An Adaptive B-Spline Method for Low-order Image Reconstruction Problems

Xin Li

July 6, 2001

Contents

List of Figures

Abstract

A common problem in signal processing is to estimate the structure of an object from noisy indirect measurements linearly related to the desired image. These problems are broadly known as *inverse problems*. A key feature which complicates the solution to such problems is their ill-posedness. That is, small perturbations in the data arising e.g. from noise can and do lead to severe, non-physical artifacts in the recovered image. The process of stabilizing these problems is known as regularization of which Tikhonov regularization is one of the most common. While this approach leads to a simple linear least squares problem to solve for generating the reconstruction, it has the unfortunate side effect of producing smooth images thereby obscuring important features such as edges. Therefore, over the past decade there has been much work in the development of edge-preserving regularizers. This technique leads to image estimates in which the important features are retained, but computationally they require the solution of a nonlinear least squares problem, a daunting task in many practical multi-dimensional applications.

In this thesis we explore low-order models for reducing the complexity of the reconstruction process. Specifically, B-Splines are used to approximate the object. If a "proper" collection B-Splines are chosen so that the object can be efficiently represented using a few basis functions, the dimensionality of the underlying problem will be significantly decreased. Consequently, an optimum distribution of splines needs to be determined. Here, an adaptive refining and pruning algorithm is developed to

solve the problem. The refining part is based on curvature information, in which the intuition is that a relatively dense set of fine scale basis elements should cluster near regions of high curvature while a sparse collection of basis vectors are required to adequately represent the object over spatially smooth areas. The pruning part is a greedy search algorithm to find and delete redundant knots based on the estimation of a weight associated with each basis vector. The overall algorithm iterates by inserting and deleting knots and end up with much fewer knots than pixels to represent the object, while the estimation error is within a certain tolerance. Thus, an efficient reconstruction can be obtained which significantly reduces the complexity of the problem.

In this thesis, the adaptive B-Spline method is applied to a cross-well tomography problem. The problem comes from the application of finding underground pollution plumes. Cross-well tomography method is applied by placing arrays of electromagnetic transmitters and receivers along the boundaries of the interested region. By utilizing inverse scattering method, a linear inverse model is set up and furthermore the adaptive B-Spline method described above is applied. The simulation results show that the B-Spline method reduces the dimensional complexity by 90%, compared with that of the pixel-based method, and decreases time complexity by 50%, without significantly degrading the estimation.

Chapter 1

Introduction

A common problem in signal processing is to estimate the structure of an object from noisy indirect measurements. These problems are broadly known as *inverse problems* with many applications such as radar/medical imaging, mine detecting and image reconstruction [1] [2] [3] [4]. Among these, a common application is the cross-well tomography problem, where we need to locate and estimate the buried objects by observations of scattered electromagnetics radiation taken along the boundaries [6] [7] .

In this thesis, we will first introduce the cross-well tomography problem, then set up the linear inverse model based on inverse scattering theory [8] [18]. The difficulty of solving the linear inverse problem lies in the ill-posed nature of the degradation kernel. The classical solutions tend to have large high-frequency artifacts which contaminate the solution. To make the solution more stable, i.e. smooth out those high-frequency components, a smoothing constraint is added to the underlying optimization problem, a procedure called *regularization*. Many regularization methods have been widely explored [19] [20], among which the Tikhonov regularization is the most common [21].

Tikhonov regularization actually adds a smoothing constraint on the solution so

that the high-frequency artifacts are filtered out. However, at the same time, it also blurs the edges of the object, which are frequently the most interesting areas to us. Therefore, more specific constraint should be added to preserve the edges. Edge-preserving regularization using non-quadratic regularizers is such an idea [24] [25] [26] [27] . The assumption is that, in the reconstruction, large changes in the parameters we are estimating are likely to be true edges which we want to preserve, while smaller ones are mostly artifacts which we want to smooth out. Use of such a regularizer leads to non-linear least squares problem, which can be solved by the Gauss-Newton method [28] or other iterative methods.

In practical multi-dimensional cases, the Gauss-Newton method needs to produce an estimate of all the pixels at each iteration, thus the complexity is very large. One idea is to represent the object by efficiently using some basis functions to aggregate the pixels. A straightforward solution is to use shape-based template if the shape of the objects are known [29]. However, in most cases, the structure of the object is complex and unknown to us, so a general basis which can be suitable to a wide variety of objects should be used.

B-Splines have been thoroughly explored and proved to be very useful in approximation and have been widely used in computer aided designs [30] [32] [31]. The B-Spline basis has several good properties and can be easily manipulated by dealing with knots. Furthermore, some efficient algorithms of construction, evaluation, and knot insertion/deletion exist [36] [37] [38] [39], so that not too much effort is needed to deal with B-Splines themselves.

If proper B-Splines are chosen so that the object can be efficiently represented using few basis functions, the dimension of the problem will be significantly decreased. To select those proper B-Splines, we should find an appropriate number of B-Splines to be used and a suitable distribution of these B-Splines. The number of basis functions to be used is the number of unknowns in the revised problem, thus determining the

complexity of the B-Spline-based problem. The placements of these knot represent where to put those B-Splines, which affects how close the object can be approximated by using these fixed number of knots. The intuition is to place few knots at flat regions and more knots at rough regions with more details [40] [41] [42]. In this thesis, an adaptive knot refining and pruning algorithm is developed to solve the problem. The knot refining part efficiently inserts knots based on curvature information. The knot pruning part is a greedy search algorithm to find and delete redundant knots based on an estimate of the weight of each knot. The overall effect is to select proper number of knots to be used and, at the same time, relocate these knots to find a suitable distribution. And as the algorithm converges, the final result uses many fewer knots than pixels to represent the object while keeping the error of estimate within certain tolerance. A 2D cross-well tomography example will be provided to test the algorithm. The simulation results show that the B-Spline method reduces the dimensional complexity by 90%, compared with that of the pixel-based method, and decreases time complexity by 50%, without significantly degrading the estimation.

1.1 Contribution

In this thesis, we propose using B-Splines to represent the object to be reconstructed. The goal is to decrease the large complexity brought by the edge-preserving regularizer, and adaptively find the solution. Our contributions are mainly in the following aspects:

- We make a thorough analysis of the edge-preserving regularizer, especially the CRB(Cramér-Rao Bound) analysis. The result is very interesting. The edge-preserving regularizer does preserve edges better compared with Tikhonov regularization. As to the variance of estimate, however, the edge-preserving regularizer performs worse than Tikhonov one. This partly reflects the difference

between the two regularization methods. Tikhonov regularization add equally constraint on all areas, while an edge-preserving regularizer puts more effort on locating the edges. The price is that the variance of estimate is increased.

- We use B-Splines to represent the object and remodel the inverse problem. By finding the optimum knot distribution, the result has a significant decreasing in dimension. More efforts are made on how to adaptively find the optimum knot distribution. A knot refining and pruning algorithm is developed to automatically find the edges as well as select proper B-Splines.

- For knot refining, a curvature based knot insertion method is developed to intelligently insert knots. The intuition is that the optimum knot distribution should reflect the underlying smoothness structure of the object. Specifically, more knots should cluster at regions with large curvatures, while fewer knots are needed for flat regions. Therefore, simply doubling the number of knots will bring overhead for the insertion, i.e. most knots inserted in flat regions will have to be deleted later. Assuming a current coarse estimate is a good hint for further refined structures, we can use the curvature information as a guide to only insert new knots in areas of potential edges. By doing so, the knots inserted will be mostly useful and need not to delete them later. In the 2D case, the idea can be further refined by using principle directions used in computing curvatures. The principle directions are directions where the minimal and maximal curvature lie on a surface. If at a point, the large curvature only happens along x dimension, and in the y dimension, the curvature is very small, we will only want to increase resolution along the x dimension. By using the principle directions, we will be able to determine whether to insert knot in one dimension or both.

- For knot pruning, we should find which knots need to be deleted. A greedy

search algorithm is developed using the estimate of the weight of each knot. The weight of each knot is measured as the change in reconstruction when deleting the knot. The more the change, the larger the weight. Knots with small weight are considered redundant, i.e. they were previously placed in regions where enough knots had already been assigned. With the weight information, we will be able to sort knots in the order of their importance and delete them one at a time until the error of resulting reconstruction (computed by the cost function of the edge-preserving regularizer) exceeds a certain tolerance. We also notice the vicinity effects of each knot, i.e. deleting a knot changes the actual weight of adjacent knots. An efficient solution is to avoid deleting adjacent knots.

1.2 Organization

This thesis is organized into five chapters; the main body is Chapter 2, 3 and 4. Chapter 2 introduces related background of the problem, including linear inverse problem and Tikhonov regularization, B-Splines basis and geometric computations. Chapter 3 investigates the edge-preserving regularization method and makes some analysis. Chapter 4 presents the developments of the adaptive B-Splines methods incorporated with edge-preserving regularizer solving the image reconstruction problem.

Chapter 2 introduces some related background knowledge of this thesis. Firstly, the cross-well tomography problem is set up. The electromagnetic inverse scattering theory is introduced to solve the problem, and the Born approximation is used to get the linear inverse problem model. Secondly, the Tikhonov regularization method is introduced to solve the problem. The idea of edge-preserving regularization is proposed after analyzing the smoothing effects of Tikhonov regularization. Thirdly, we introduce the basics of B-Spline basis. With the good properties it possesses, we

propose using it to decrease the complexity of the problem. Lastly, some advanced geometric computation of B-Splines are introduced. The most important is computing the curvature of B-Spline surfaces, which will be helpful to our knot refining algorithm developed in Chapter 4.

Chapter 3 analyzes the deficiency of Tikhonov regularization. It then introduces the edge-preserving regularizer using non-quadratic regularization functions. The common properties of regularization functions are introduced. Furthermore, a CRB analysis is provided to examine the estimation properties of the edge-preserving regularizer. Finally, we apply the edge-preserving regularizer to solve the cross-well tomography problem.

Chapter 4 first points out the large complexity the edge-preserving regularization brings, then B-Spline basis is proposed to be used. By representing the object by B-Splines, the inverse problem is remodeled. It is shown that if the B-Splines are carefully selected, we can obtain a good reconstruction while significantly decreasing the dimension of the problem. An adaptive knot refining and pruning algorithm is developed to automatically find a suitable knot distribution. The knot refining part involves knot insertion under the guidance of curvature information; the knot pruning part involves knot deletion based on the estimate of weight of each knot. As the algorithm converges, which is generally within 10 iterations, the result will be the fewest possible number of knots that can be used to make an estimate within a certain error. Also, the 2D example used in Chapter 3 is revisited here to illustrate the algorithm.

Chapter 5 discusses some conclusions of this thesis and some possible ideas for the future work.

Chapter 2

Background

2.1 Cross-well Tomography Problem

In this section, we will set up the cross-well tomography problem which is the motivation of this thesis. We will revisit the problem as an example later in the following chapters to examine our approach.

2.1.1 Problem Setup

The problem comes from the application of finding underground pollution plumes. It is proposed to use a cross-well tomography method: one array of electromagnetic transmitters and one array of receivers are placed in two wells along the boundaries of the region of interest(Fig. 2.1). By measuring the scattered electromagnetic waves at receivers, we can determine the physical properties, such as conductivity, permittivity and permeability, of the medium in the region. Since the pollution plumes will have different electromagnetic properties than common soils, we will be able to determine the existence and shape of the plumes.

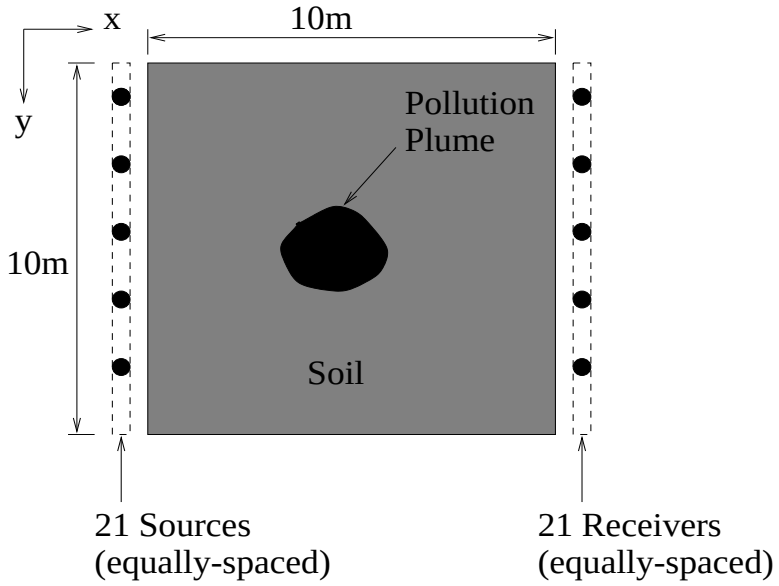


Figure 2.1: Cross-well tomography problem.

For the work in this thesis, as shown in Fig. 2.1, 21 equally-spaced dipole transmitters and receivers are placed along the boundaries respectively. The frequency of electromagnetic transmitted is $\omega = 5 \times 10^7 Hz$. To represent the electromagnetic properties of the region, we use typical settings for wet soils, where electric permittivity $\varepsilon = 20$, magnetic permeability $\mu = \mu_0$ is the permeability of free space, and electric conductivity $\sigma = 0.01 s/m$.

2.1.2 Electromagnetic Inverse Scattering

Electromagnetic scattering theory has played an important role in many applications, which are concerned with the effect an inhomogeneous medium has on an incident wave. More specifically, it involves two aspects of the problem, one is *direct scattering problem*, which is to determine the scattered field from the incident field; another is *inverse scattering problem* we will be focusing on, which is to determine the

nature of the inhomogeneity from the knowledge of the scattered field. Lots of work has been done in this region and many different approaches exist [8] [9] [10]. The inverse scattering problem is important in cases where details about the structure or composition of an object are required but can not be measured directly, instead, the measurements are often taken remotely without affecting the object. Thus, inverse scattering has been applied in many areas such as radar/sonar, medical diagnostics, and nondeconstructive testing [11] [12] [13].

The cross-well tomography problem we have set up can be shown, from the wave scattering point of view, in Fig. 2.2.

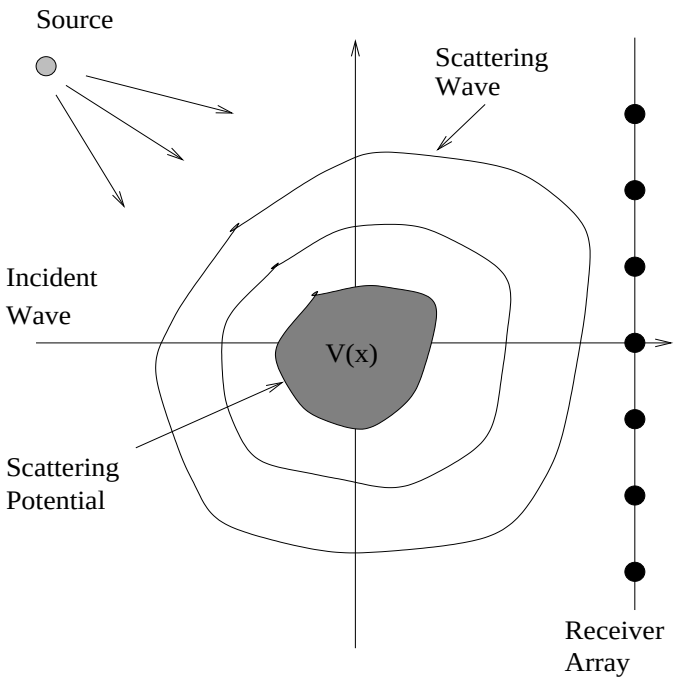


Figure 2.2: Cross-well tomography problem.

where we consider a finite scattering region mathematically described by a scattering potential $V(x)$ embedded in a homogeneous medium of permittivity ε_1. Thus the distribution of permittivity through the scatterer can be written as:

$$\varepsilon(\mathbf{x}) = \varepsilon_1 + \triangle\varepsilon(\mathbf{x}) \tag{2.1}$$

where $\triangle\varepsilon(\mathbf{x}) = 0$ outside the scatterer. With k^2 represents the wave number, we have the non-homogeneous Helmholtz equation [8]:

$$\nabla^2\Psi + k^2(\varepsilon_1 + \triangle\varepsilon(\mathbf{x}))\Psi = 0 \tag{2.2}$$

where θ is the incident angle of the plane wave and ∇^2 is the square of the gradient operator:

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \tag{2.3}$$

Now we introduce the Green's function which will translate equation (2.2) into an integral form. The Green's function $g(\mathbf{x} \mid \mathbf{x}')$ is defined as the solution to the following equation [14]:

$$\nabla^2 g(\mathbf{x}, \mathbf{x}') + \mathbf{k^2}\varepsilon_\mathbf{1}\mathbf{g}(\mathbf{x}, \mathbf{x}') = -\delta(\mathbf{x} - \mathbf{x}') \tag{2.4}$$

where $\mathbf{x}'$ is the coordinate of source and $\mathbf{x}$ is the position to be measured with respect to $\mathbf{x}'$. And it has been proved [8] if the Green's function satisfies certain boundary conditions, it possesses an important property:

$$g(\mathbf{x}', \mathbf{x}'') = \mathbf{g}(\mathbf{x}'', \mathbf{x}') \tag{2.5}$$

Thus, equation (2.2) can be rewritten as:

$$\begin{cases} \Psi(\mathbf{x}) = \Psi^{inc}(\mathbf{x}) + \int k^2 \triangle\varepsilon(\mathbf{x}')g(\mathbf{x}, \mathbf{x}')\Psi(\mathbf{x}')d\mathbf{x}' & \text{(a)} \\ \Psi^{scat}(\mathbf{x}) = k^2 \int g(\mathbf{x}, \mathbf{x}')\triangle\varepsilon(\mathbf{x}')\Psi(\mathbf{x}')d\mathbf{x}' = \text{data} & \text{(b)} \end{cases} \tag{2.6}$$

The above equation can be solved provided all the boundary conditions are satisfied by both $\Psi(\mathbf{x}')$ and $g(\mathbf{x}, \mathbf{x}')$ are given. And the field Ψ here is actually the combination of the local incident field Ψ^{inc} and Ψ^{scat}, i.e.

$$\Psi = \Psi^{inc} + \Psi^{scat} \tag{2.7}$$

2.1.3 Born Approximation

In practice the integral equations in (2.6) bring into a non-linear structure, because Ψ in (2.6)(b) depends on $\triangle\varepsilon$ in (2.6)(a). It usually does not have close form solution. Thus, some approximate solution should be considered, among which the Born approximation [15] [16] has become the prevalent technique because of its easy implementation and simple physical interpretation. It replaces $\Psi(\mathbf{x})$ in (2.6)(b) with $\Psi^{inc}(\mathbf{x})$ which yields to:

$$\Psi^{scat}(\mathbf{x}) = \int_{\tau'} k^2 g(\mathbf{x}, \mathbf{x}') \triangle\varepsilon(\mathbf{x}') \mathbf{\Psi^{inc}}(\mathbf{x}') \mathbf{d}\tau' \tag{2.8}$$

The approximation has been proved to be good if $\triangle\varepsilon(x)$ is small. So the Born approximation results in the form of a first-kind Fredholm integral equation, which can be discretized furthermore to a finite dimension matrix problem of the form [5]:

$$y = Af + n \tag{2.9}$$

where each element of vector y is the scattered field observations obtained along the receiver array; A is a matrix associated with the integral kernel; f corresponds to the conductivity perturbation and n represents additive noise.

Therefore, the cross-well tomography problem has been translated into the form of a typical linear inverse model. Further details of such model will be discussed in the following section.

2.2 Regularization of Linear Inverse Problems

In this section, we will investigate the ill-posed property of discretized linear inverse problems and the energy-based regularization method as a solution. Based on these discussions, we will be able to address the edge-preserving regularization techniques in Chapter 5.

2.2.1 Linear Inverse Problems

Generally, we can describe a linear inverse problem as a linear process with a certain distortion model which has the following form:

$$y = Af + n, \tag{2.10}$$

where A is a known degradation process; n is additive, Gaussion noise; y is the degraded signal. Our problem is to obtain an estimate of the real object f, which is usually mapping of some physical properties, from the data vector y.

2.2.2 Nature of Ill-posed Problems

A first and obvious solution to the equation (2.10) we established above would be least squares solution:

$$f_{LSQ} = \left[A^T A\right]^{-1} A^T y = \left[A^T A\right]^{-1} AA^T f + \left[A^T A\right]^{-1} A^T n \tag{2.11}$$

but for practical problems, the above approach does not make sense, i.e. f_{LSQ} tends to be contaminated by large amplitude high-frequency artifacts, and this is much different from the true object. Those non-physical artifacts of the solution f_{LSQ} comes from the ill-conditioned nature of degradation kernel A, i.e. the condition

number of A is always very large in practice which means the columns of A are nearly linearly dependent. Though we can replace A by a well-conditioned matrix derived from A, we still can not get a useful solution. In another word, we need to filter out the high-frequency artifacts to get useful solution. While in practice, generally A is a degradation process which is a kind of lowpass process, thus $\left[A^T A\right]^{-1}$ is a highpass filter which would only amplify the high-frequency noise in the data.

2.2.3 SVD Analysis

To gain in-depth understanding, we will analyze the ill-posed problems and regularization methods from the SVD (Singular Value Decomposition) point of view.

The Singular Value Decomposition of A is as the following:

$$A = U\Sigma V^T = \sum_{i=1}^{n} u_i \sigma_i v_i^T \tag{2.12}$$

where $\Sigma = diag(\sigma_1, ...\sigma_n), \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$

Thus, the condition number of A is equal to σ_1/σ_n. For ill-posed problems, the singular values σ_i typically decay gradually to zero, and u_i and v_i tend to have more sign changes as σ_i decreases. This shows the nature of ill-posed problem in two aspects: 1. In ill-posed problems, A is highly ill-conditioned, i.e. A has large condition number σ_1/σ_n. The vectors v_i associated with the small (close to zero) σ_i are numerical null-vectors of A and have many sign changes. 2. Af is a smooth process of f and the inverse problem amplifies the high-frequency oscillations in the right-hand side y. It is difficult to get an acceptable solution, because small errors in data may cause large errors in the solution. A simple 1D example is shown in Fig. 2.3.

We can also draw the same conclusion from the following mathematical analysis:

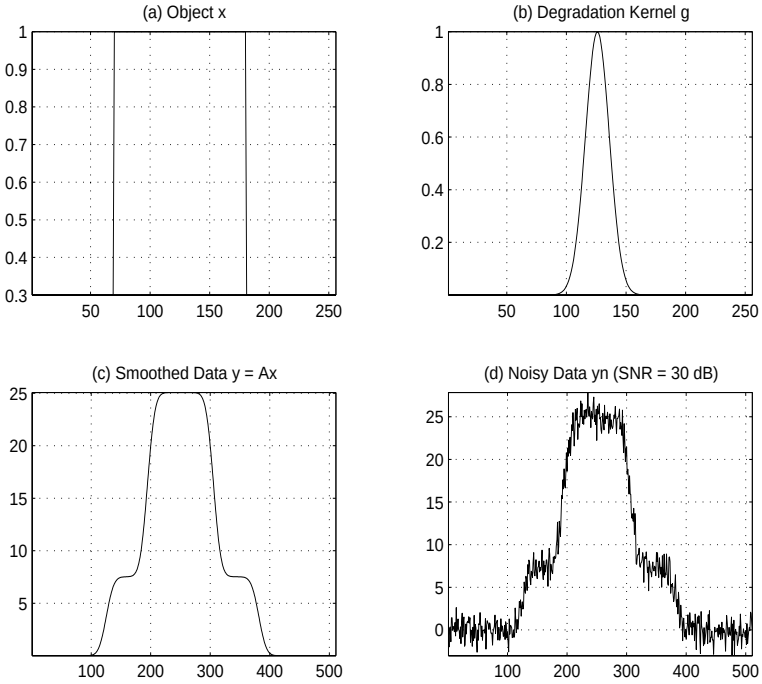


Figure 2.3: A simple 1-d inverse problem, with a Gaussian function as the kernel.

Consider linear least-squares problems:

$$\min_f \|Af - y\|_2, A \in \mathcal{R}^{m \times n} m > n \tag{2.13}$$

from the SVD of A in equation (2.12), the least-squares solution will be:

$$f_{LSQ} = \sum_{i=1}^{n} \frac{u_i^T y}{\sigma_i} v_i \tag{2.14}$$

Apparently, since u_i, v_i has many sign changes with σ_i degrades to zero, the solution f_{LSQ} has many large components of high-frequencies. Therefore, f_{LSQ} appears to be random and is not acceptable.

From the above analysis, the idea of regularization can be naturally induced.

Specifically, the purpose of regularization would be to filter out the contributions of the small singular values to the solution. One well-known regularization method is Tikhonov regularization [21]. The idea is to define the regularized solution f_λ as the minimizer of the following combination of the residual norm and the side constraint:

$$f_{Tik} = \underset{f}{argmin} \|Af - y\|_2^2 + \lambda^2 \|L(f - f^*)\|_2^2, \tag{2.15}$$

where the *regularization parameter* λ controls the weight of compromise of fidelity of solution to the residual norm and to the side constraint norm. Generally, if $L = I_n$ and $f^* = 0$,

$$f_{Tik} = \sum_{i=1}^{n} g_i \frac{u_i^T y}{\sigma_i} v_i \tag{2.16}$$

where, $g_i = \sigma_i^2/(\sigma_i^2 + \lambda^2)$. Hence the filter factor g_i counters the contributions of small σ_i.

2.2.4 L-Curve

In 2.15, the regularization parameter λ is critical and need to be wisely chosen. L-curve is an convenient way to find proper λ [23]. Given 2.15, we can draw out a L-curve as shown in Fig. 2.4.

The vertical part of the L-curve corresponds to solutions where $||Lf_{Tik}||_2$ is very sensitive to changes in the regularization parameter λ. The horizontal part of the L-curve corresponds to solutions where it is the residual norm $||Af_{Tik} - b||_2$ that is most sensitive to the regularization parameter λ. Thus, we can find the optimal regularization parameter by locating the corner of this L-curve. This is the general procedure of solving the Tikhonov regularization problem. Using the previous 1D example, some results corresponding to different regularization parameters are shown in Fig. 2.5.

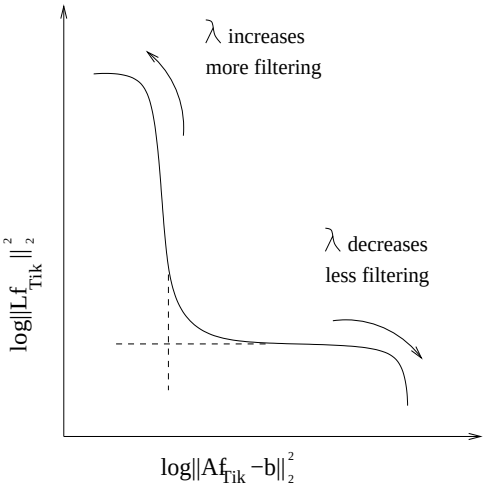


Figure 2.4: Illustration of L-curve

2.2.5 General Regularizations

In general, regularization method can be written as the following:

$$f_{reg} = \underset{f}{argmin} \|Af - y\|_2^2 + \lambda\Omega(f) \tag{2.17}$$

In Tikhonov regularization, $\Omega(f)$ is of the form of 2-norm or derivative of it. This quadratic regularization function adds large constraint on large frequency components of the solution, thus leads to a 'smooth' solution. Therefore, Tikhonov regularization is not good for reconstruction of objects with sharp edges which are critical to describe the shape and internal structures of the objects for our cross-well problem. Since in practice sharp edges of an object usually have very large frequency components, it is natural to think of applying non-quadratic regularization functions which can smooth out relatively small perturbations while preserving larger ones, thus preserve edges. In Chapter 5, we will discuss the edge-preserving regularization in details.

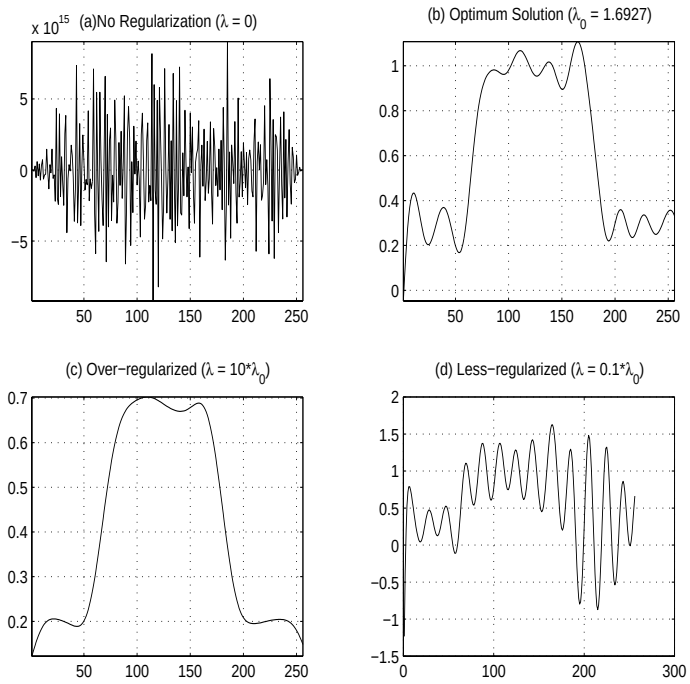


Figure 2.5: Tikhonov regularization effects for the 1-d problem.

2.3 Introduction to B-Spline Functions

In this section, we will introduce some background material about B-Spline functions. First, we will define the B-Spline basis to represent a certain function or object. Then we will talk about how the knot distribution affect the properties of the representation, which is the motivation of using B-Splines in this problem. Also we will expand it to 2D case by introducing tensor-product B-Splines to fit in our application. Lastly, we will discuss the geometries of B-Splines, which are required in calculating the differentials and curvatures.

2.3.1 B-Spline Basics

B-Splines are widely used in many aspects of numerical analysis (statistical data interpolation, data smoothing, numerical solution of inverse problem with integral model, computer aided geometric design, etc. [32] [33] [34]). There exists a set of well-developed theories on B-Splines, their properties and applications, including some efficient algorithms for computation using B-Splines [36] [37] [38] [39]. Also, B-Splines are a strong candidate for use in our inverse problem, especially considering their local support and smoothness property which we will discuss later.

Definition 1 [32] A function $s(x)$, defined on a finite interval $[a, b]$, is called a *spline function* of *degree* $k > 0$ (*order* $k + 1$), having *knots* as the strictly increasing sequence $\lambda_j, j = 0, 1, ..., n(\lambda_0 = a, \lambda_n = b)$, if the following conditions are satisfied:

1. $s(x) \in \mathcal{P}_k$, for $x \in [\lambda_j, \lambda_{j+1}]$, $j = 0, 1, ..., n$.
2. $s(x) \in \mathcal{C}^{k-1}[a, b]$.

where $\mathcal{P}_k$ is the set of all polynomial functions of k orders, $\mathcal{C}^{k-1}$ is the set of all $k-1$ continuous functions.

Definition 2 The *B-Spline* $N_{i,k+1}$ of degree k with knots $\lambda_i, ..., \lambda_{i+k+1}$ is defined as :

$$N_{i,k+1}(x) = (\lambda_{i+k+1} - \lambda_i) \sum_{j=0}^{k+1} \frac{(\lambda_{i+j} - x)_+^k}{\prod_{l=0, l\neq j}^{k+1} \lambda_{i+j} - \lambda_{i+l}} \tag{2.18}$$

where

$$(\lambda - x)_+^k = \begin{cases} (\lambda - x)^k, & \text{if } \lambda > x; \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

Based on the above definition, B-Splines have the following important properties:

1. *Positivity and Local Support :*

$$\begin{aligned} &N_{i,k+1}(x) \geq 0 \text{ for all } x. \\ &N_{i,k+1}(x) = 0 \text{ if } x \notin [\lambda_i, \lambda_{i+k+1}]. \end{aligned} \tag{2.20}$$

2. *Boundary values :*

$$N_{i,k+1}^{(l)}(\lambda_i) = N_{i,k+1}^{(l)}(\lambda_{i+k+1}) = 0, l = 0, ..., k-1. \tag{2.21}$$

where $(\cdot)^{(l)}$ stands for the l order derivative of the function.

3. *Minimal support :* If a piecewise polynomial with the same smoothness property over the same knot vector has less support than N_i^{k+1}, it must be the zero function.

4. *Linear Independence :*

$$\sum_{i=0}^{n-1} c_i N_{i,k+1}(x) = 0 \text{ implies } c_i = 0 \text{ for all } i. \tag{2.22}$$

Some simple examples of B-Splines are shown in Fig. 2.6

The *B-Splines* form a *basis* for the space consist of piecewise polynomial functions; every piecewise polynomial function $s(x)$ over $[a, b]$ has a unique representation:

$$s(x) = \sum_{i=-k}^{n-1} c_i N_{i,k+1}(x), \tag{2.23}$$

where $N_{i,k+1}(x)$ for $i = -k, ..., -1$ and $i = n-k, ...n-1$ are generated as before by introducing 'arbitrary' additional knots $\lambda_{-k} \leq \lambda_{-k+1} \leq ... \leq \lambda_0 = a$, and $b = \lambda_n \leq \lambda_{n+1} \leq ... \leq \lambda_{n+k}$. The reason of using these additional knots is that each B-Spline has support on several adjacent knots (As shown in Fig. 2.6). Thus, the B-Splines defined on these additional knots also have support on the knots at or near the ends. Different choices of these additional knots will bring different ending effects on the curves/surfaces[32].

As we shall see in the next chapter, these basic properties make B-Splines a good

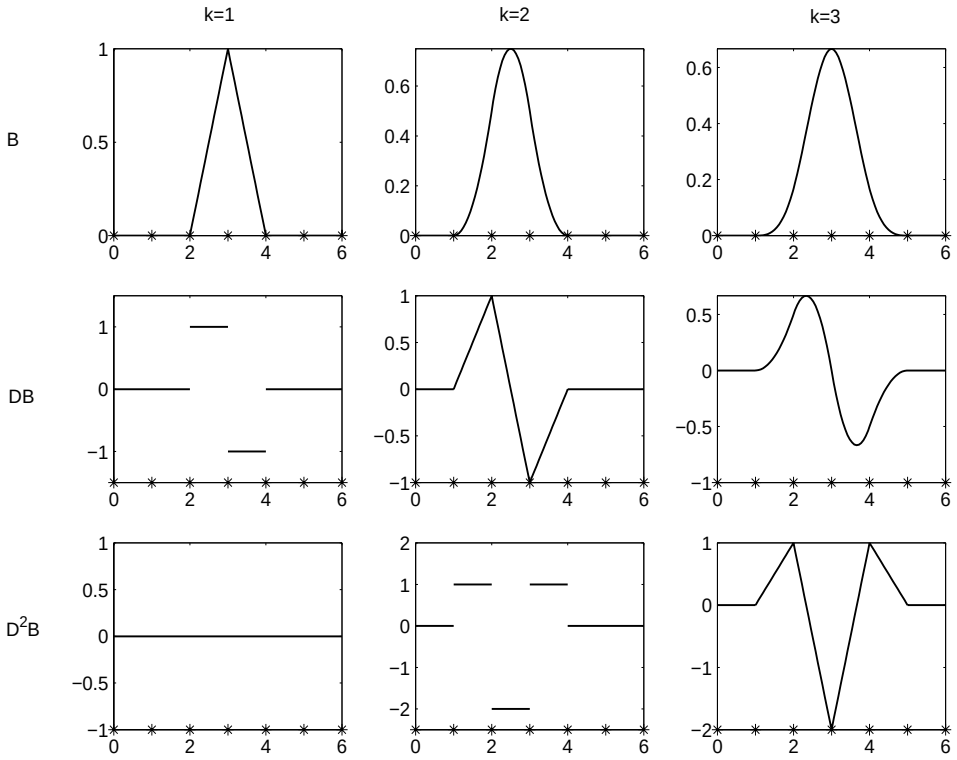


Figure 2.6: B-Splines $N_{i,k+1}$ with different order $(k+1)$ and their first and second derivatives. ($i = 3$, knots sequence $\lambda = [0123456]$)

candidate to be used in our inverse problem to obtain multiscale solutions. Furthermore, there are efficient algorithms [36] for evaluating the values and derivatives of B-Splines.

To fit into our 2D problem, we need to define Bivariate B-Splines. There are different ways in extending the univariate splines to 2D case. *Tensor product B-Splines* are the most widely used due to their simplicity.

Definition 3 Having strictly increasing sequences :

$$\begin{aligned} a = \lambda_0 < \lambda_1 < ... < \lambda_m = b, \\ c = \mu_0 < \mu_1 < ... < \mu_n = d, \end{aligned} \tag{2.24}$$

The function $s(x, y)$ on $R = [a, b] \times [c, d]$, of degree $k > 0$ in x and $l > 0$ in y, with knots $\lambda_i, i = 0, 1, ..., m$ in the x-direction and knots $\mu_j, j = 0, 1, ..., n$ in the y-direction, is called a *Tensor Product Spline*, if the following conditions are satisfied:

$$\begin{aligned} &1.\ s(\lambda_i, \mu_j) \in \mathcal{P}_k \otimes \mathcal{P}_l, i = 0, 1, ..., m; j = 0, 1, ..., n. \\ &2.\ \frac{\partial^{i+j} s(x, y)}{\partial x^i y^j} \in \mathcal{C}(R), i = 0, 1, ..., k-1; j = 0, 1, ..., l-1. \end{aligned} \tag{2.25}$$

Similarly, let the $N_{i,k+1}$ and $M_{j,l+1}$ are the normalized univariate B-Splines defined as before, on the knot sequences λ and μ respectively, we can uniquely represent every spline $s(x, y) \in (\lambda_0, ..., \lambda_m; \mu_0, ..., \mu_n)$ as the following :

$$s(x, y) = \sum_{i=-k}^{m-1} \sum_{j=-l}^{n-1} c_{i,j} N_{i,k+1}(x) M_{j,l+1}(y), \tag{2.26}$$

Some examples of tensor product B-Splines are shown in Fig. 2.7.

Tensor product B-Splines retain all the good properties of univariate B-Splines. Especially, since the basis corresponding to x-direction and y-direction are separable, it can be easily handled and the efficient algorithms in 1D case can be directly employed. A certain drawback would be that only rectangular approximation patches can be applied thus the refinement procedure are tied to existing rectangular grid. There are other approaches using triangular patches which bring more freedom and adaptation [49] [50]. We will leave this effort to future work.

2.3.2 Knots Distribution and Smoothness

As stated before, given a certain knot sequence, we can uniquely define a set of B-Spline basis which are piecewise polynomials of degree k. We can combine the B-Spline curves/surfaces to approximate a wanted curve/surface. It can be proved [35]

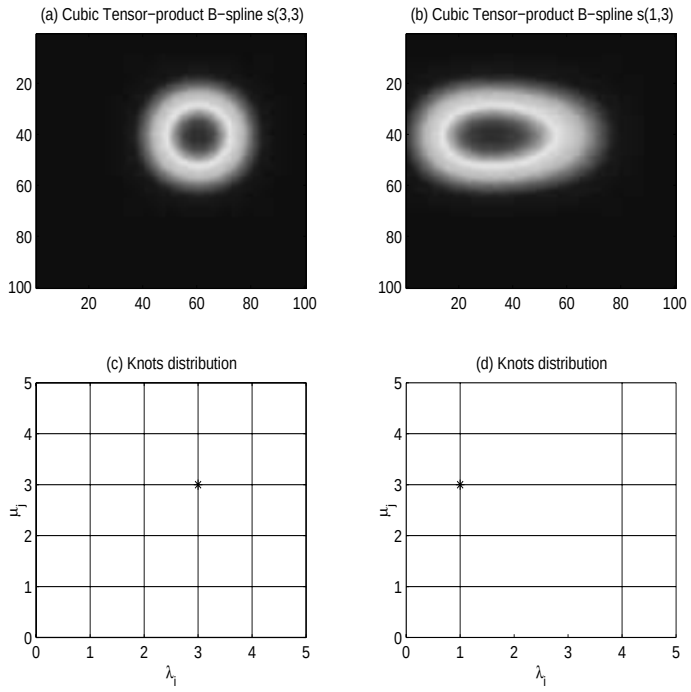


Figure 2.7: Examples of tensor product B-Splines and corresponding knots distribution.

that if sufficiently many knots are inserted into the knot sequence, the resulted approximation will be arbitrarily close to the curve/surface. Thus instead of describing the curve/surface directly, a set of knots would be sufficient. This is the whole basis of using B-Splines to approximate curves/surfaces. In practice, we are interested in finding the 'best' approximation which uses as few knots as possible within a certain error tolerance. A simple example is given in Fig. 2.8.

The sample data are Titanium Heat data which gives a certain property of titanium as a function of temperature. It has been used extensively as test data for Spline fitting. As the number of knots increases, the resulted Spline curves are closer to real data in the means of Mean Square Error(MSE). In 2.8(d), 12 non-equally-spaced and well-selected knots are used which leads to a less MSE than that of Fig. 2.8(c) where

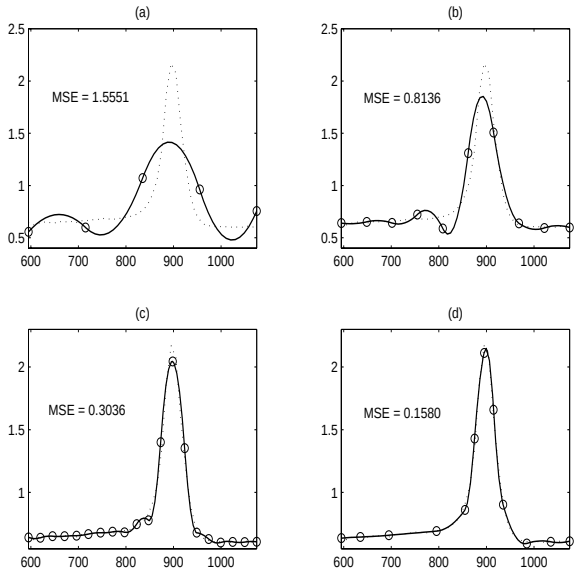


Figure 2.8: Illustration of using cubic B-Splines to approximate Titanium Heat data. (a)5 equally-spaced knots; (b)10 equally- spaced knots; (c)20 equally-spaced knots. (d)12 selected knots.

20 equally-spaced knots were used. This is a very interesting and helpful property of B-Splines approximation and is the motivation of using B-Splines in our adaptive reconstruction approach to solve the inverse problems.

Another important property of B-Splines is their local support. It allows the easy alteration of a complex curve/surface in one region without affecting the remote portion of the curve. Since each B-Spline is defined over a certain knot sequence $\lambda_0, ..., \lambda_n$, it is natural to see that if the knots are set apart, the B-Spline support will be large, consequently we will get relatively flat basis. On the contrary, if we set the knots to be very near to each other, the B-Spline will have very narrow support which lead to more local details (Fig. 2.9). Since the knots alteration will only locally affect the approximated function and remote part of the curve will remain the same, it is natural to assign a large density of knots to areas with significant fine scale variations.

Only a 'sparse' distribution of knots are needed to well approximate the flat part of the curve.

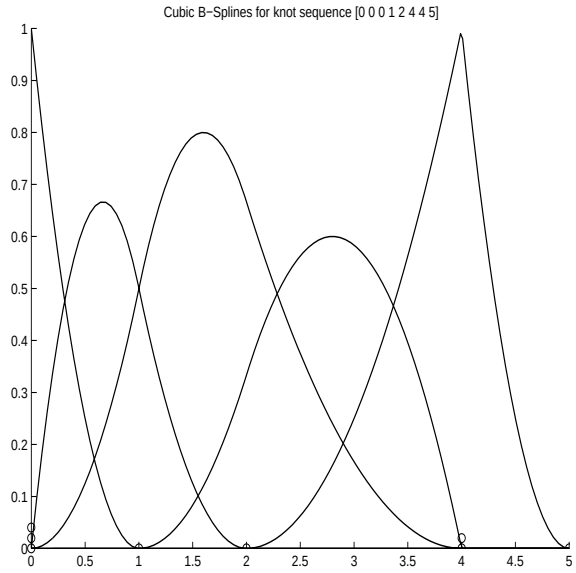


Figure 2.9: Distant knots leads to B-Spline of large support (flat) and vice versa; Certain multiple knots leads to discontinuity.

In Fig. 2.9, we also note that to construct cubic B-Splines, additional knots have been added at end points. At one end, we use multiple knots ([0 0 0]), which brings high-order discontinuity there; at the other end, we extend knots out of the interested region, i.e. add one knot at position 5, which makes the reconstruction at that end point more smooth. But the shortcoming is that we will have leakage outside the focused region.

As stated before, we are interested in efficiently representing a surface, i.e. a function of two variables. Our approach will be to use B-Splines with the fewest possible knots to best approximate the surface within a certain error tolerance. In this case, the approximation will essentially be adapted to the smoothness of the surface. Thus, the problem can be stated as to find a suitable set of knots based on

which the B-Splines can approximate the surface efficiently.

To find an adaptive procedure of achieving this goal, we will need to reallocate the knots to better reflect the underlying smoothness of the curve/surface. Curvature information will be a good hint to this further redistribution of knots. Specifically, if the knots is equally spaced, then, given a certain error tolerance, we will need to insert more knots to the regions of large curvatures and delete redundant knots at regions of small curvatures. The computation of knot insertion and deletion have been thoroughly discussed in many papers [37] [38] [39]. And in the following section we will focus on the geometries needed to compute spline curvature.

2.4 Geometries of Splines

As stated before, we will need to measure the smoothness of a reconstructed B-Spline curve/surface, then use it as a guide in further refinement. Specifically, we want to redistribute the knots in such a way that large density of knots are assigned to regions of large curvatures, and fewer knots to relative flat regions. In this section, we will introduce the curvature computing of B-Spline curves/ surfaces.

2.4.1 Curvature of Space Curves

Before discussing surface curvature, we introduce the geometric properties of curves.

In general, it is helpful to represent a space curve (in $\mathcal{R}^3$) by the following parametric form:

$$\mathbf{x} = \mathbf{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, t \in [a, b] \subset \mathcal{R} \tag{2.27}$$

Furthermore, we can reparametrize the curve by:

$$s(t) = \int_{t_0}^{t} ds = \int_{t_0}^{t} (\dot{x}^2 + \dot{y}^2 + \dot{z}^2)^{1/2} dt = \int_{t_0}^{t} \|\dot{\mathbf{x}}(t)\| dt \tag{2.28}$$

where the dots denote derivatives with respect to t, and s is called *arc length* reparameterization of the curve [44](an example is shown in Fig. 2.10). From 2.28, we have $\dot{s}(t) = \|\dot{\mathbf{x}}(t)\|$, which is called *arc element* of the curve. Furthermore, it indicates that the variation rate of arc length with respect to curve parameter is the speed of the curve parameterization. For B-spline curves, where the parameterization is regular, i.e. $\|\dot{\mathbf{x}}(t)\| \neq 0$ [44], we may write $dt/ds = 1/\|\dot{\mathbf{x}}(t)\|$. Let $\alpha(s) = \mathbf{x}(t(s))$ be the arc length parameterized curve, we have:

$$\|\frac{d\alpha}{ds}\| = \|\frac{d}{dt}(\mathbf{x}(t(s)))\frac{dt}{ds}\| = \|\dot{\mathbf{x}}(t(s))\frac{1}{\|\dot{\mathbf{x}}(t(s))\|}\| = 1 \tag{2.29}$$

Hence, arc length reparameterization has unit speed for regular curves.

Now, we introduce a local orthonormal coordinate system, *the Frenet Frame* $\mathbf{t}, \mathbf{m}, \mathbf{b}$[44], which is very helpful to provide the local behavior of curves. By observing the changes of the Frenet Frame as a function of parameter t, we can get the curvature information of space curves. The Frenet Frame is the frame formed by *tangent vector* $\mathbf{t}$, *main normal vector* $\mathbf{m}$ and *binormal vector* $\mathbf{b}$, where :

$$\mathbf{t} = \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}, \quad \mathbf{b} = \frac{\dot{\mathbf{x}} \times \ddot{\mathbf{x}}}{\|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\|}, \quad \mathbf{m} = \mathbf{b} \times \mathbf{t}. \tag{2.30}$$

Given the Frenet Frame, we can construct the *osculating circle* [45] which lies in the osculation plane formed by $\mathbf{t}$ and $\mathbf{m}$, and has the same first and second derivative vectors as the curve. The inverse of the osculating circle radius ρ is called *curvature* κ.

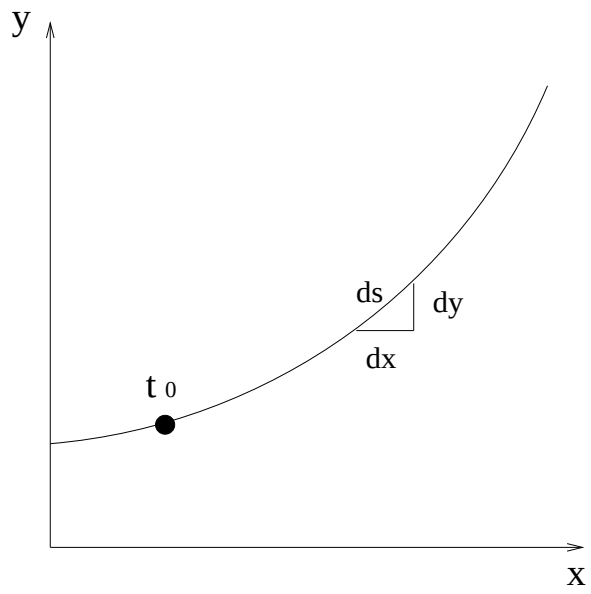


Figure 2.10: Arc length reparameterize of curve $\mathbf{x}(t)$

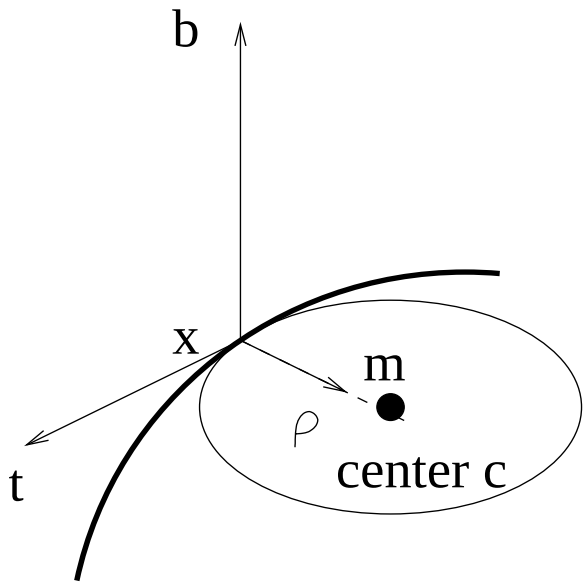


Figure 2.11: Franet Frame and Osculating Circle

By applying the arc length reparametrization, we have the following results:

$$\kappa = \kappa(s) = \|\mathbf{x}''\| \tag{2.31}$$

where the prime denotes derivative with respect to arc length s. In terms of the actual parameter t, we have:

$$\kappa = \kappa(t) = \frac{1}{\rho} = \frac{\|\dot{\mathbf{x}} \times \ddot{\mathbf{x}}\|}{\|\dot{\mathbf{x}}\|^3} \tag{2.32}$$

The geometric interpretation of curvature is straightforward: consider the angle $\triangle\theta$ between two tangent vectors $\mathbf{t}$ and $\mathbf{t}(s + \triangle s)$, thus $\kappa = d\theta/ds$.

2.4.2 Curvature of Surface Curves

The above discussion can be extended to surfaces. Assume a regular parametric surface:

$$\mathbf{x} = \mathbf{x}(\mathbf{u}) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} \; ; \; \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} \in [\mathbf{a}, \mathbf{b}] \subset \mathcal{R}^{\mathbf{2}} \tag{2.33}$$

with

$$\mathbf{x}_u \times \mathbf{x}_v \neq 0 \text{ for } \mathbf{u} \in [\mathbf{a}, \mathbf{b}] \tag{2.34}$$

Consider a regular curve $\mathbf{x}(\mathbf{u}(t))$ on the surface, the squared arc element (*first fundamental form*) is defined as the following [44]:

$$ds^2 = Edu^2 + 2Fdudv + Gdv^2 \tag{2.35}$$

where

$$E = \mathbf{x}_u \cdot \mathbf{x}_u, \; F = \mathbf{x}_u \cdot \mathbf{x}_v, \; G = \mathbf{x}_v \cdot \mathbf{v}_v \tag{2.36}$$

Analogous to the space curves, we can also define a frame $\mathbf{x}_u, \mathbf{x}_v, \mathbf{n}$ for the surface. The partial $\mathbf{x}_u$ and $\mathbf{x}_v$ at a point $\mathbf{x}$ span the tangent plane to the surface. The normalized normal $\mathbf{n}$ [45] to the surface is:

$$\mathbf{n} = \frac{\mathbf{x_u} \times \mathbf{x_v}}{\|\mathbf{x_u} \times \mathbf{x_v}\|} \tag{2.37}$$

Let $\mathbf{u}(t)$ be a curve on the surface $\mathbf{x}(\mathbf{u})$ in the direction of $\mathbf{t}$ as defined in 2.30. In 2D case, consider $\mathbf{x}_u = d\mathbf{x}/du, \mathbf{x}_v = d\mathbf{x}/dv$, the direction of $\mathbf{t}$ can be represented by $\lambda = dv/du$) (Fig. 2.12), applying the above definition of space curve curvature to the surface curve, the normal curvature at point $\mathbf{x}$ is given by:

$$\kappa(\lambda) = \frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2} \tag{2.38}$$

where

$$L = \mathbf{n} \cdot \mathbf{x}_{uu},\ M = \mathbf{n} \cdot \mathbf{x}_{uv},\ N = \mathbf{n} \cdot \mathbf{x}_{vv} \tag{2.39}$$

Generally, κ always changes as λ changes, thus for a given point $\mathbf{x}$ on the surface, the normal curvatures corresponding to different surface curves which pass the same point $\mathbf{x}$ are different. This result provides useful information on the smoothness of surface along different directions.

2.4.3 Gaussian and Absolute Curvature

The definition of normal curvature of a surface, $\kappa(\lambda)$ is of rational quadratic form. Thus the extreme values κ_1 and κ_2 of $\kappa = \kappa(\lambda)$, which are called *principal curvatures* of the surface, are the roots of

$$det \begin{bmatrix} \kappa E - L & \kappa F - M \\ \kappa F - M & \kappa G - N \end{bmatrix} = 0 \tag{2.40}$$

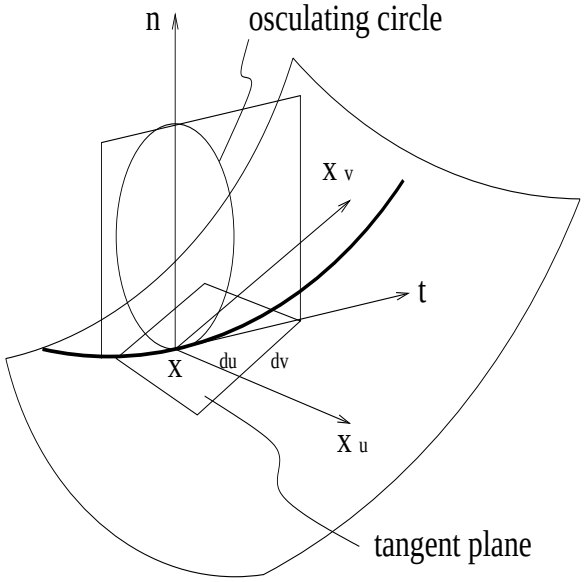


Figure 2.12: Frame for surface

And the corresponding quantities λ_1 and λ_2 define the *principal directions*, which are the surface curve directions in the tangent plane.

Having the principal curvatures, we have two important definitions of the surface curvatures on point $\mathbf{x}$. One is *Gaussian Curvature*, which is

$$K_{Gaussian} = \kappa_1 \kappa_2 = \frac{LN - M^2}{EG - F^2} \tag{2.41}$$

The other is *absolute curvature*:

$$K_{abs} = |\kappa_1| + |\kappa_2| \tag{2.42}$$

In our approach to the 2D inverse problem, we want to add new knots at places of large curvatures and remove knots at places of small curvatures. By using tensor-product B-Splines, we can add or remove knots independently in two directions. To

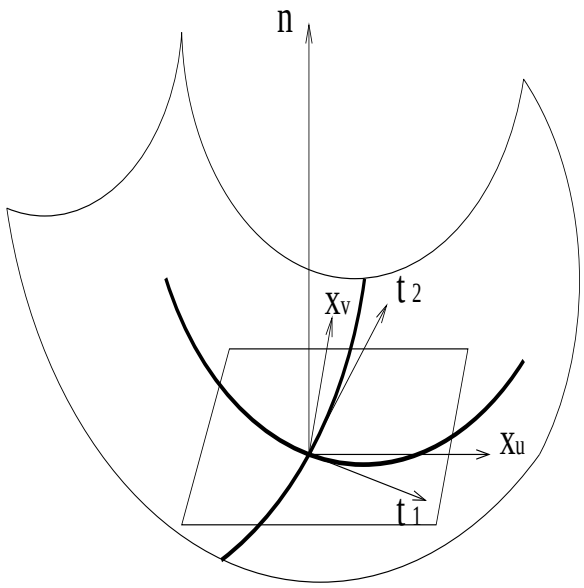


Figure 2.13: Example of only need to add knots in one direction

make the knot insertion/deletion procedure more efficient, we should decide whether to add or remove knots on a specific direction. For example, if we have a surface bending along u-direction and is relatively flat along v-direction (Fig. 2.13), the Gaussian or absolute curvature maybe large so that we would be tempted to add new knots in both u and v-directions. Apparently, there will be a waste for all the knots inserted on u-direction. To solve the problem, we think of using the principal directions, which are two directions correspond to the largest and smallest curvature respectively. If the principal direction of the largest curvature κ_1 is almost along the u-direction (t_1 in Fig. 2.13), we will only consider adding knots in the u-direction. Because it can be induced from the principal direction information that the measured surface is relatively flat at v-direction, thus no knots need to be added there. Further details will be discussed in Chapter 4.

Chapter 3

Edge-preserving Regularizer

In this chapter, we will continue the discussion on regularization methods in Chapter 2. A thorough discussion will show that the Tikhonov regularization tends to bring smoothing effects to the reconstruction thus is not good at preserving the edges. Then the idea of edge-preserving regularization will be introduced and analyzed. Some examples and comparisons will be provided at the end of this chapter.

3.1 Deficiency of Tikhonov Regularization

As introduced in Chapter 2, the general regularization method can be written in the following form:

$$f_{reg} = \underset{f}{argmin} \|Af - b\|_2^2 + \lambda\Omega(f) \tag{3.1}$$

In Tikhonov regularization, $\Omega(f)$ is of the 2-norm form : $\Omega(f) = \|Lf\|_2^2$, where L is a derivative operator. Thus the regularization term is defined as a sum of derivatives to the object f, which is a quadratic function (Fig. 3.1(a)) of Lf. In some cases, L is the first order derivative operator to f[19]. The idea is to counter the high-frequency artifacts by adding large constraint on them. The resulting reconstruction will be a

rather smooth one. The problem is that in many applications, the objects always have sharp edges which are very important for accurately determining the location and precisely describing the shape of discrete structures in the overall scene. In this case, the Tikhonov reconstruction will not be satisfying in that the smoothing effect blurs the edges as well as filters out the high-frequency artifacts. Note the different scales used in the four plots of Fig. 3.1, we can see the quadratic function brings significantly more constraints on large components of Lf, which usually correspond to sharp edges of the object. Therefore, we need to find other methods preserving edges better than Tikhonov regularization.

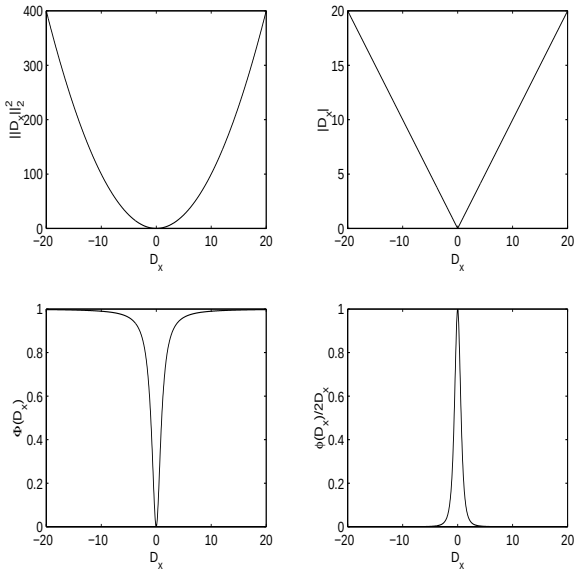


Figure 3.1: Illustrations of regularization functions (a)Quadratic (Tikhonov) (b)1-norm (c)Edge-preserving ($\varphi(t) = \frac{t^2}{1+t^2}$) (d)Edge-preserving ($\frac{\varphi'(t)}{2t} = \frac{1}{(1+t^2)^2}$)

3.2 Regularization Functions

A natural thought is to preserve larger fluctuations which are more likely to be edges of the object, and eliminate the smaller ones which are mostly to be high-frequency artifacts. To achieve this goal, we may want to apply 1-norm (Fig. 3.1(b)) instead of 2-norm to the regularization term, which will bring relatively smaller constrains on large components of Lf. A even better approach would be using the function as shown in Fig. 3.1(c) as the regularization term. In this case, only those regions with fluctuations exceeding a certain threshold will be considered to be edges and relatively little constraint will be applied there. The edge-preserving regularization can be written as:

$$\Omega_{ep}(f) = \sum_{i,j} \varphi[(D_x f)_{i,j}] + \sum_{i,j} \varphi[(D_y f)_{i,j}] \tag{3.2}$$

where D_x and D_y are first derivatives:

$$\begin{aligned} (D_x f)_{i,j} &= (f_{i,j+1} - f_{i,j}) \\ (D_y f)_{i,j} &= (f_{i+1,j} - f_{i,j}). \end{aligned} \tag{3.3}$$

It has been shown [24] that $\varphi(t)$ should have the following special properties in order to preserve the edges:

$$\begin{aligned} &\text{i) } \varphi'(t)/2t \text{ continuous and strictly decreasing on} [0, +\infty). \\ &\text{ii) } \lim_{t \to +\infty} \frac{\varphi'(t)}{2t} = 0 \\ &\text{iii) } \lim_{t \to 0^+} \frac{\varphi'(t)}{2t} = M, 0 < M < +\infty. \end{aligned} \tag{3.4}$$

And a group of candidates of function φ have been given in [24], in our application, we use $\varphi(t) = t^2/(1+t^2)$ (Fig. 3.1(c)).

3.3 Edge-preserving Regularizer

Given the edge-preserving regularization function $\varphi(t) = \frac{t^2}{1+t^2}$, we can rewrite the inverse problem in a new form with the following procedure.

First, for simplicity, we represent the 2D object in a 1D vector form by stacking all the elements column by column, i.e.

$$\mathbf{f} = [f_{11}, ..., f_{m1}, f_{12}, ..., f_{m2}, ..., f_{1n}, ..., f_{mn}]^T \tag{3.5}$$

Then, in 3.2 let $\mathbf{g} = D_x\mathbf{f}$ and $\Phi^T(\mathbf{g})\Phi(\mathbf{g}) = \sum_{i,j} \varphi[\mathbf{g}_{i,j}]$, thus

$$[\Phi(\mathbf{g})]_i = \sqrt{\varphi(\mathbf{g}_i)} \tag{3.6}$$

We can rewrite the problem as:

$$\mathbf{f}_{ep} = \underset{f}{argmin} \|A\mathbf{f} - b\|_2^2 + \lambda_x^2 \|\Phi^T(D_x\mathbf{f})\Phi(D_x\mathbf{f})\|_2^2 + \lambda_y^2 \|\Phi^T(D_\mathbf{f})\Phi(D_y\mathbf{f})\|_2^2. \tag{3.7}$$

Then we can deem it as a non-linear least squares problem which has the following form:

$$\min \|F(\mathbf{f})\|_2^2 = \mathbf{e}^T(\mathbf{f})\mathbf{e}(\mathbf{f}). \tag{3.8}$$

Our edge-preserving regularizer can be fitted into this form by defining:

$$\mathbf{e}(\mathbf{f}) = [(y - A\mathbf{f}), \lambda_x\Phi(D_x\mathbf{f}), \lambda_y\Phi(D_y\mathbf{f})]^T. \tag{3.9}$$

We can use the Tikhonov reconstruction as an initial guess then find the local minimum of the non-linear least squares problem by Gauss-Newton method [28].

In the above approach, two regularization parameters λ_x and λ_y are used instead of one. It is because in most applications, the degradation effects brought by kernel

A can vary sharply in different directions. And by specifically tune the regularization on both x and y directions, we can reconstruct the object more precisely.

3.4 Example

In this section, a 2D example will be used to compare Tikhonov and edge-preserving regularization methods. Further analysis about the complexity and the error of estimate(Cramér-Rao bound) will also be provided.

The examples are from the simulation of the cross-well tomography problem we set up in Chapter 2. Two ideal objects, one '+' shaped and one rectangular shaped (Fig. 3.4(a),(b)), are given and white Gaussian noise (SNR=20) is assumed. The Tikhonov reconstructions are shown in Fig. 3.4(c)(d), which have lots of small fluctuations in 'flat' area and smoothing effects on edges.

Now we apply the edge-preserving regularization to the problem. Firstly, notice that the Tikhonov reconstructions reflect the different properties of kernel A in x and y directions. Specifically, edges in x direction tends to be more difficult to restore than those in y direction because the transmitters and receivers are placed along y direction. Therefore, different regularization parameters λ_x and λ_y should be used in the two dimensions respectively. And intuitively, λ_x should greater that λ_y to add more regularization in the x direction. Secondly, by using Gauss-Newton method to solve the non-linear least squares problem, a initial guess of the object should be decided. Here, it is natural to apply the Tikhonov result as the initial value. The remaining difficulty would be to find the best regularization parameters.

For 1D problem, it has been shown in Chapter 2 that a L-curve can be used to determine the best regularization parameter λ. For the 2D problem here, though we have 2 distinct regularization parameters λ_x and λ_y, analogously they can be determined by finding the corner of the L-surface. The process is shown in Fig. 3.4.

The x and y dimension of a L-surface are the two regularization terms $\lambda_x \varphi(D_x \hat{f}_{ep}))$ and $\lambda_y \varphi(D_y(\hat{f}_{ep}))$ respectively. The z dimension is the residue $\|y - A\hat{f}_{ep}\|_2^2$. The best set of λ_x and λ_y would correspond to the corner of the L-surface, which has the largest curvature. In Fig. 3.4(c)(d), the curvature of the L-surface are computed, thus the optimum regularization parameters are determined by locating the largest curvature [22]. As what we expected, λ_x is greater than λ_y.

Then we can get the edge-preserving regularization result (Fig. 3.4(e)(f)). Compared with the Tikhonov result, it filters out the small high-frequency artifacts and at the same time preserves the edges well. As stated before, we have placed transmitters and receivers along y direction, thus we can get better resolution along y dimension which brings the asymmetry of the reconstructions between x and y directions.

However, the edge-preserving regularizer has one major backdrawing. By applying non-quadratic regularization functions, it turns into a non-linear least squares problem which significantly increases the complexity of the problem. Since the algorithm is pixel-based, and most applications have huge amount of pixels, in each iteration of the Gauss-Newton method all the pixel values need to be evaluated, the whole process will be very slow. Further discussions will be provided in Chapter 5 and a B-Spline basis solution will be given there.

Another interesting analysis is the Cramér-Rao Bound(CRB) evaluation. If the reconstruction is considered to be an estimate of the object, the CRB analysis in detection and estimation theory can be applied. Given the cost function as the estimation function $G(f)$, we can get the CRB for both Tikhonov and edge-preserving regularizer. (Appendix A). For our example, the CRB of Tikhonov regularizer is shown in Fig. 3.4(a)(b), the CRB of edge-preserving regularizer is shown in Fig. 3.4(c)(d). The CRB represents the upper bound on the variance of the estimation error for different cross-well regions(pixels). Comparing the results, Tikhonov regularizer put relatively equal efforts to the estimation of edges and other 'flat' areas.

Edge-preserving regularizer preserves the edges better, which means it locates the edges more accurately; thus as a compensation, the variance of the estimation error in such areas increases.

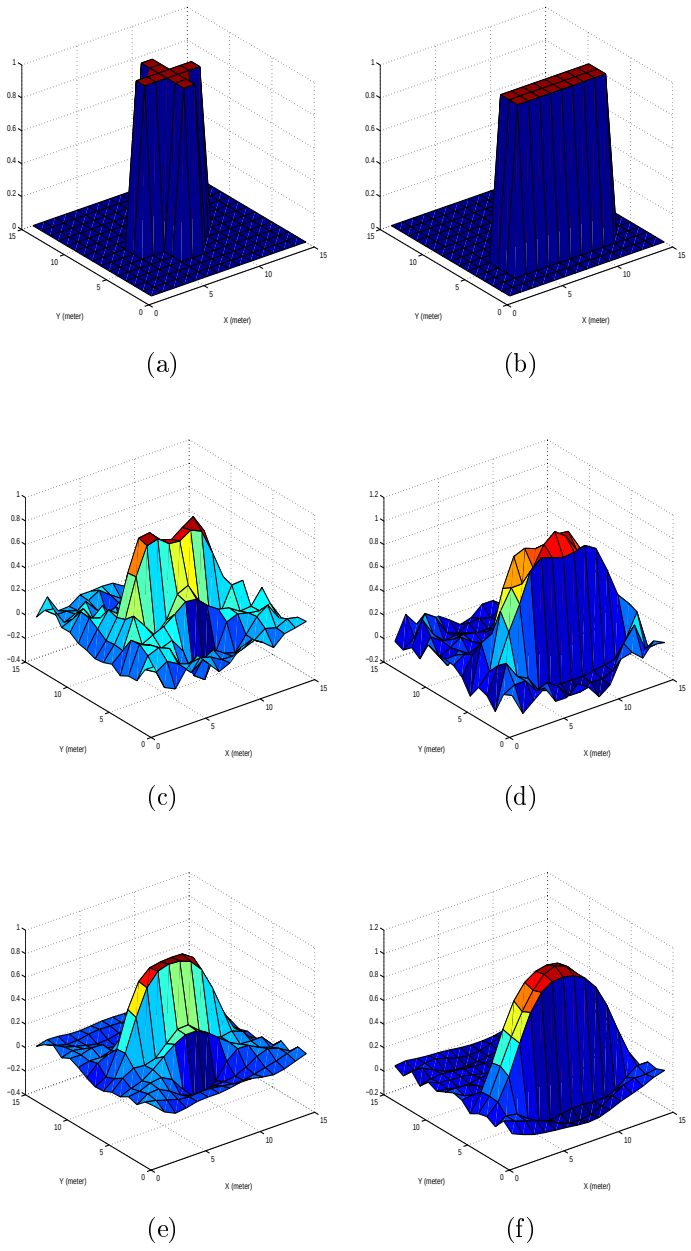
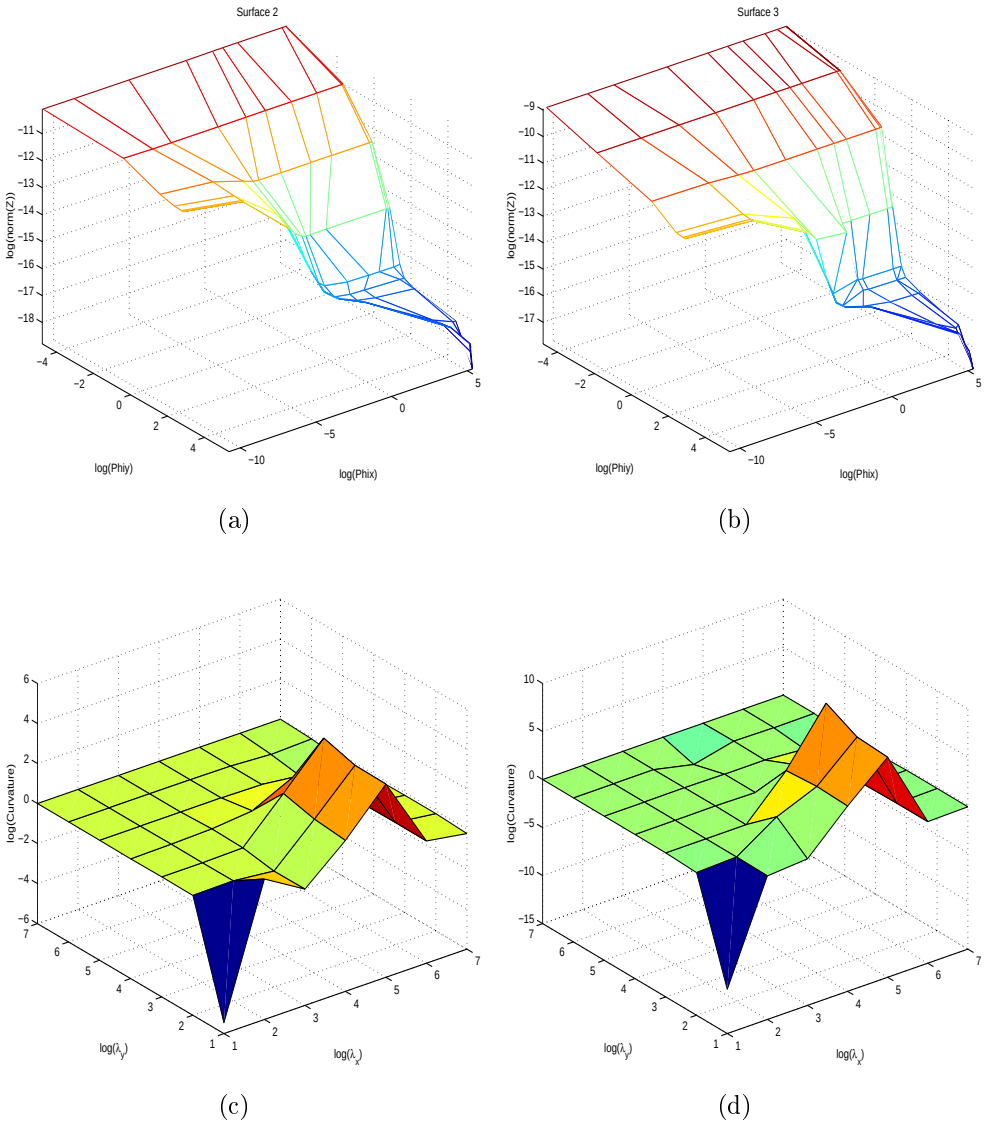


Figure 3.2: Example: (a),(b) objects; (c),(d) Tikhonov reconstructions; (e),(f) Edge-preserving reconstructions. Transmitters and receivers are placed along y dimension ($x = 0$ and $x = 15$).



(a) (b)

(c) (d)

Figure 3.3: Example: (a),(b) L-surfaces; (c),(d) Curvature of the L-surfaces.

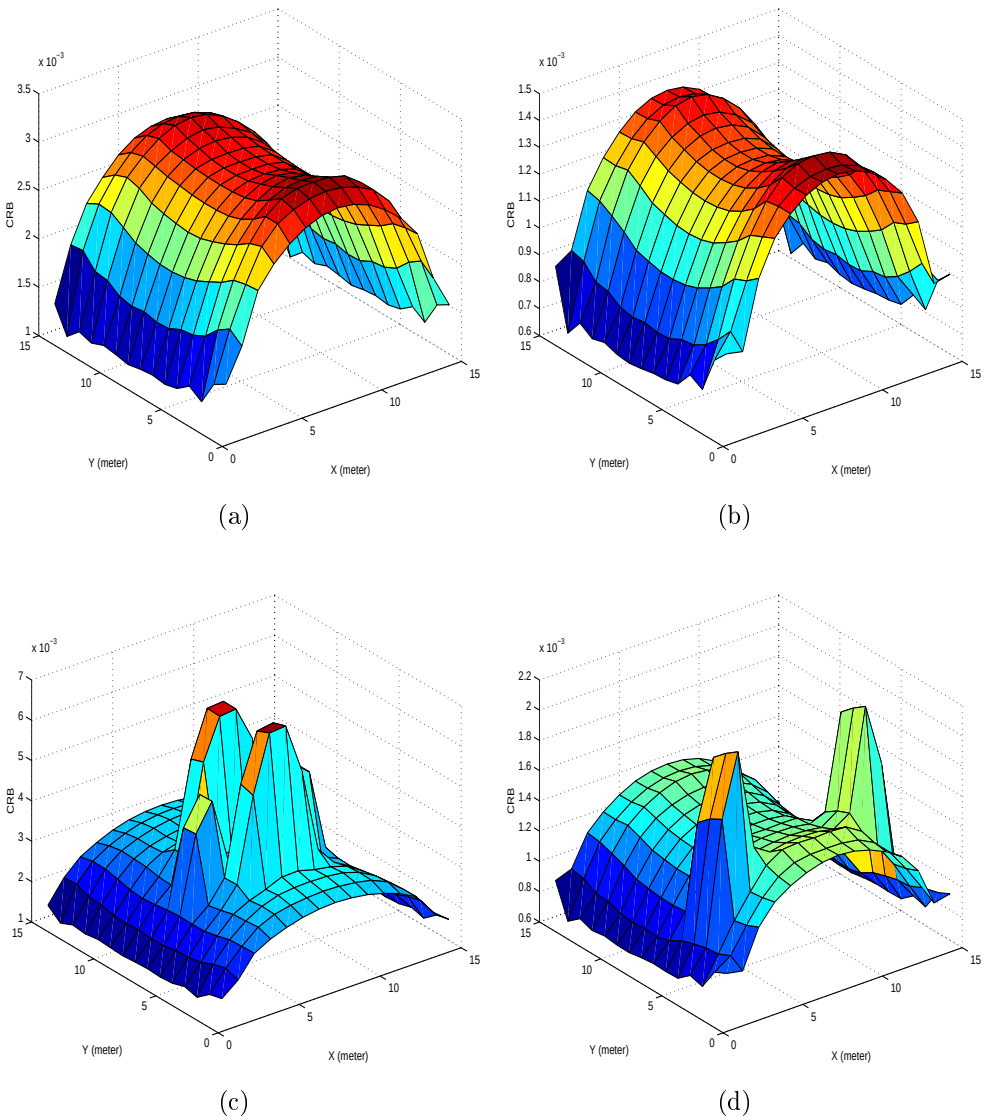


Figure 3.4: Example: (a),(b) CRB of Tikhonov regularizer; (c),(d) CRB of edge-preserving regularizer.

Chapter 4

Adaptive B-Spline Reconstruction

In Chapter 3, the edge-preserving regularization method was discussed and one major deficiency brought by solving the corresponding non-linear least squares problem is the significant complexity in applications. To solve the problem, a B-Spline based adaptive algorithm is developed. The cross-well 2D example will be revisited to show the improvement at the end of this chapter.

4.1 Motivation

For reconstruction methods previous introduced, both Tikhonov and edge-preserving regularization, are based on pixel-by-pixel estimate. In practical cases, especially in 2D or even 3D problems, the total number of pixels is always very large. Thus the reconstruction problems will need significant time and effort to obtain a solution. The situation will be even worse for edge-preserving regularizer, because the Gauss-Newton method used to solve the corresponding non-linear least squares problem requires evaluation of estimations of all the pixels in each iteration.

To solve the problem, a natural thought is that if we know the shape of the object, a proper shape-base template would be efficient to precisely describe the

object and significantly reduce complexities [29]. However, in practice, we don't have prior knowledge of the object shape and size, so a compromised way would to find a set of proper basis to represent the object. The basis should have the properties that flat regions can be represented by one or few coarse scale functions while more fine scale functions can be used to represent details of regions of interests , e.g. edges.

Based on the discussions in chapter 2, B-Spline basis would be a good candidate. First, it is natural to represent an unknown object by a set of k-order piecewise polynomial functions. Second, the multiscale property of B-Splines can be utilized to efficiently represent the object hence decrease the complexity of the problem. Third, efficient algorithms of evaluating and manipulating B-Splines exist which will not bring much extra efforts in dealing with the basis itself. Fourth, the smoothness and scale of B-Splines can be easily manipulated by knots distribution, which means an adaptive algorithm may exist for choosing a suitable set of B-Splines.

4.2 Inverse Model using B-Splines Basis

Using B-Spline basis instead of pixels to represent object f, the inverse problem can be remodeled as:

$$y = Af + n = ANa + n = Ba + n, \tag{4.1}$$

where N is the B-Splines basis and a is a vector of expansion coefficients for the B-Splines. Thus the problem can be replaced by a new inverse model whose degradation kernel will be $B = AN$, and the new object to be reconstructed is a. If proper B-Splines basis are chosen to represent f, i.e. the representation is efficient, the dimension would be significantly decreased compared with that of f. Furthermore, the edge-preserving regularization method can be used to solve the new problem.

In the 2D example, tensor-product B-Splines can be utilized because it is a simple extension of 1D B-Splines, and knots in x and y direction can be manipulated separately. The drawback is that the rectangular patches may lose some flexibility. Two examples of tensor-product B-Splines and corresponding knot distribution have been shown in Fig. 2.7.

4.3 Optimum Knots Distribution

Many sets of B-Spline basis can be chosen to represent a certain object, the best would be the one that yields the most efficient representation. In another word, the B-Spline basis needs to be distributed in the way that flat regions are covered by few coarse B-splines while a larger density of B-Splines cluster at rough regions. By utilizing such a set of B-Splines, the complexity of the inverse process can be decreased.

From the discussions of B-Splines in Chapter 2, the number of knots corresponds to the number of B-Splines being used, thus the number of unknowns in the revised model (4.1). The knot locations directly affect the support(scale) and distribution of B-Splines. Therefore, the problem is to find an optimum knots distribution of the B-Splines. It has two aspects: 1. How many B-Splines are sufficient to represent the object? 2. Given a fixed number of knots (B-Splines), what is the optimum distribution of those knots?

The second problem is straightforward that the knots distribution should reflect the shape/smoothness of the object. Specifically, large density of knots should cluster at rough regions, i.e. regions with large curvature. And only few knots would be enough to represent the remaining flat regions.

The first problem can be interpreted as the following: Given a certain tolerance of estimation error, what is the fewest number of knots/B-Splines can be used to make

the reconstruction? Roughly the process involves inserting new knots where the estimate is poor and deleting redundant knots where the estimate is over accurate than needed which will unnecessarily increase the complexity of the problem.

Based on the above discussions, an ideal solution would be able to find the optimum set of B-Splines, including the number of B-Splines and the placement of each B-Spline. It is difficult to accomplish because the searching effort will be significant, and even if we can find such solutions, the computational cost may be not worth the improvement gained compared with the pixel-based solutions. An alternate way would be to efficiently find a suitable set of B-Splines instead of an optimum one. The desired algorithm will be able to end up with a highly suitable knot distribution in the sense that the number of knots is much fewer than that of pixels, while the reconstruction does not degrade much.

Our processing approach can be represented in a tree structure [46] [47]. As shown in 4.1, for simplicity we use binary tree to show the cases of 1D problem, each level of the tree corresponds to different number of knots being used. Then, we can think of using an adaptive algorithm to automatically find the suitable knot distribution. First, the algorithm should be able to iterate between coarser and finer level of reconstruction to determine the most suitable number of knots to be used. If one knot is not good enough to cover a certain region, the algorithm should be able to split it into 2 knots on the next finer level and vice versa. Second, at the same time, the algorithm should be able to intelligently alter the locations of knots to make better reconstruction in a certain resolution level. Therefore, the placement of each knot is not fixed but varies during the iteration of the algorithm. The overall algorithm is to find a set of knots with much fewer knots than pixels based on which the estimate is within a certain error.

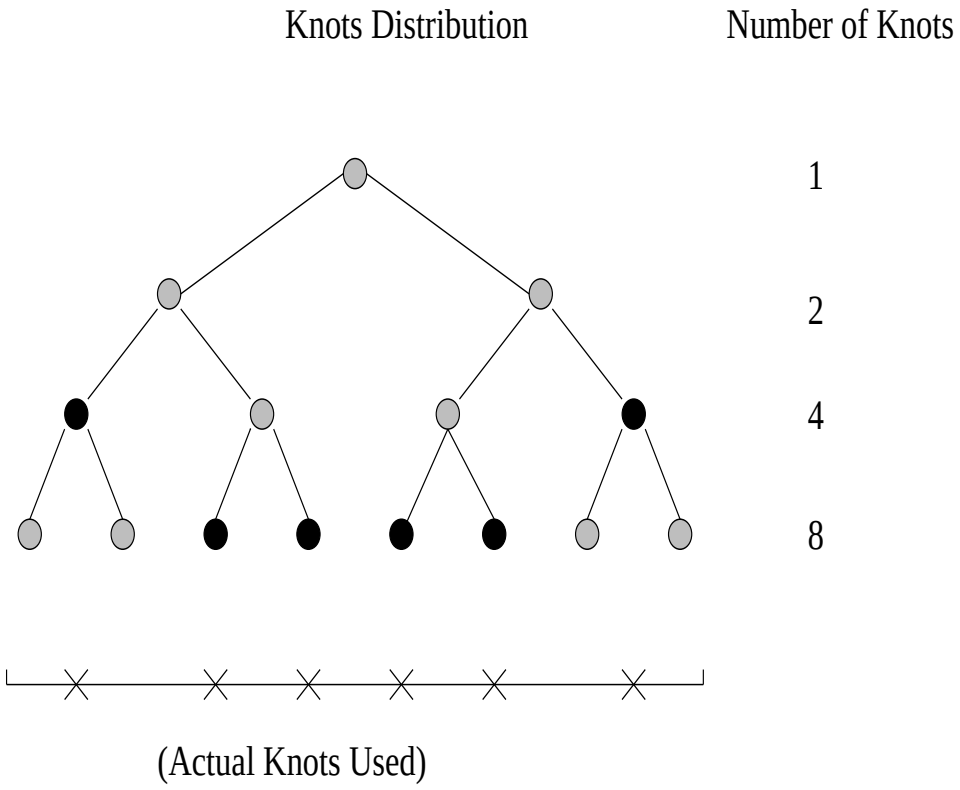


Figure 4.1: Illustration of Tree Structure for Knots Searching (1-d case)

4.4 Knots Refining

Given a coarser level of reconstruction, one may want to add more details in some interesting areas, e.g. edges. A simple way is to double the knots/B-Splines used in reconstruction. The process is to insert new knots in between all existing knots, thus the resolution of all areas are doubled. The drawback is that the total complexity is at least doubled. For example, in the 1D problem showed in Fig. 2.8, notice that many knots are inserted in uninteresting areas, e.g. flat areas where existing knots are sufficient to have an acceptable estimation Fig. 2.8(c). Therefore, a more adaptive way should be found to insert new knots. Specifically, knots should only be inserted as necessary, i.e. the algorithm should automatically know where the areas

of interests– potential edges are and only inserts new knots in such areas.

In 2.8(a), notice that though the coarser level(with only 5 knots) reconstruction is rough, we can still get some information of the object shape/smoothness. In another word, the coarser level reconstruction gives a hint to the shape/smoothness of the finer level reconstruction. Since we are interested in edges which have large roughness or discontinuity, we can use this hint as a guide of inserting new knots. The process is called *knot refining* because it is not a simple knot insertion procedure[41][43]. It also involves curvature computing (Chapter 3.3) based on current estimate. The knots refining process is described as the following:

Algorithm 1: *Knot Refining*

Input: Knot sequence $P_i, i = 1, 2, ..., m$

Corresponding reconstruction $\hat{f}(P)$

Threshold K_0

Output: New knot sequence $Q_j, j = 1, 2, ..., n$

Begin:

$K = CurvatureComputing(P, f(P))$;

for $i = 1$ to m,

if $(K_i > K_0)$ then

$KnotInsertion(K_i)$;

(K_i is the curvature at knot P_i)

end if

end for;

End

Specifically, in the above algorithm, the knot insertion function splits one knots into two equally space knots to cover the same region.

In the above algorithm, an important assumption is that coarser level reconstructions will provide correct guide to get finer resolutions. But the reconstruction may not be good enough to show the rough structure of the object if only too few B-Splines are used. It can be avoided, in practice, by always starting from a relatively fine level guess, and the knot deletion algorithm proposed in next section will guarantee not too many knots will be deleted.

Another important issue is to choose a proper threshold K_0 for inserting new knots. If K_0 is too large, i.e. only few knots will be inserted to the areas with very large curvature, the resulted reconstruction will be only focusing on several large abruptly changing regions. Another drawback is that the refining process may be misled by some false edges in coarser level. On the other hand, if K_0 is too small, too many knots will be inserted while most of them will bring no good to the reconstruction and have to be deleted later. How to automatically find the proper K_0 will need further investigation. In this thesis, the threshold K_0 is determined by experimenting different values and comparing the results.

4.5 Knot Pruning

After inserting a set of new knots, a refined reconstruction can be obtained. The new reconstruction should have different shape/smoothness and different curvature structures than the previous coarser one. Since the knots inserted were based on the coarser reconstruction and it was only a guess of the underlying refined details, some of the knots may be misplaced, i.e. knots were actually inserted in a flat region rather than the expected rough region. On the other hand, the regions considered to be flat may appear to have large curvatures. The *Knot Pruning* algorithm is used to prune the new set of knots, find and delete those improper and redundant ones.

4.5.1 Weight of Knots

The most important problem here is how to prune the knot sequence and find those redundant ones. Therefore, a measurement of each knot's importance should be provided to determine which ones are less important to the reconstruction and can be deleted.

The most important knot should be the one that contribute the most to the current reconstruction, i.e. by deleting it, the reconstruction will have the greatest change. The underlying thought is that only few knots are needed to well approximate a flat region, while a cluster of knots may needed in rough regions with lots of details to make an equally good approximation. So those knots appear to be less important to the reconstruction means there are too many of them than needed compared with the information in these areas, i.e. redundant. A natural measurement of the change in reconstruction is the underlying cost function for the problem. For edge-preserving regularizer, the cost function is:

$$\text{cost } = \|y - Ba\|_2^2 + \lambda_x^2 \varphi[D_x(Ba)] + \lambda_y^2 \varphi[D_y(Ba)]. \tag{4.2}$$

The edge-preserving regularization is to find the solution $\hat{a}$ minimizes the cost function. And apparently, the more knots used(or the larger dimension the a has), the better resolution the reconstruction should have. Thus the cost function increases each time a knot is deleted from current knot sequence. And the knots that cause the larger increase of the cost function are of the more importance, i.e. have larger weight in the reconstruction.

An illustration is shown in Fig. 4.2. Here a simple B-Spline approximation problem is used to get better understanding. The cost of deleting each knot is measured by Mean Squared Error(MSE). For the cross-well reconstruction problem, the situation is quite analogous, except that the cost function is measured by the edge-preserving

regularizer.

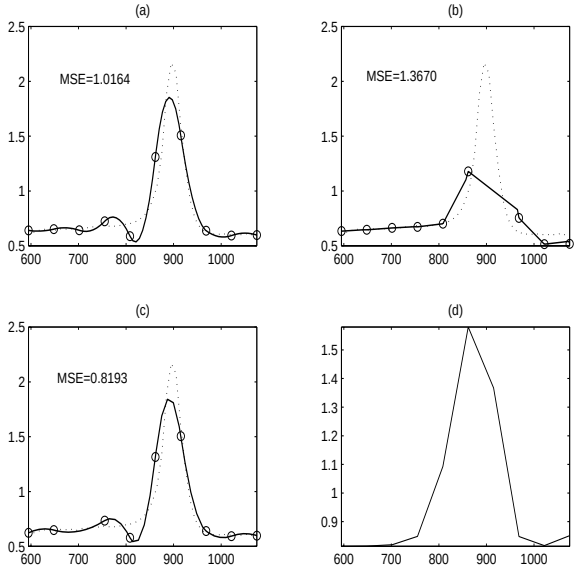


Figure 4.2: Illustration of weight of knot. (a)10 equally-spaced knots to approximate the data(dotted line); (b)Approximation of deleting the 7th knot; (c)Approximation of deleting the 3rd knot; (d)Cost(Weight) of each knot.

4.5.2 Knots Deletion

After measuring the weight of each knot, we can start to delete those redundant ones. First, sort the knots in the descendant order of weight. Then delete knots one by one in the order of their weights until the resulting reconstruction is coarse enough but not too much worse. The criteria is set to be that the cost function value after deletion does not exceed the one at the beginning of each iteration(i.e. before the knot insertion). The idea is that the whole knot insertion and deletion process is actually to improve the knot distribution. Thus, for each iteration of such process, the resulted cost should not exceed the initial one, otherwise, the knot distribution is not improved but worsened. The specific mechanics of deleting a single knot has

been discussed in many papers [38] [39].

Consider a simple 1D example in Fig. 4.3, notice the knot distribution here is not the same as of Fig. 4.2. More knots are assigned to the bump area to simulate the result of inserting cluster of knots in large curvature regions. In this case, the weight of each knot are shown in Fig. 4.3(b). If we delete knots in the order based on this weight measurement, we may delete the 12th knot first then the 13th knot because they have the smallest cost. However, as shown in Fig. 4.3(c), after deleting the 12th knot, the actual weights of some knots change, specifically for those adjacent knots. If we still try to delete the 13th knot obeying the deletion order given by the previous weight measurement, the resulted approximation error may be too large so that the deletion process will terminate. The result is far from satisfying, because actually we can delete more knots than we did without violating the tolerance.

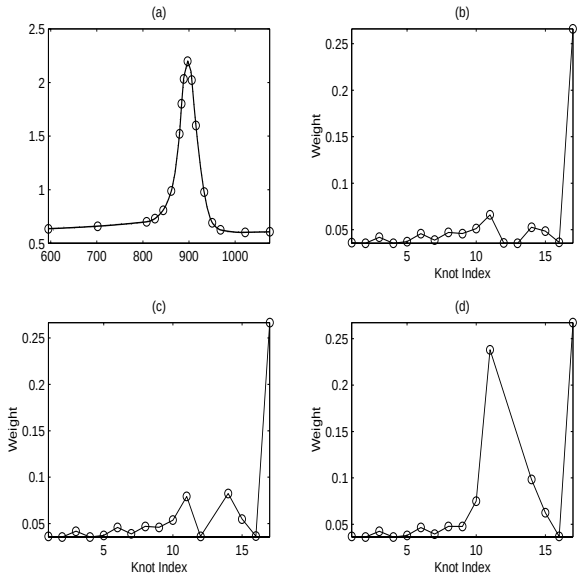


Figure 4.3: Illustration of vicinity effects of weight measurement. (a)Initial knots distribution and approximation curve; (b)Weight measurement; (c)Weight of each knot after deleting the 12th knot; (d)Weight of each knot after deleting the 12th and 13th knot.

The problem arises from the fact that B-Splines have supports leaking into adjacent knot intervals, thus we can not ignore their effects on vicinity regions. In another word, deleting a knot does have effects on its vicinity regions, and change the actual weight of adjacent knots. To solve the problem, one solution would be to recompute the weight of each knot in each step after a knot has been deleted. This will significantly increase the complexity of the algorithm. Considering the local support of B-Splines, deleting one knot will not bring much effect to remote regions. Thus, an alternative and much efficient way to avoid the vicinity effects may be continue using the weights we get before the deletion but simply not to delete adjacent knots to the one who has been just deleted. Therefore, we will only need to measure the weight of each knot once and choose and delete several knots at a time. Since we have avoided deleting adjacent knots, the measured weight of each knot would be a good estimate of the actual weight during the knot deletion procedure.

The knot pruning algorithm is described as the following:

Algorithm 2: *Knot Pruning*

Input: Knot sequence $Q_i, i = 1, 2, ..., m$

Corresponding reconstruction $f(Q)$

Threshold C_0

Output: New knot sequence $P_j^*, j = 1, 2, ..., n$

Begin:

$W = Weight(Q);$ **Compute weights(costs) **

$[W, index] = Sort(W);$ **Sort in ascending order**

$i = 0; cost = 0;$

$P^* = Q;$

While $cost \leq C_0$

{

$i = i + 1$

if $(Neighbour(Q(index(i)))) \in P^*$ then {

$\{P^*\} = \{P^*\} - Q(index(i))$ *Delete one knot at a time**

$cost = Weight(P^*)$;

}

}

End

4.6 Overall Algorithm

The knot refining and pruning algorithm can be integrated together to form the overall algorithm to find a suitable knot distribution. The flow chart of the overall algorithm is shown below in Fig. 4.4:

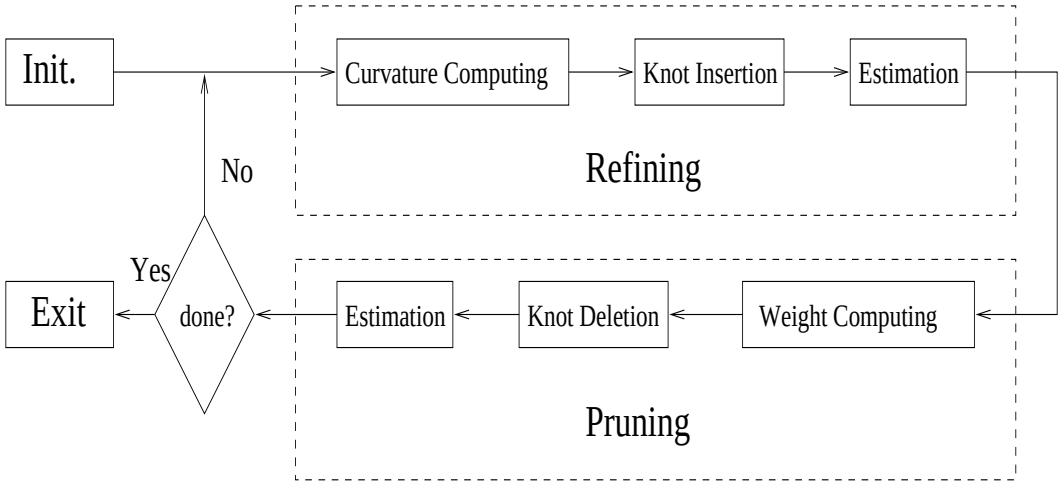


Figure 4.4: Flow chart of knot optimization algorithm.

The knot refining part is to efficiently insert new knots according to the curvature information of current coarse reconstruction. The knot pruning part is to find

and delete the redundant knots, either previously exist or newly inserted, based on a finer reconstruction. Therefore, the overall effect is to insert knots firstly then make correction to it by pruning. In each iteration, the resulted knot distribution is guaranteed to be better than the previous one in the sense of having less cost. As the algorithm converges, a more suitable knot distribution can be found, i.e. with much fewer number of knots than that of pixels and the estimation is within certain error. The algorithm will terminate while no more improvement has been obtained, i.e. the current cost function almost equals the last result.

Fig.4.5 shows the result of using our algorithm for the example in previous Chapters using the Titanium data. The resulting knot distribution makes the approximation with $MSE < 0.1$. Compared with Fig. 2.8, our algorithm gets a smaller approximation error while using fewer but well-distributed knots. In Fig. 2.8(d), 12 selected knots were used. Our algorithm only ends up with 10 knots and the approximate error is less in the means of MSE.

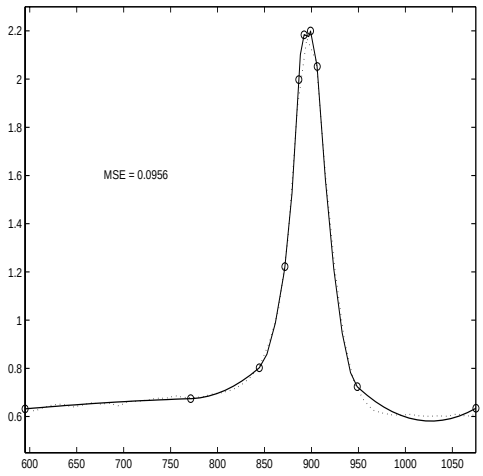


Figure 4.5: Using knots refining and pruning algorithm on the Titanium data fitting problem.

4.7 Examples

The 2D cross-well example used in Chapter 3 is revisited here. The previous Tikhonv reconstruction and edge-preserving recontruction results are shown in Fig. 4.6 as comparisons. The resulting reconstructions and corresponding knot distributions are shown in Fig. 4.7. The algorithm converges fast (Fig. 4.8). The knot distributions are what we expected, i.e. cluster around edges and only few on flat areas. The total number of B-Spline basis used are less than 50, compared with the total pixel number $21 \times 21 = 441$, approximately a 8 times or more reduction in complexity is achieved. Consider that the objects are large and have many details, especially for the '+' shaped one, compared with the total area, the results are satisfying. For applications with larger dimensions and relatively small objects, the decrease in complexity will be significant. Although the MSE almost doubled , we can still state by visionary obervations that the B-Spline based reconstructions do not degrade much compared with the pixel based ones.

Another example is the 3-layer structure we may be of interest in practice, the result is shown in Fig. 4.8. The algorithm works well, converges fast and only ends up with about 30 knots.

The only problem is that the saw shaped parts of the object are not well reconstructed. Only one knot is assigned in the vertical direction. The problem arises from the using of tensor-product B-Splines, the approximation patches should always be rectangular. For the object we have, in the vertical direction, the upper side of it has one significant edge, while the lower side of it has lots of details(saws). Since we were using rectangular patches, a compromise should be made for the knots distribution here. Using few knots will decrease the complexity but sacrifice the details at lower part. Using more knots are a waste for the upper side but are needed by the details at lower part. It's a difficulty for our approach. Better solutions may be using triangular

patches instead of tensor-product B-Splines[49] [50]. It will be considered in future work.

4.8 Monte-Carlo Simulation

To better understand the performance of the algorithm, we will analyze Monte-Carlo simulations in this section. The process is to repeatedly simulate the test by applying different noises at the same SNR level. With large number of such tests, we can get the statistics of how the algorithm performs under such SNR level.

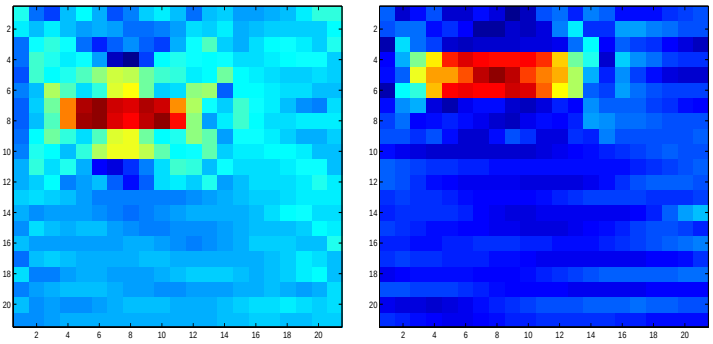
In this section, we use a 3-layer example to simulation the actual situation of the cross-well problem stated before. The result in Fig. 4.9(b) is obtained by applying the B-Splines edge-preserving reconstruction method. The algorithm converges fast and the overall time of achieving the estimate is about half of the time of the pixel-based reconstruction.

In Fig. 4.9(a), the average pixel-based reconstruction of the 3-layer problem is shown. The average B-Spline reconstruction is shown in Fig. 4.9(b). The B-Spline result is quite good and does not degrade much than the pixel-based one, and at the same time, by utilizing much fewer basis, the complexity has been significant reduced. To understand the knot distributions of the Monte-Carlo simulations, a 2D histogram for all the knots ended in the 200 tests is shown in Fig. 4.9(c). It shows the knot distributions have certain relationship with the object structure as we expected. Specifically, there are higher possibilities for knots being place around edges of the object, and the density of knots are also larger there. To make it clear, a scaled version of knot distribution has been shown in Fig. 4.9(d). It only shows those knots whose frequencies of being chosen are larger than 0.1 during all the 200 tests. In another word, the knots shown here are essential and most efficient to represent the object.

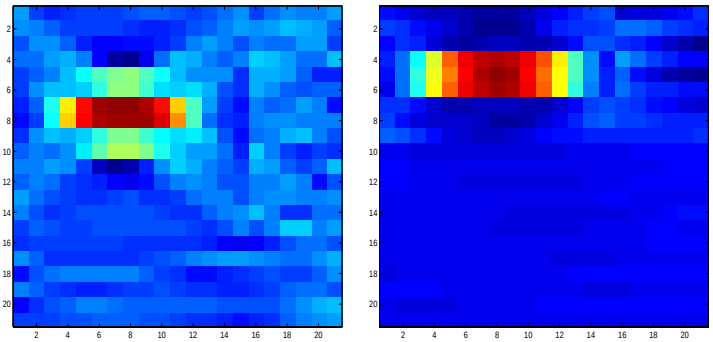
To further determine the average knot distribution, since we used tensor-product B-Splines, separate the 2D histogram into two 1D histograms will be helpful. The results are shown in fig. Fig. 4.10.

In Fig.4.10(a)(b), the 1D histograms of knot distribution along x and y directions are shown. Since we use tensor-product B-Splines, the separation of knots in different dimensions is applicable. By choosing certain threshold, we can determine which knots are to be used. Here, a threshold of 0.13 has been applied to both x and y dimensions, and the resulted reconstruction and knot distribution are shown in 4.10(c)(d). Since we use tensor-product B-Splines, the overall knot distribution is abided by the rectangular grid. The variance of estimates over the 200 reconstructions are shown in Fig. 4.11.

The Monte-Carlo simulation has shown that the B-Spline algorithm is efficient that it ends up with much fewer knots than that of pixels and the reconstruction does not degrade much. Generally, the algorithm can lead to a 8-10 times decreasing in the complexity of the problem.



(a) (b)



(c) (d)

Figure 4.6: Example: Reconstruction obtained by Tikhonov regularization and edge-preserving regularization method. (a)'+' shaped object, Tikhonov regularization (MSE = 5.7393) (b)rectangular object, Tikhonov regularization (MSE = 5.2943) (c)'+' shaped object, edge-preserving regularization (MSE = 3.3930) (d)rectangular object, edge-preserving regularization (MSE = 2.9372)

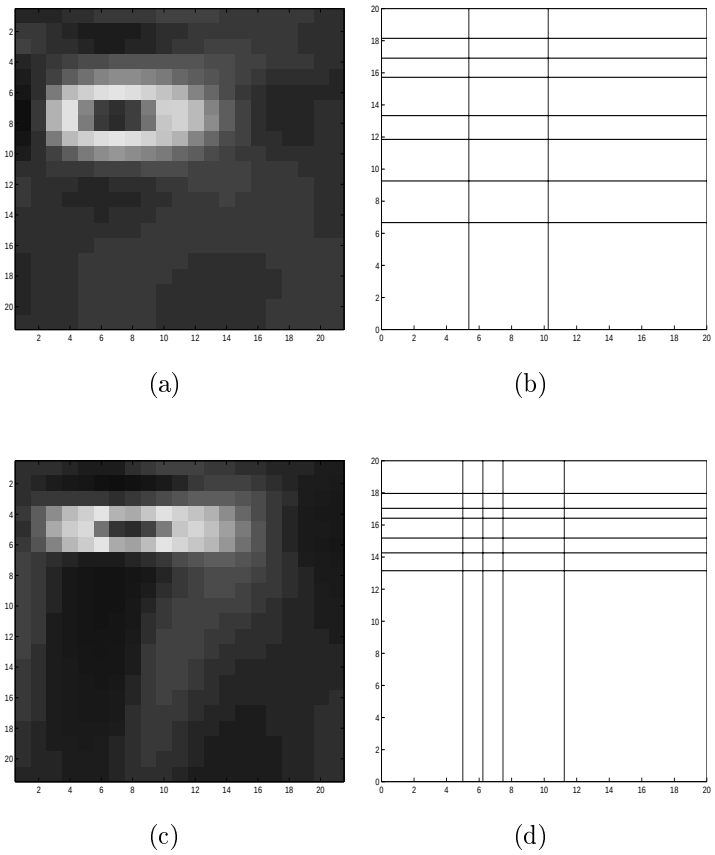


Figure 4.7: Example: Reconstruction and knots distribution using B-Spline based edge-preserving regularizer. (a),(b) '+' shaped object; (MSE = 7.8461) (c),(d) rectangular object. (MSE = 5.7867)

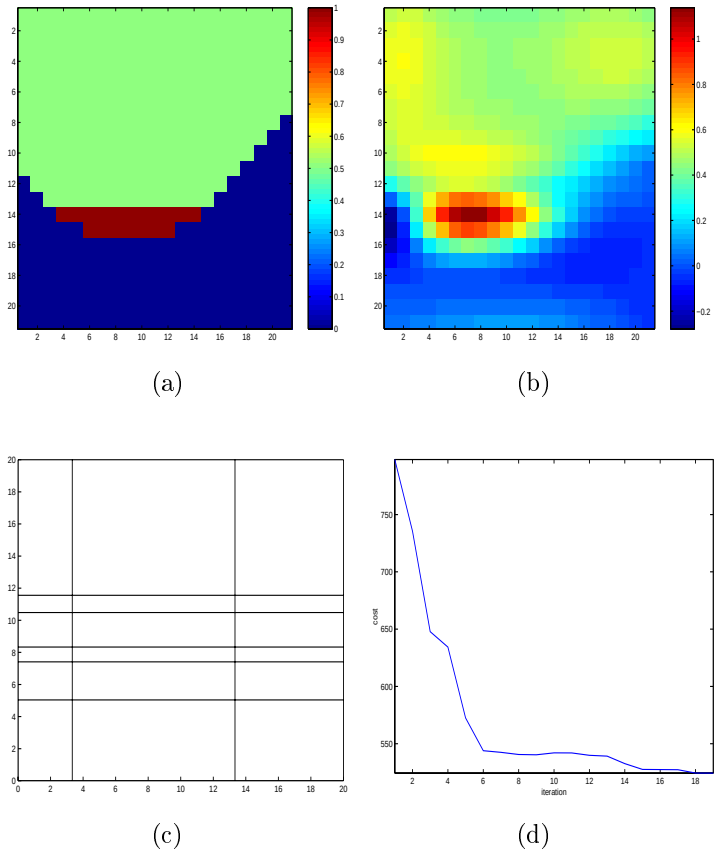
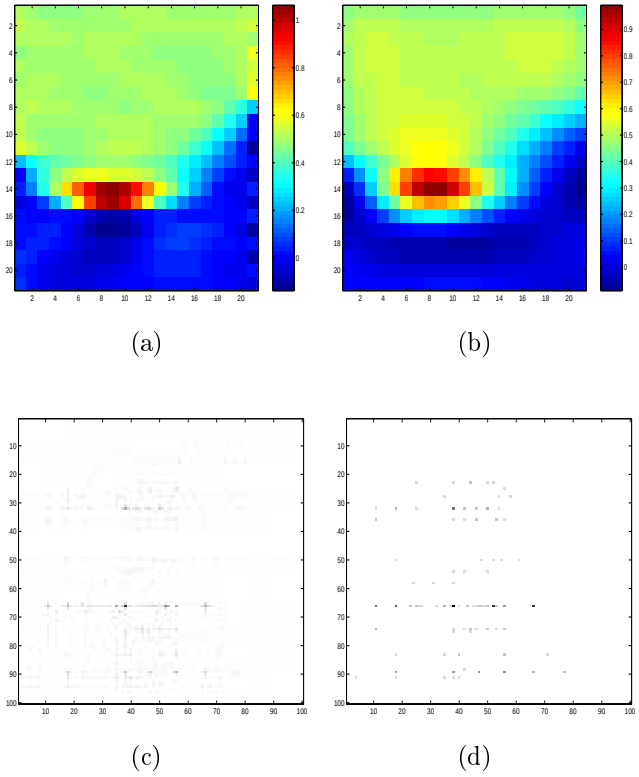


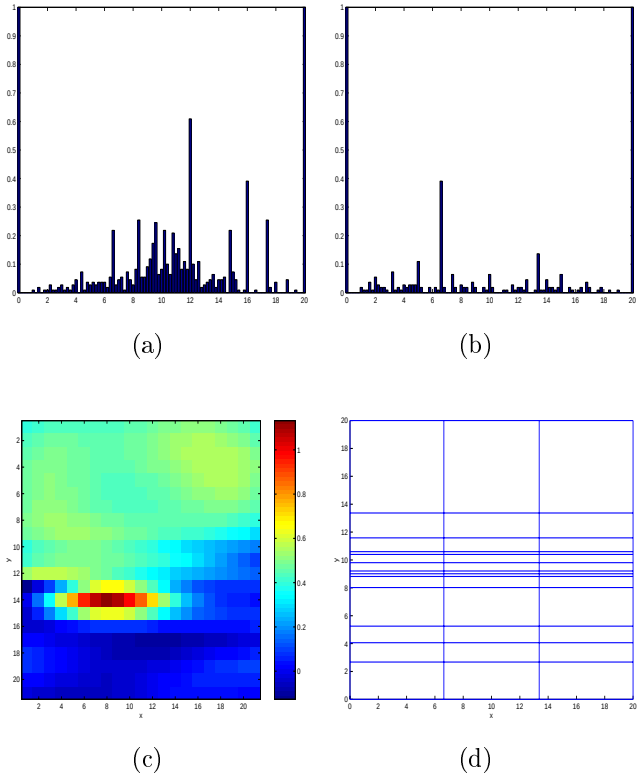
Figure 4.8: Example: B-Spline reconstruction for 3-layer problem (a)Object; (b)Reconstruction; (c)Knots distribution; (d)Convergence of cost (Each iteration consists of knot inserting and knot pruning procedure.)



(a) (b)

(c) (d)

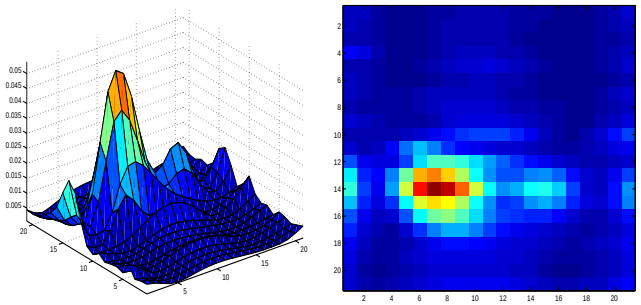
Figure 4.9: Monte-Carlo simulation for the 3-layer problem. (SNR = 20dB) (a)Average pixel-based reconstruction (MSE = 2.0874); (b)Average B-Spline reconstruction (MSE = 5.8922); (c)Total knots distribution; (d)Scaled knots distribution.



(a) (b)

(c) (d)

Figure 4.10: Knot selection by using separate 1D histograms. (a)Histogram of knot distribution along x direction. (b)Histogram of knot distribution along y direction. (c)Reconstruction based on histogram analysis. (d)Corresponding knot distribution.



(a) (b)

Figure 4.11: Variance of the Monte-Carlo simulation (2 different viewpoints)

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis developed an adaptive B-Spline method for solving low order image reconstruction problems. Also an edge-preserving regularizer was introduced and integrated to make fine reconstructions. The edge-preserving regularizer has been shown to be very efficient in countering the smoothing effects of Tikhonov regularization and preserving the edges. But for multi-dimensional problems, the edge-preserving regularization leads to significant large complexity. To overcome this deficiency, B-Spline basis was chosen to make simple but efficient representation of the objects which decreased the dimension of the problem. Furthermore, an adaptive knot refining and knot pruning method was developed to automatically find a suitable knot distribution which is critical in solving the problem. Some examples were shown to illustrate the adaptive algorithm and the results were satisfying.

In Chapter 2, some background knowledge was introduced. First, the linear inverse problem model was given. The difficulty in solving the inverse problems lies in the ill-posed nature of such problems. As a result, the least squares solution always tends to have large high-frequency artifacts, and small observation errors would bring

large errors in solutions. After analyzing the problem from the SVD(Singular Value Decomposition) point of view, the regularization methods were introduced to filter out the abruptness brought by small singular values σ_i. The smoothing effects of Tikhonov regularization were investigated and led to the thought of edge-preserving regularization was introduced. Second, we introduced the basic knowledge of B-Splines. After giving the definition, some interesting properties, such as local support, minimal support and linear independent of B-Splines were introduced. A further study in the effects of knot distribution to the smoothness of the approximation showed that we can change the approximation curve/surface by simply manipulating the knots. An example was shown that proper placement of knots can lead to efficient representation of the object. Third, the geometries of B-Splines were introduced to compute the B-Spline surface curvatures, which is the guide of our manipulation of knots distribution. Tensor-product B-Splines was used in 2D case, and Gaussian and absolute curvatures were introduced.

In Chapter 3, a thorough discussion of edge-preserving regularization was given. The deficiency of Tikhonov regularization is that it blurs the edges while filtering out high-frequency artifacts. A study on the regularization term showed that a non-quadratic regularization functions can be used to preserve edges. Some common properties of such regularization functions were given and a non-linear least squares solution was proposed to solve the problem. Then a simple 1D example showed the edge-preserving regularizer did work. While the CRB analysis showed an interesting fact that with better preserving the edges, the error of estimate of edge-preserving regularizer actually had larger variance. A major problem of edge-preserving regularizer was its huge complexity, since the Gauss-Newton method need to evaluate all pixels in each iteration.

In Chapter 4, a B-Spline based adaptive algorithm was developed to decrease the large complexity of edge-preserving regularizer. The motivation was from that

the object can be efficiently represented using B-Splines. Thus after remodeling the inverse problem we had, we found that the dimension of the problem can be significantly decreased if we could find a suitable set of B-Splines. The process of finding such basis was actually to find a good knot distribution. Based on the discussions in Chapter 3, a knot refining and pruning algorithm was developed to adaptively finding a suboptimal solution. The refining part was to efficiently insert new knots according to curvature information. The pruning part was to find and delete those redundant knots. A measurement of the weight of each knot was given to assist the pruning process. The whole algorithm works to find a suitable set of knots with the reconstruction is within certain error. The 2D cross-well problem was revisited by using the algorithm, and the result appeared to be satisfying. It was able to decrease the dimension of the problem to 1/8 compared with the pixel-based approach, and the iteration process converges in about 10 iterations .

5.2 Future Work

Overall, this thesis has shown that the adaptive B-Spline incorporated with edge-preserving regularizer is useful and efficient in solving image reconstruction problems. The algorithm can adaptive locate the edges of the object as well as significantly decrease the complexity. However, there are still many aspects need to be explored and refined.

- The edge-preserving regularizer has been shown to be able to reconstruct the edges well. However, the CRB analysis in Chapter 3 also showed that the estimation variance at the edges increased compared with Tikhonov solutions. It is an interesting phenomenon. It seems the edge-preserving regularizer can locate the edges more accurately, but at the same time the estimation variance has been sacrificed. Further analysis in the framework of detection and estimation

theory may lead to some in-depth understanding of the problem.

- The knot refining process used curvature information as a guide, thus it depended much on the selection of threshold K_0. We were just using heuristic values in the examples. One could use a set of different thresholds to provide multiple results, each focusing on curvatures of different ranges. It'd better if there is a way to adaptively set the threshold.

- The knot pruning process is currently a greedy search process. An estimation of the weight of each knot was made first, then we tried to delete them one by one until the error of estimate exceeded a certain value. Though we have considered the vicinity effects and will not delete adjacent knots in the same iteration, the weight estimation may not be good enough to guide the deletion process. Actually, we also tried to organize the knots in quad-tree structure and use Branch-and-Bound method to search it [51]. Unfortunately, it was too difficult to determine the bounding function and we did not get good results. Further study of the pruning method may be helpful.

- In 2D case, we used tensor-product B-Splines. The advantage is that it is easily to be manipulated and the knots on different dimensions are separable. The major disadvantage is that we are bound to use rectangular patches to make the estimation thus lose some flexibility in manipulating the knots. A triangular patch [49] [50] may be interesting, though the computations are more difficult, the flexibility it brings will be worthy of trying.

Bibliography

[1] M. Bertero, "Linear Inverse and Ill-posed Problems", *Advances in Electronics and Electron Physics*, Vol. 75. Academic Press, 1989.

[2] A. S. Willsky, P. W. Fieguth and W. C. Karl, "Multiresolution Stochastic Imaging of Satellite Oceanographic Altimeter Data", *IEEE International Conference on Image Processing*, Vol. 2, 1994.

[3] E. L. Miller and A. S. Willsky, "A Multiscale Approach to Sensor Fusion and The Solution to Linear Inverse Problems", *Applied and Computational Harmonic Analysis*, Vol. 2, pp. 127-147, 1995

[4] J. Zhang, G. Wang and G. W. Pan, "Solution of Inverse Problems in Image Processing by Wavelet Expansion", *IEEE Trans. Image Processing*, Vol. 4, No. 5, pp. 579-593, 1995.

[5] R. F. Harrington, *Field Computations by Moment Methods*, Macmillan, 1968.

[6] K. A. Dines and R. J. Lytle, "Computerized Geophysical Tomography", *Proc. IEEE*, Vol. 67, pp. 471-480, 1979.

[7] E. L. Miller and A. S. Willsky, "A Multiscale, Statistically Based Inversion Scheme for Linearized Inverse Scattering Problems", *IEEE Trans. Geoscience and Remote Sensing*, Vol. 34, No. 2, 1996.

[8] K. I. Hopcraft and P. R. Smith, *An Introduction to Electromagnetic Inverse Scattering*, Kluwer Academic, 1992.

[9] R. G. Newton, *Scattering Theory of Waves and Particles*, Springer-Verlag, 1982.

[10] J. Lekner, *Theory of Reflection of Electromagnetic and Partical Waves*, Kluwer Academic, 1987.

[11] S. Coen, K. K. Mei, and D. J. Angelakos, "Inverse scattering technique applied to remote sensing of layered media", *IEEE Trans. Antennas Propagation*, Vol. AP-29, No. 2, pp. 298-306, 1981.

[12] D. K. Ghodgonkar, O. P. Gandhi, and M. J. Hagmann, "Estimation of Complex Permittivities of Three-dimensional Inhomogeneous Biological Bodies", *IEEE Trans. Microwave Theory Tech*, Vol. MTT-31, pp. 442-446, 1983.

[13] E. L. Miller, L. Nicolaides, and A. Mandelis, "Nonlinear Inverse Scattering Methods for Thermal Wave Slice Tomography: A Wavelet Domain Approach", *J. Optical Society of America (A)*, Vol. 15, No. 6, pp. 1545-1556, 1998.

[14] G. F. Roach, *Green's Functions*, Cambridge University Press, 2nd Edition, 1982.

[15] E. Wolf, "Three-dimensional Structure Determination of Semi-transparent Objects from Holographic Data", *Optical Communications*, Vol. 1, pp. 153-156, 1969.

[16] W. H. Carter, and P. C. Ho, "Reconstruction of Inhomogeneous Scattering Objects from Holograms", *Applied Optics*, Vol. 13, pp. 162-172, 1974.

[17] D. L. Colton, *Integral Equation Methods in Scattering Theory*, New York: Wiley, 1983.

[18] A. J. Devaney, "Geophysical Diffraction Tomography", *IEEE Trans. Geoscience and Remote Sensing*, Vol. GE-22, No. 1, pp. 3-13, 1984.

[19] G. Demoment, "Image Reconstruction and Restoration: Overview of Common Estimation Structures and Problems", *IEEE Trans. Acoustics, Speech And Signal Processing.* Vol. 37. No. 12, pp. 2024-2035, 1989.

[20] G. Archer and D. M. Titterington, "On Some Bayesian/Regularization Methods for Image Restoration", *IEEE Trans. Image Processing.* Vol. 4, No. 7, pp. 989-995, 1995.

[21] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*, Washington, DC. Winston. 1977.

[22] Belge, Murat, "Multiscale and Curvature Methods for the Regularization of Linear Inverse Problems", PhD Thesis, Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, August, 1999

[23] H. W. Engl and W. Grever, "Using the l-curve for determining the optimal regularization parameters", *Numer. Math.* Vol. 69, No. 1, pp. 25-31, 1994.

[24] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Deterministic Edge-Preserving Regularization in Computed Imaging", *IEEE Trans. Image Processing.* Vol. 6, No. 2. pp, 298-311, 1997.

[25] D. Geman and C. Yang, "Nonlinear Image Recovery with Half-Quadratic Regularization", *IEEE Trans. Image Processing.* Vol. 4, No. 7, pp. 932-945, 1995.

[26] C. Bouman and K. Sauer, "A Generalized Gaussian Image Model for Edge-Preserving MAP Estimation", *IEEE Trans. Image Processing.* Vol. 2, No. 3, pp, 296-310, 1993.

[27] M. Belge, M. Kilmer and E. L. Miller, "Wavelet Domain Bayesian Image Restoration Using Edge Preserving Prior Models", 1998 International Conf. on Image Processing, Chicago Il., Oct. 1998.

[28] P. E. Gill, W. Murry and M. H. Wright, *Practical Optimization*. New York: Academic, 1981.

[29] J. A. Schnabel, "Multi-Scale Active Shape Description in Medical Imaging", Ph.D. thesis (Dissertation), Department of Computer Science, University College London, Gower Street, London. WC1E6BT, 1997.

[30] R. C. Beach, *An Introduction to The Curves and Surfaces of Computer-Aided Design*, New York: Van Nostrand Reinhold, 1991.

[31] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A practical Guide*, 2nd edition. Academic 1990.

[32] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics & Geometric Modeling*, Morgan Kaufmann, 1987.

[33] T. Cataldi, and T. Rotunno, "Kalman Filter Optimized Cubic Spline Functions for Digital Smoothing", *Analytical Methods and Instrumentation*, Vol. 2, No. 1, pp. 27-34, 1995.

[34] Voskoboinikov, and Y. Ye, "Descriptive splines in Inverse Heat Conduction Problems: Method and Algorithms", *J. Engineering Physics and Thermophysics*, Vol. 65, No. 6, pp. 1230-1253, 1993.

[35] P. Dierckx, *Curve and Surface Fitting with Splines*, New York: Oxford University Press, 1993.

[36] C. de Boor, "On Calculation of B-Splines", *J. Approx. Theory*, Vol. 6, pp. 50-62, 1972.

[37] W. Boehm, and H. Prautzsch, "The Insertion Algorithm", *Computer-Aided Design*, Vol. 17, No. 2, 1985.

[38] M. Eck, and J. Hadenfeld, "Knot removal for B-Spline Curves", *Computer Aided Geometric Design*, Vol. 12, No. 3, pp. 259-272, 1995.

[39] R. N. Goldman, and T. Lyche, "Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces", *SIAM*, 1993.

[40] T. Cham, and R. Cipolla, "Automated B-Spline Curve Representation Incorporating MDL and Error-Minimizing Control Point Insertion Strategies", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 21, No. 1, pp. 49-53, 1999.

[41] G. Farin, and N. Sapidis, "Curvaure and The Fairness of Curves and Surfaces", *IEEE Computer Graphics and Applications*, Vo. 9, pp. 53-57, 1989.

[42] D. Nira, and Y. Itai, "Optimal Distribution of Knots for Tensor-Product Spline Approximation", *Quarterly of Applied Mathematics*, Vol. 49, No. 1, pp. 19-36, 1991.

[43] N. Sapidis, and G. Farin, "Automatic Fairing Algorithm for B-Spline Curves", *Computer Aided Design*, Vol. 22, pp. 121-129, 1990

[44] A. Davies, and P. Samuels, *An Introduction to Computational Geometry for Curves and Surfaces*. New York: Oxford University Press. 1996.

[45] H. Flanders, *Differential Forms with Applications to the Physical Sciences*, New York: Dover Publications, 1989.

[46] H. Radha, R. Leonardi, M. Vetterli, and B. Naylor, "Binary Space Partitioning Tree Representation of Images", *J. Visual Communication and Image Representation.* Vol. 2, No. 3, pp. 202-221, 1991.

[47] G. Hunter and K. Steiglitz, "Operations on Images Using Quad Trees", *IEEE Trans. Pattern Analasys and Machine Intelligence*, Vol. PAMI-1, No. 2, pp. 145-153, 1979.

[48] W. Zhu, Y. Wang, Y. Deng, Y. Yao and R. L. Barbour, "A Wavelet-Based Multiresolution Regularized Least Squares Reconstruction Approach for Optical Tomography", *IEEE Trans. Medical Imaging.* Vol. 16, No. 2, pp. 210-217, 1997.

[49] M. Eck, T. deRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes", *Computer Graphics(SIGGRAPH'95 Proceedings)*, pp. 173-182, 1995.

[50] W. Welch, and A. Witkin, "Free-form Shape Design Using Triangulated Surfaces", *Computer Graphics(SIGGRAPH'94 Proceedings)*, pp. 247-256, 1994.

[51] E. Horowitz, S. Sahni, and S. Rajasekaran, *Computer Algorithm*, New York: Computer Science Press, 1998.

Appendix A

Derivation of CRB

In our cross-well tomography problem, we set up the following methods:

1. Tikhonov method:

$$G(f) = \frac{1}{\sigma^2}\|y - Af\|_2^2 + \lambda_x^2\|D_x f\|_2^2 + \lambda_y^2\|D_y f\|_2^2 \tag{A.1}$$

2. Edge preserving method:

$$G(f) = \frac{1}{\sigma^2}\|y - Af\|_2^2 + \lambda_x^2\Phi_x^T(f)\Phi_x(f) + \lambda_y^2\Phi_y^T(f)\Phi_y(f) \tag{A.2}$$

It will be helpful to learn the nature of the problem if we can find the Cramer-Rao Bound(CRB) evaluated for the given object f, which represent the estimation variance of data for different cross-well regions.

For this problem we can find the CRB by the following way,

Write G(f) as:

$$G(f) = e^T(f)e(f) \tag{A.3}$$

and Let

$$J = \frac{\partial e(f)}{\partial f} \tag{A.4}$$

Note here f is a column vector, and the partial derivative is defined as:

$$[J]_{i,j} = \frac{\partial e_j}{\partial f_i} \tag{A.5}$$

Thus the Fisher's Information is:

$$F = J^T * J \tag{A.6}$$

and CRB can be found by:

$$CRB = F^{-1} = (J^T * J)^{-1} \tag{A.7}$$

A.1 Tikhonov Method

In this case, it's easily to see that:

$$e(f) = [\frac{1}{\sigma}(y - Af),\ \lambda_x D_x f,\ \lambda_y D_y f]^T \tag{A.8}$$

Where D_x and D_y are derivative matrix such that:

$$Dx(f) = D_x * f = f_{i+1,j} - f_{i,j} \tag{A.9}$$

$$Dy(f) = D_y * f = f_{i,j+1} - f_{i,j} \tag{A.10}$$

Here, simply use the definition of derivative of matrices and vectors:

$$\frac{d}{dx}Ax = A^T \tag{A.11}$$

$$\frac{d}{dx}x^T A = A \tag{A.12}$$

where, A is matrix and x is vector.

We can get:

$$J = \frac{\partial e}{\partial f} = \begin{bmatrix} \frac{1}{\sigma}A \\ \lambda_x D_x \\ \lambda_y D_y \end{bmatrix} \tag{A.13}$$

and,

$$CRB = (J^T * J)^{-1} \tag{A.14}$$

A.2 Edge Preserving Method

In this, we have,

$$e(f) = [\frac{1}{\sigma}(y - Af),\ \lambda_x \Phi(D_x f),\ \lambda_y \Phi(D_y f)]^T \tag{A.15}$$

where,

$$\Phi^T(g)\Phi(g) = \frac{g^2}{1+g^2} \tag{A.16}$$

$$\Phi(g) = \frac{g}{\sqrt{1+g^2}} \tag{A.17}$$

Thus, we can compute the derivative of e(f) by using the following equation:

$$[\frac{\partial \Phi}{\partial f}]_{i,j} = \frac{\partial \Phi_j}{\partial f_i} = \sum_k \frac{\partial \Phi_j}{\partial g_k}\frac{\partial g_k}{\partial f_i} \tag{A.18}$$

First, compute $\frac{\partial \Phi}{\partial g}$, where $g = D_x f$ or $g = D_y f$:

$$\frac{\partial \Phi_j}{\partial g_k} = \frac{\partial}{\partial g_k}(\frac{g_j}{\sqrt{1+g_j^2}}) = (1+g_j^2)^{-\frac{3}{2}}\delta(j-k) \tag{A.19}$$

Second,

$$\frac{\partial D_x f}{\partial f} = D_x^T \tag{A.20}$$

$$\frac{\partial D_y f}{\partial f} = D_y^T \tag{A.21}$$

Therefore, we get,

$$[\frac{\partial \Phi}{\partial f}]_{i,j} = \frac{\partial \Phi_j}{\partial f_i} = \sum_k \frac{1}{(1+g_j^2)^{\frac{3}{2}}} \delta(j-k)[D_x]_{k,i} = \frac{1}{(1+g_j^2)^{\frac{3}{2}}}[D_x]_{j,i} \tag{A.22}$$

Write it in matrix form, we get:

$$\frac{\partial \Phi_x}{\partial f} = G_x * D_x, \ \frac{\partial \Phi_y}{\partial f} = G_y * D_y \tag{A.23}$$

where

$$G_x = diag([1+(D_x f)_i^2]^{-\frac{3}{2}}), \ G_y = diag([1+(D_x f)_i^2]^{-\frac{3}{2}}) \tag{A.24}$$

Finally, we get:

$$J = \frac{\partial e}{\partial f} = \begin{bmatrix} \frac{1}{\sigma} A, \\ \lambda_x G_x D_x, \\ \lambda_y G_y D_y \end{bmatrix} \tag{A.25}$$

and

$$CRB = (J^T * J)^{-1} \tag{A.26}$$