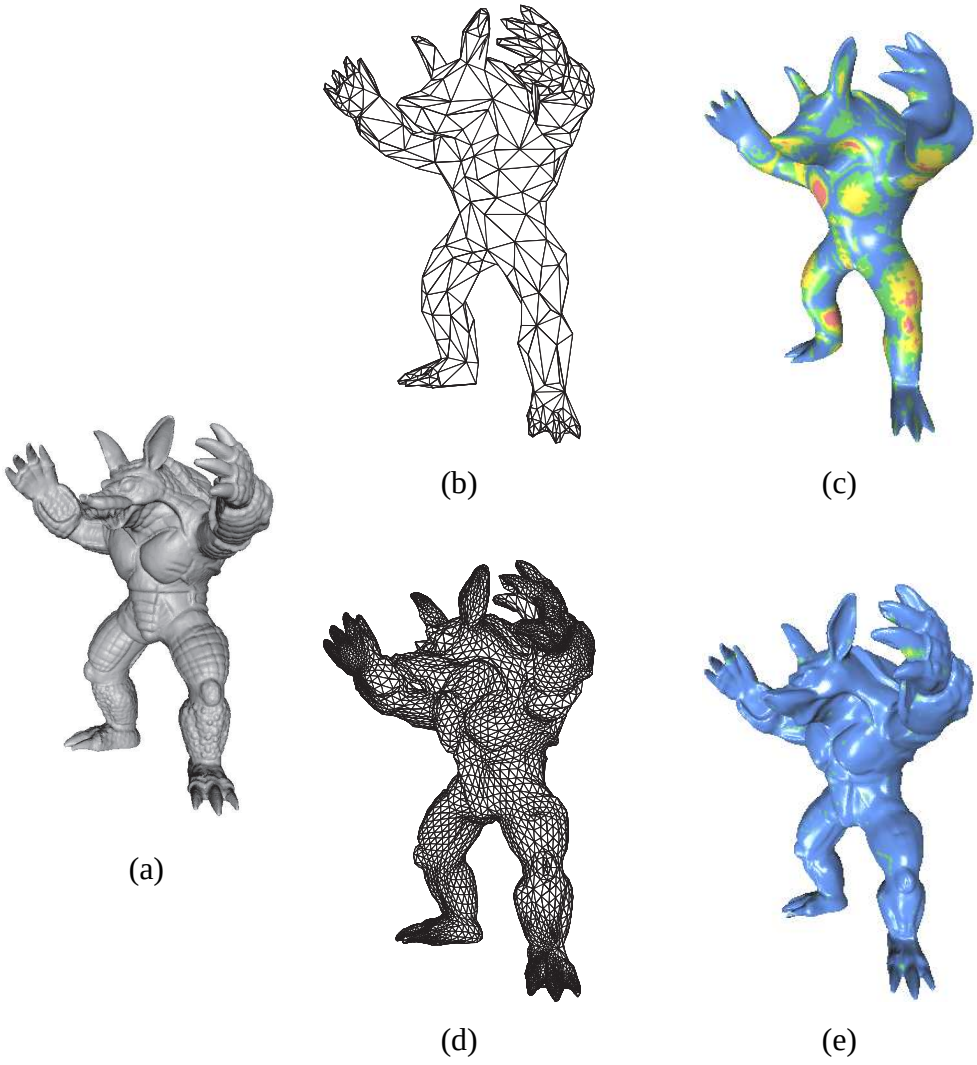
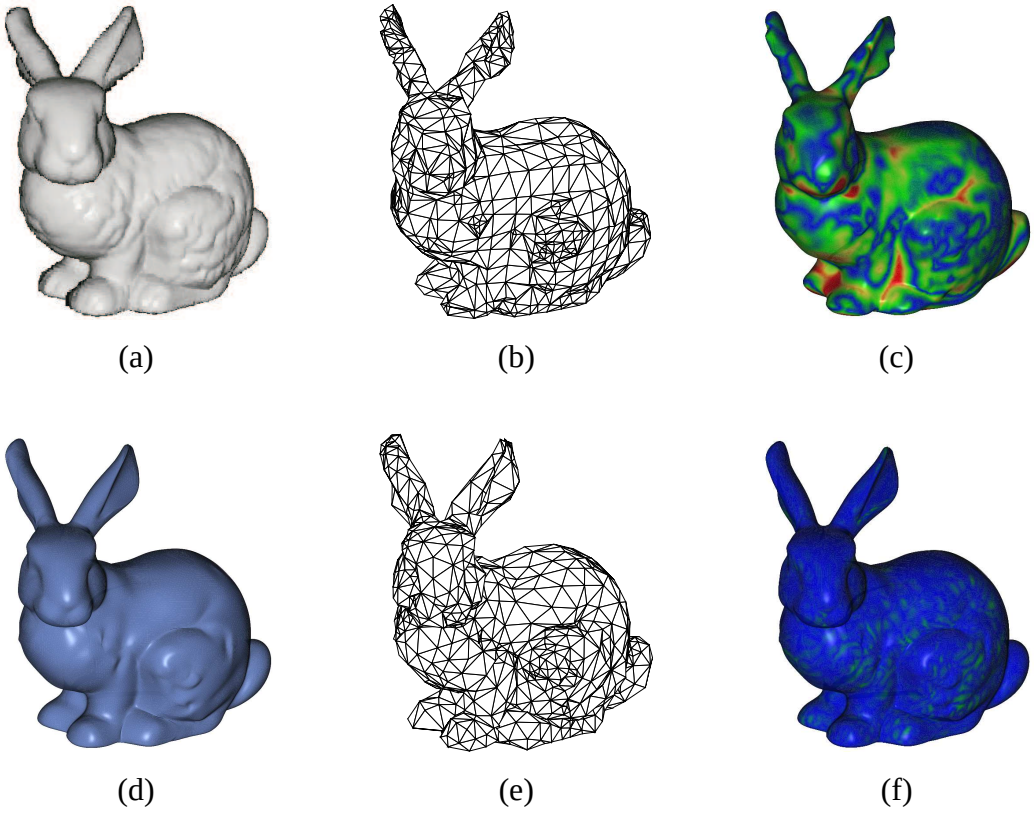


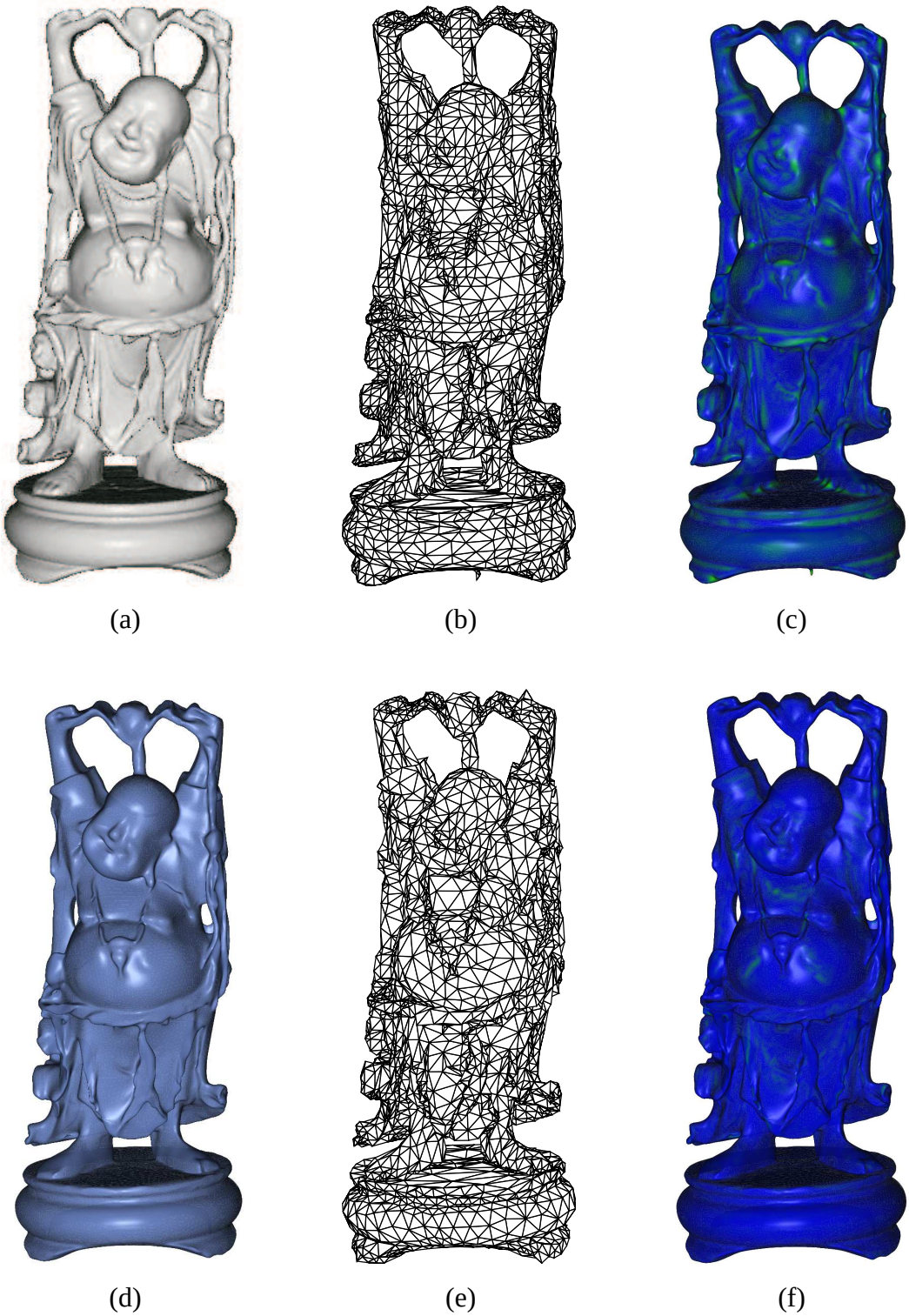
Figure 6.62: Armadillo: (a) Point cloud. (172974 points; dimensions: $0.84 \times 0.76 \times 1$) (b) Initial mesh. ($602$ control points) (c) Initial subdivision surface. (d) Optimized mesh. (9602 control points) (e) Optimized subdivision surface. Max. Err.: 0.0212; RMS. Err.: 0.0020.



(a) (b) (c)

(d) (e) (f)

Figure 6.63: Bunny: (a) Point cloud. (35201 points; dimensions: $1 \times 0.78 \times 0.99$) (b) Initial mesh. ($919$ control points) (c) Initial subdivision surface. (d) Optimized subdivision surface. (e) Optimized mesh. (996 control points) (f) Optimized subdivision surface. Max. Err.: 0.0037; RMS. Err.: 0.0009.



(a) (b) (c)

(d) (e) (f)

Figure 6.64: Buddha: (a) Point cloud. (543652 points; dimensions: $0.41 \times 0.41 \times 1$) (b) Initial mesh. ($4668$ control points) (c) Initial subdivision surface. (d) Optimized subdivision surface. (e) Optimized mesh. (4773 control points) (f) Optimized subdivision surface. Max. Err.: 0.0032; RMS. Err.: 0.0004.

Chapter 7

Conclusion and Future Work

In this piece of research work, we have studied different optimization methods, which include PDM, TDM, SDM, for fitting subdivision surfaces to unorganized points. Among them, SDM is newly introduced to the problem of subdivision surface fitting. On top of that, we apply the LM method (a trust region method) and the Armijo method (a line search method), to improve the stability of the fitting process. From the experiments, we observe that the behaviors of various fitting methods agree with what the underlying theories in optimization predict.

An important contribution of this thesis is that we give a clear picture about the relationships between different distance error functions in goal functions in surface fitting problem and the optimization theories. Specifically, we show that PDM, TDM and SDM are indeed the gradient descent method, the Gauss-Newton method and the Newton method in optimization theories. Theoretically, we prove that SDM is derived from the standard Newton method. To ensure the positive definiteness of the Hessian, slight modification is applied at some circumstances.

Our experiments show that both SDM and TDM converge much faster than PDM. Since the Newton method has quadratic convergence, this result explains the superior convergence behavior of SDM over PDM, which is known to have linear convergence. TDM, as the Gauss-Newton method, is also expected to have faster convergence rate than the commonly used PDM. From experiments in which large curvature regions are present in the target data, TDM does not work well around the large curvature regions. And this behavior can be explained by the fact that some curvatures-related second order terms in the Hessian are discarded in the Gauss-Newton method.

With the use of the LM method, the agreement between the local approximated model and the actual goal function is checked. As demonstrated in the experiments, the LM method solves the stability problem of TDM when the target shapes contain large curvature regions.

With the use of the Armijo method, the step size of a step is computed after the direction is determined by some other methods. As shown in the experiments, the Armijo method improves the stability of the optimization process.

There are several problems that still call for further research. First, a more effective method needs to be devised to determine the coefficient λ for the smoothness term F_s. Second, features like edges and corners in data sets can be detected in order to make the subdivision surfaces preserve the detected features in a better way. Third, we would like to study the various fitting methods described in this thesis to fitting surfaces to noisy point clouds, for which it is difficult to have accurate curvature estimation required by SDM presented here. Fourth, the fitting methods described in this thesis should be able to be adapted to deal with target shapes with open boundaries. Fifth, SDM should be able to handle the 3D deformable model registration problem after appropriate modifications. Finally, the fitting approach described here is still a local optimization technique. In this regard, there is a need to design an active subdivision scheme that allows the fitting surface to evolve from a simple initial shape to converge to a given target shape in a global manner, as what has been done in other active contour approaches such as the snake method [12, 13] and the level-set method [15–19].

Bibliography

[1] C. Loop, Smooth Subdivision Surfaces based on Triangles, Master's thesis, Department of Mathematics, University of Utah (1987).

[2] D. Zorin, Subdivision and Multiresolution Surface Representations, Ph.D. thesis, Caltech, Pasadena, Calif (1997).

[3] D. Zorin, P. Schröder, A. DeRose, J. Stam, L. Kobbelt, J. Warren, Subdivision for Modeling and Simulation, in: SIGGRAPH '99 Course Notes, 1999.

[4] H. Hoppe, Surface Reconstruction from Unorganized Points, Ph.D. thesis, the University of Washington (1994).

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface Reconstruction from Unorganized Points, in: Proc. SIGGRAPH '92, 1992, pp. 71–78.

[6] M. Halstead, M. Kass, T. DeRose, Efficient, Fair Interpolation using Catmull-Clark Surfaces, in: Proc. SIGGRAPH '93, 1993, pp. 35–44.

[7] W. Ma, J. P. Kruth, Parameterization of Randomly Measured Points for Least Squares Fitting of B-spline Curves and Surfaces, Computer-Aided Design 27 (9) (1995) 663–675.

[8] W. Ma, N. Zhao, Catmull-Clark Surface Fitting for Reverse Engineering Applications, in: Proc. GMP 2000, 2000, pp. 274–284.

[9] W. Ma, N. Zhao, Smooth Multiple B-spline Surface Fitting with Catmull-Clark Subdivision Surfaces for Extraordinary Corner Patches, The Visual Computer 18 (2002) 415–436.

[10] M. Marinov, L. Kobbelt, Optimization Techniques for Approximation with Subdivision Surfaces, in: ACM Symposium on Solid Modeling and Applications, 2004, pp. 1–10.

[11] H. Suzuki, S. Takeuchi, T. Kanai, Subdivision Surface Fitting to a Range of Points, in: The Seventh Pacific Conference on Computer Graphics and Applications, 1999, pp. 158–167.

[12] A. Blake, M. Isard, Active Contours, Springer, 1998.

[13] M. Kass, A. Witkins, D. Terzopoulos, Snakes – Active Contour Models, International Journal of Computer Vision 1 (4) (1987) 321–331.

[14] D. Liersch, A. Sovakar, L. Kobbelt, Parameter Reduction and Automatic Generation of Active Shape Models, Workshop Bildverarbeitung fur die Medizin.

[15] R. Malladi, J. A. Sethian, B. C. Vemuri, Shape Modeling with Front Propagation: A Level Set Approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (2) (1995) 158–175.

[16] J. A. Sethian, Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.

[17] R. Whitaker, A Level-Set Approach to 3D Reconstruction from Range Data, The International Journal of Computer Vision 29 (3) (1998) 203–231.

[18] H. K. Zhao, T. Chan, S. Osher, A Variational Level Set Approach to Multiphase Motion, Journal of Computational Physics 127 (1996) 179–195.

[19] H. K. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and Non-parametric Shape Reconstruction from Unorganized Data using a Variational Level Set Method, Computer Vision and Image Understanding 80 (2000) 295–319.

[20] C. Lürig, L. Kobbelt, T. Ertl, Hierarchical Solutions for the Deformable Surface Problem in Visualization, Graphical Models 62 (2000) 2–18.

[21] Z. Wood, M. Desbrun, P. Schröder, D. Breen, Semi-Regular Mesh Extraction from Volumes, in: Proc. IEEE Visualization, 2000, pp. 275–282.

[22] Y. Duan, H. Qin, Extracting Boundary Surface of Arbitrary Topology from Volumetric Datasets, in: Volume Graphics Workshop 2001, 2001, pp. 149–158.

[23] Y. Duan, H. Qin, Intelligent Balloon: A Subdivision-based Deformable Model for Surface Reconstruction of Arbitrary Unknown Topology, in: Proceedings of

the Sixth ACM Symposium on Solid Modeling and Applications, 2001, pp. 47–58.

[24] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level Partition of Unity Implicits, in: Proc. SIGGRAPH '03, 2003, pp. 463–470.

[25] I. Tobor, P. Reuter, C. Schlick, Efficient Reconstruction of Large Scattered Geometric Datasets using the Partition of Unity and Radial Basis Functions, Journal of WSCG 2004 12 (3) (2004) 467–474.

[26] A. Adamson, M. Alexa, Approximating and Intersecting Surfaces from Points, in: Proceedings of the Eurographics Symposium on Geometry Processing, 2003, pp. 245–254.

[27] A. Adamson, M. Alexa, Approximating Bounded, Non-orientable Surfaces from Points, in: Shape Modeling International 2004, 2004, pp. 243–252.

[28] N. Litke, A. Levin, P. Schröder, Fitting Subdivision Surfaces, in: Proc. IEEE Visualization 2001, 2001, pp. 319–324.

[29] B. Jüttler, A. Felis, Least-Squares Fitting of Algebraic Spline Surfaces, Advances in Computational Mathematics 17 (2002) 135–152.

[30] T. Kanai, MeshToSS: Converting Subdivision Surfaces from Dense Meshes, in: Proceedings of the Vision Modeling and Visualization Conference 2001, 2001, pp. 325–332.

[31] W.-K. Jeong, I. Ivrissimtzis, H.-P. Seidel, Neural Meshes: Statistical Learning Based on Normals, in: Pacific Graphics 2003, 2003, pp. 404–408.

[32] I. Ivrissimtzis, W.-K. Jeong, H.-P. Seidel, Neural Meshes: Statistical Learning Methods in Surface Reconstruction, Technical Report MPI-1-2003-4-007, Max-Planck-Institut für Informatik (2003).

[33] T. Speer, M. Kuppe, J. Hoschek, Global Reparametrization for Curve Approximation, Computer Aided Geometric Design 15 (1998) 869–877.

[34] H. Pottmann, S. Leopoldseder, A Concept for Parametric Surface Fitting which avoids the Parametrization Problem, Computer Aided Geometric Design 20 (2003) 343–362.

[35] M. Marinov, L. Kobbelt, Optimization Methods for Scattered Data Approximation with Subdivision Surfaces, Graphical Models.

[36] G. Chaikin, An Algorithm for High Speed Curve Generation, Computer Graphics and Image Processing 3 (1974) 346–349.

[37] R. Riesenfeld, On Chaikin's Algorithm, IEEE Computer Graphics and Applications 4 (3) (1975) 304–310.

[38] D. Doo, M. Sabin, A Subdivision Algorithm for Smoothing Down Irregularly Shaped Polyhedrons, in: Proceeding. Int'l Conf. Interactive Techniques in Computer Aided Design, 1978, pp. 157–165.

[39] E. Catmull, J. Clark, Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes, Computer Aided Design 10 (6) (1978) 350–355.

[40] N. Dyn, J. A. G. amd D. Levin, A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control, ACM Transactions on Graphics 9 (1990) 160–169.

[41] L. Kobbelt, Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology, Computer Graphics Forum 15 (3) (1996) 409–420.

[42] J. Hoschek, Intrinsic Parameterization for Approximation, Computer Aided Geometric Design 5.

[43] J. Hoscheck, D. Lasser, Fundamentals of Computer Aided Geometric Design, AK Peters, 1993.

[44] V. Weiss, L. Andor, G. Renner, T. Varady, Advanced Surface Fitting Techniques, Computer Aided Geometric Design 19 (1) (2002) 19–42.

[45] A. Björck, Numerical Methods for Least Squares Problems, Mathematics Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[46] J. Dennis, R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Mathematics Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[47] C. Kelley, Iterative Methods for Optimization, Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[48] H. Pottmann, S. Leopoldseder, M. Hofer, Approximation with Active B-spline Curves and Surfaces, in: Proc. Pacific Graphics 02, 2002, pp. 8–25.

[49] H. Pottmann, M. Hofer, Geometry of the Squared Distance Function to Curves and Surfaces, Visualization and Mathematics III, Springer (2003) 223–244.

[50] H. P. Yang, W. Wang, J. G. Sun, Control Point Adjustment for B-spline Curve Approximation, Computer-Aided Design 36 (2004) 639–652.

[51] W. Wang, H. Pottmann, Y. Liu, Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization, ACM Transactions on Graphics.

[52] S. Schaefer, J. Warren, Dual Marching Cubes: Primal Contouring of Dual Grids, in: Proc. Pacific Graphics'04, 2004, pp. 70–76.

[53] J. Stam, Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values, in: Proc. SIGGRAPH'98, 1998, pp. 395–404.

[54] J. Stam, Evaluation of Loop Subdivision Surfaces, in: SIGGRAPH'98 CDROM Proceedings, 1998.

[55] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C++: The Art of Scientific Computing, Cambridge University Press, 2002.

[56] C. Kelley, Iterative Methods for Linear and Nonlinear Equations, Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[57] G. H. Golub, C. F. V. Loan, Matrix Computations, The Johns Hopkins University Press, 1996.

[58] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer, 1997.

[59] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 1999.

[60] H. van der Vorst, Iterative Krylov Methods for Large Linear Systems, Cambridge, 2003.

[61] J. Goldfeather, V. Interrante, A Novel Cubic-Order Algorithm for Approximating Principal Direction Vectors, ACM Transactions on Graphics 23 (1) (2004) 45–63.

[62] S. Leopoldseder, H. Pottmann, H. K. Zhao, The d^2 Tree: A Hierarchical Representation of the Squared Distance Field, Tech. Rep. 101, Institute of Geometry, Vienna University of Technology (2003).

[63] W. Lorensen, H. Cline, Marching Cubes: A High Resolution 3-D Surface Construction Algorithm, Computer Graphics 21 (1987) 163–169.

[64] M. Garland, P. Heckbert, Surface Simplification using Quadric Error Metrics, in: Proc. SIGGRAPH '97, 1997, pp. 209–216.

[65] W.-K. Jeong, C.-H. Kim, Direct Reconstruction of a Displaced Subdivision Surface from Unorganized Points, Graphical Models 64 (2) (2002) 78–93.

[66] R. Bank, A. Sherman, A. Weiser, Refinement Algorithm and Data Structures for Regular Local Mesh Refinement, in: Scientific Computing, 1983, pp. 3–17.

[67] G. Nemhauser, R. Kan, M. Todd, Handbooks in Operations Research and Management Science Volume 1: Optimization, North-Holland, 1989.

[68] S. Nash, A. Sofer, Linear and Nonlinear Programming, McGraw-Hill, 1996.

[69] K. Madsen, H. Nielsen, O. Tingleff, Methods for Non-Linear Least Squares Problems, Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[70] P. Frandsen, K. Jonasson, H. Nielsen, O. Tingleff, Unconstrained Optimization, Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

[71] A. Ruhe, P.-A. Wedin, Algorithms for Separable Nonlinear Least Squares Problems, SIAM Review 22 (1980) 318–337.

[72] D. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, Massachusetts, 1999.