Inverse Problem Study in One Order ODEs

Wenchao Zhang,Qikun Wu, Jiale Wang, Luo Tao
South University of Science and Technology of China

November 29, 2012

Contents

Abstract

We explore methods of Discritizing Inverse Problems, Tikhonov Regularization, Iterative method, Interpolation and real application using MATLAB.

1 Introduction

Inverse problems are always fascinating since the solution to it is not always unique: it may differ under different circumstance. Our question to inverse problem is:[**?**]

1. According to the output can we determine at least a model (existence)?
2. How can we obtain parameters from such model (approximation)?
3. Is the model existing the only one model (uniqueness)?
4. Can the model endure little error of the output (stability)?

As we can see, so many questions we will face and we cannot solve all the questions. So we are going to explore the existence, approximation of these inverse problem while having a glimpse on the uniqueness and stability.

1.1 The Work in Models

Usually, the scientific procedure for the study of a physical system can be divided into the following three steps[**?**]:

I. Parameterization of the system : discovery of a minimal set of model parameters whose values completely characterize the system.

II. Forward modeling: To find the physical laws allowing us by construction with given values of the model parameters, and make predictions on the results of measurements on some observable parameters.

III. Inverse modeling : use of the actual results of some measurements of the observable parameters to infer the actual values of the model parameters.

The first two steps are mainly inductive,while the third one is deductive, so it is hard to do the first two steps.On the contrary, the mathematical theory of logic seems to apply quite well to the third step.

1.2 Model Space and Data Space

The relate physical parameters always characterize a model,$m \in M$, and collected observations also make up some set of data,$d \in D$. Suppose we have found a model which satisfies the following equation

$$\mathbf{d} = \mathbf{G}\left(\boldsymbol{m}\right) \tag{1.1}$$

In practice, d may be a function of time and/or space, or may be a collection of discrete observations. And In facts,actual observations always contain some amount of noise which we have to deal with.We can thus envision data as generally consisting of noiseless observations from a "perfect" experiment, $\boldsymbol{d}_{\mathrm{true}}$, plus a noise component η, i.e.

$$\boldsymbol{d} = \boldsymbol{G}\left(\boldsymbol{m}_{\mathrm{true}}\right) + \eta = \boldsymbol{d}_{\mathrm{true}} + \eta \tag{1.2}$$

where $\boldsymbol{d}_{\mathrm{true}}$ exactly satisfies for $\boldsymbol{m}$ equal to the true model,$\boldsymbol{m}_{\mathrm{true}}$, and we assume that the forward modeling is exact. When $\boldsymbol{m}$ and $\boldsymbol{d}$ are functions, we typically refer to $\boldsymbol{G}$ as an operator.

1.3 Linear Systems

Definition 1 *Let $\boldsymbol{d}$ and $\boldsymbol{m}$ be connected the equation* **??**. *A linear system $\boldsymbol{G}(\boldsymbol{m})$ satisfies following two conditions:*

I. $\boldsymbol{G}(\boldsymbol{m}_1 + \boldsymbol{m}_2) = \boldsymbol{G}\left(\boldsymbol{m}_1\right) + \boldsymbol{G}\left(\boldsymbol{m}_2\right)$ *for all* m_1,$m_2 \in M$

II. $\boldsymbol{G}(\alpha\boldsymbol{m}) = \alpha\boldsymbol{G}(\boldsymbol{m})$

1.3.1 Fredholm Integral Equations of the First Kind

In a continuous linear inverse problem, G can often be expressed as a linear integral operator, where (1.1) has the form

$$\int_a^b g(s,x)m(x)\,\mathrm{dx} = d(s) \tag{1.3}$$

and the function $g(s,x)$ is called the kernel. The linearity of (1.3) is easily seen. Equations in the form of (1.3), where $m(x)$is the unknown, are called **Fredholm integral equations of the first kind (IFKs)**

1.4 Well-posed Problems

Hadamard is the first one who raise the idea of Well-posedness.In the sense of Hadamard,essential issues of a problem that must be considered include solution existence, solution uniqueness, and instability of the solution process[**?**].

I. **Existence**. There may be no model that exactly fits the data. This can occur in practice because our mathematical model of the system's physics is approximate or because the data contain noise.

II. **Uniqueness**. If exact solutions do exist, they may not be unique, even for an infinite number of exact data points. That is, there may be other solutions besides m_{ture} that exactly satisfy $G(m) = d_{\mathrm{ture}}$. This situation commonly occurs in potential field problems.

III. **Stability**. The process of computing an inverse solution can be, and often is, extremely unstable in that a small change in measurement can lead to an enormous change in the estimated model.

Here,we give a definition of well-posedness:

Definition 2 *Let$X$and$Y$be normed spaces, $K : X \to Y$ a mapping. The equation$Kx = y$ is called properly posed or well-posed if:*

I. ***Existence***. *For every $y \in Y$ there exist $x \in X$ such that $Kx = y$.*

II. ***Uniqueness***. *For every $y \in Y$ there is only one $x \in X$ such that $Kx = y$.*

III. ***Stability***. *The solution depends continuously on y, i.e., for every sequence $(x_n) \subset X$ with $Kx_n \to Kx$, it follows that $x_n \to x(n \to \infty)$*

Equations which cannot satisfy all three properties are called ill-posed.

1.5 Idea of Regularization

Inverse problems where this situation arises are referred to as ill-posed in the case of continuous systems, or ill-conditioned in the case of discrete linear systems.
A key point is that it is commonly possible to stabilize the inversion process by imposing additional constraints that bias the solution, a process that is generally referred to as regularization. Regularization is frequently essential to producing a usable solution to an otherwise intractable ill-posed or ill-conditioned inverse problem.

2 Discretizing Continuous Inverse Problems

2.1 Quadrature Methods

To obtain useful numerical solutions to IFKs, we will frequently seek to discretize them into forms that are tractably solvable using linear algebra methods. We first assume that d(s) is known at a finite number of points s_1, s_2,...,s_m.For a finite number of data points we can write the inverse problem as

$$d_i = d(s_i) = \int_a^b g(s_i,x)m(x)\,\mathrm{dx} = \int_a^b g_i(x)m(x)\,\mathrm{dx} \tag{2.4}$$

The functions $g_i(x)$ are referred to as representers or data kernels.

The simplest quadrature rule is themidpoint rule , where we divide the interval $[a, b]$ into n subintervals and pick points x_1, x_2,...,x_n in the middle of each subinterval. The points are given by

$$x_j = a + \frac{\Delta x}{2} + (j-1)\Delta x, \text{where}\, \Delta x = \frac{b-a}{n}$$

Thus,we have an approximate formula:

$$d_i \approx \sum_{j=1}^{n} g_i(x_j) m(x_j) \Delta x, i = 1, 2, 3, ..., m \tag{2.5}$$

2.2 Expansion in Terms of Representers

2.2.1 Procedures of Expansion Steps

In the Gram matrix technique for discretizing a linear inverse problem, a continuous model $m(x)$ is written as a linear combination of the m represents equation **??**.

$$m(x) = \sum_{j=1}^{m} \alpha_j g_j(x) \tag{2.6}$$

where the α_j are coefficients to be determined. The representers form a basis for a subspace of the space of all functions on the interval(a, b). Substituting equation **??** into equation **??** gives

$$d(s_i) = \int_a^b g_i(x) \sum_{j=1}^{m} \alpha_j g_j(x)\, \mathrm{dx} = \sum_{j=1}^{m} \alpha_j \int_a^b g_i(x)\, g_j(x)\, \mathrm{dx}, \text{for}\, i = 1, 2, \ldots, m \tag{2.7}$$

2.2.2 Gram Matrix

Gram matrix $\mathbf{\Gamma}$ with elements as following:

$$\Gamma_{i,j} = \int_a^b g_i(x) g_j(x)\, \mathrm{dx}, \text{for}\, i = 1, 2, ..., m \tag{2.8}$$

The **IFK** can thus be discretized as an $m \times m$ linear system of equations:

$$\mathbf{\Gamma}\alpha = \boldsymbol{d} \tag{2.9}$$

Once we solve above equation,then we will get $m(x)$ by equation **??**.
The Gram matrix tends to become very badly conditioned as m increases[**?**]. On the other hand, we want to use as large as possible a value of m so as to increase the accuracy of the discretization. Thus there is a trade-off between the discretization error and ill-conditioning.

2.3 Expansion in Terms of Orthonormal Basic Function

Generalizing last approach, suppose we are given suitable functions $h_1(x), h_2(x), ..., h_n(x)$ that form a basis for a function space **H** . We could then approximate m(x) by

$$m(x) = \sum_{j=1}^{n} \alpha_j h_j(x) \tag{2.10}$$

Substituting this approximation into equation **??** gives

$$d(s_i) = \int_a^b g_i(x) \sum_{j=1}^{n} \alpha_j h_j(x)\, \mathrm{dx} = \sum_{j=1}^{n} \alpha_j \int_a^b g_i(x) h_j(x)\, \mathrm{dx}, \text{for}\, i = 1, 2, ..., m \tag{2.11}$$

This leads to an m by n linear system

$$\boldsymbol{G\alpha} = \boldsymbol{d} \tag{2.12}$$

Where

$$\boldsymbol{G}_{kl} = \int_a^b g_k(x) h_l(x) dx, \boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots \alpha_n)^T, \boldsymbol{d} = (d(s_1), d(s_2), \cdots d(s_n))^T$$

Definition 3 *(dot product) We define the dot product or inner product of two functions to be*

$$f \cdot g = \int_a^b f(x)g(x)\,\mathrm{dx}$$

If the basis functions $h_j(x)$ are orthonormal with respect to this inner product, then the projection of $g_i(x)$ onto the space **H** spanned by the basis is

$$\mathrm{proj}_H\, g_i(x) = (g_i \cdot h_1)h_1(x) + (g_i \cdot h_2)h_2(x) + \cdots + (g_i \cdot h_n)h_n(x)$$

The elements in the G matrix are given by the same dot products

$$G_{i,j} = g_i \cdot h_j$$

Thus we have effectively projected the original representers onto our function space **H**. An important advantage of using an orthonormal basis is that it can be shown that$\|m\,(x)\|_2 = \|\alpha\|_2$.

2.4 The Method of Backus and Gilbert

The method of Backus and Gilbert[**?**] is applicable to continuous linear inverse problems of the IFKs.We want to estimate $m(x)$ at some point $\hat{x}$ given the m data values d_j . Since the only data that we have are the d_j values, we will consider estimates of the form

$$m(\hat{x}) \approx \hat{m} = \sum_{j=1}^{m} c_j d_j \tag{2.13}$$

where the c_j are coefficients to be determined.
Combining equation **??** equation **??** gives

$$\{\hat{m}\} = \sum_{j=1}^{m} c_j \int_a^b g_j(x)m(x)\,\mathrm{dx} = \int_a^b \left(\sum_{j=1}^{m} c_j g_j(x)\right) m(x)\,\mathrm{dx} = \int_a^b A(x)m(x)\,\mathrm{dx} \tag{2.14}$$

where $A(x) = \sum_{j=1}^{m} c_j g_j(x)$,and the function$A(x)$ is called an averaging kernel.

Ideally, we would like the averaging kernel to closely approximate a delta function $A(x) = \delta(x-\hat{x})$,because, assuming the data were exact, equation **??** would then produce exact agreement $(\hat{m} = m(\hat{x}))$.between the estimated and the true model.
Since this is not possible, we will instead select the coefficients so that the area under the averaging kernel is one, and so that the width of the averaging kernel around the $\hat{x}$ is as small as possible.
Then,$\int_a^b A(x)\,\mathrm{dx} = 1$.Let $q_j = \int_a^b g_j(x)\,\mathrm{dx}$.Hence, we get $\boldsymbol{q}^T c = 1$.
More over,the problem of finding the optimal coefficients can be written as

$$\min\{c^T \boldsymbol{H} c, \boldsymbol{q}^T c\} = 1 \tag{2.15}$$

$$\sum_{j=1}^{m} c_j^2 \sigma_j^2 \leq \Delta \tag{2.16}$$

where σ_j is the standard deviation of the j-th observation.Smaller values of Δ decrease the variance of the estimate but restrict the choice of coefficients so that the width of the averaging kernel increases. There is also a trade-off between stability of the solution and the width of the averaging kernel.

3 The SVD, Generalized Inverse and best approximation

3.1 The SVD

A method of analyzing and solving least squares problems that is of particular interest in ill-conditioned and/or rank-deficient systems is the singular value decomposition ,or SVD.In the SVD[**?**] an m by n matrix G is factored into

$$G = \mathrm{USV}^T$$

where

I. U is an m by m orthogonal matrix with columns that are unit basis vectors spanning the data space, R^m.

II. V is an n by n orthogonal matrix with columns that are basis vectors spanning the model space, R^n.

III. S is an m by n diagonal matrix with nonnegative diagonal elements called singular values, which is given by following definition

Definition 4 *(singular values)*
G is a $m \times n$ matric. $G^H G$ has eigenvalues of $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r \geq \lambda_{r+1} = \cdots = \lambda_n = 0$, then $\sigma_i = \sqrt{\lambda_i}, i = 1, 2, \cdots r$ are called the singular values of G

The **SVD** matrices can be computed in MATLAB with the svd command. It can be shown that every matrix has a singular value decomposition[**?**].

A SVD decomposition is illustrated as follows, where $A = USV^T$

```
>> A=reshape(1:12,4,3)
A =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
>> [U,S,V]=svd(A)
U =
   -0.4036   -0.7329   -0.2819   -0.4696
   -0.4647   -0.2898    0.7259    0.4160
   -0.5259    0.1532   -0.6060    0.5768
   -0.5870    0.5962    0.1620   -0.5232
S =
   25.4368         0         0
         0    1.7226         0
         0         0    0.0000
         0         0         0
V =
   -0.2067    0.8892   -0.4082
   -0.5183    0.2544    0.8165
   -0.8298   -0.3804   -0.4082
>> U*S*V'
ans =
    1.0000    5.0000    9.0000
    2.0000    6.0000   10.0000
    3.0000    7.0000   11.0000
    4.0000    8.0000   12.0000
```

Moreover, SVD can solve the least square solution of $Gx = b$. The process is given below

I. Decompose $G = \mathrm{USV}^T$

II. The least square solution to $Gx = b$ is identical to the least square solution to $\mathrm{SV}^T x = \mathrm{U}^H b$

III. Let $y = \mathrm{V}^T x, c = \mathrm{U}^H b$. Then the equation becomes $\mathrm{S}\, y = c$. Since S is a $m \times n$ diagonal matrix with nonnegative diagonal elements, the solution to y is obvious:

$$y = \left(\frac{c_1}{\sigma_1}, \frac{c_2}{\sigma_2} \cdots \frac{c_r}{\sigma_r}, 0, \cdots 0 \right)^T$$

since $y = \mathrm{V}^T x$, then the least square solution is

$$x = \left(\mathrm{V}^T\right)^H y \tag{3.17}$$

3.2 Moore-Penrose Pseudoinverse

The singular values along the diagonal of S are customarily arranged in decreasing size.Note that some of the singular values may be zero. If only the first p singular values are nonzero, we can partition **S** as

$$\begin{pmatrix} S_p & 0 \\ 0 & 0 \end{pmatrix}$$

where S_p is a p by p diagonal matrix composed of the positive singular values.
Expanding the SVD representation of G in terms of the columns of U and V gives

$$G = (U_p, U_0) \begin{pmatrix} S_p & 0 \\ 0 & 0 \end{pmatrix} (V_p, V_0)^T = U_p S_p V_p^T \tag{3.18}$$

The **SVD** can be used to compute a generalized inverse of G, called the **Moore–Penrose pseudoinverse**.

Definition 5 $\boldsymbol{G^\dagger = V_p S_p^{-1} U_p^T}$ *is the generalized inverse of G in the sense of Moore and Penrose.*

MATLAB has a `pinv` command that generates $G^\dagger$.Using Definition 5, we define the pseudoinverse solution to be

$$\boldsymbol{m}_\dagger = G^\dagger \boldsymbol{d} = V_p S_p^{-1} U_p^T \boldsymbol{d} \tag{3.19}$$

4 Tikhonov Regularization

4.1 Origins of Tikhonov Regularization

For a general linear least squares problem there may be infinitely many least squares solutions.If we consider that the data contain noise, and that there is no point in fitting such noise exactly,it becomes evident that there can be many solutions that adequately fit the data in the sense that $||\boldsymbol{Gm} - \boldsymbol{d}||_2$ is small enough.
In Tikhonov regularization, we consider all solutions with$||\boldsymbol{Gm} - \boldsymbol{d}||_2 \leqslant \delta$,and select the one that minimizes the norm of **m**:

$$\begin{array}{l} ||\boldsymbol{Gm} - \boldsymbol{d}||_2 \leqslant \delta \\ \min \|\boldsymbol{m}\|_2 \end{array} \tag{4.20}$$

the minimization of $\|\boldsymbol{m}\|_2$ should ensure that unneeded features will not appear in the regularized solution. Note that as δ increases, the set of feasible models expands, and the minimum value of $\|\boldsymbol{m}\|_2$ decreases. We can thus trace out a curve of minimum values of $\|\boldsymbol{m}\|_2$ versus δ.
We have got conditions condition **??** .It is also possible to trace out this curve by considering problems of the form:

$$\begin{array}{l} \min ||\boldsymbol{Gm} - \boldsymbol{d}||_2 \\ \|\boldsymbol{m}\|_2 \leqslant \varepsilon \end{array} \tag{4.21}$$

As ε decreases, the set of feasible solutions becomes smaller, and the minimum value of$||\boldsymbol{Gm} - \boldsymbol{d}||_2$ increases.
So to sum up above two situation,our final option is to consider the damped least squares problem

$$\min \left(J_\alpha \left(m\right)\right) = \min(||\boldsymbol{Gm} - \boldsymbol{d}||_2^2 + \alpha^2 \|\boldsymbol{m}\|_2^2) \tag{4.22}$$

which arises when we apply the method of Lagrange multipliers to condition **??**, where α is a regularization parameter.It can be shown that for appropriate choices of δ ,ε, and α ,the three methods (condition **??**, **??** and **??**)yield the same solution.

4.2 The Method of Tikhonov Regularization

Maybe it's easy to think as above direction, but we have got less information about it, i.e, now we should find a proper m which satisfies formula **??**.
Now we recall the first equation **??**, as we have done,if the system is discrete,we can rewrite it as an equation of the form

$$Gm = d \tag{4.23}$$

where G is a compact linear operator from a Hilbert space H_1 into a Hilbert space H_2.

For we can not always find the inverse of G,so the generally equation **??** does not have a unique solution,therefore we seek a particular generalized solution,namely the least square solution of minimum norm. Changed form haven't dealt with trivial case, our question comes to be a well-posed question until we assume that $d \in \mathcal{D}\left(G^{\dagger}\right)$ and find the approximation of $G^{\dagger}d$.
The generalized solution $m = G^{\dagger}d$ of equation **??** is a least squares solution and therefore it satisfies the normal equations

$$G^*Gm = G^*d \tag{4.24}$$

where G^* is the adjoint of G.Now the self-adjoint operator G^*G has nonnegative eigenvalues and therefore for none zero number α,the operator $G^*G + \alpha^2 I$,where I is identity operator on H_1,has strictly positive eigenvalues.In particular,the operator $G^*G + \alpha^2 I$ has a bounded inverse, that is, the problem of solving the equation

$$\left(G^*G + \alpha^2 I\right) m_\alpha = G^*d \tag{4.25}$$

is well-posed.The second kond of equation **??** is called a regualarized form of equation **??** and its unique solution

$$m_\alpha = \left(G^*G + \alpha^2 I\right)^{-1} G^*d \tag{4.26}$$

is called the Tikhonov approximation to $G^{\dagger}y$,the minimun norm solution of equation **??**.
This can be accomplished conveniently in terms of a singular system $\{v_j, u_j; \mu_j\}$ for $G.m_\alpha$ can be written as following,

$$m_\alpha = \sum_{j=1}^{\infty} \frac{\mu_j^2}{\mu_j^2 + \alpha^2} \langle d, u_j \rangle v_j \tag{4.27}$$

The true minimum norm least squares solution is ,according to the following equation[**?**]:

$$G^{\dagger}d = \sum_{j=1}^{\infty} \frac{1}{\mu_j} \langle d, u_j \rangle v_j \tag{4.28}$$

The following theorem demostrates the solution of equation **??** satisfying condition **??**.

Theorem 6 *Let G:$\boldsymbol{d} \longrightarrow \boldsymbol{m}$ be a linear and bounded operator between Hilbert spaces and $\alpha \neq 0$.Then the Tikhonov functional J_α has a unique minimum $m_\alpha \in \boldsymbol{m}$.This minimum m_α is the unique solution of the normal equation*

$$\left(G^*G + \alpha^2 I\right) m_\alpha = G^*d \tag{4.29}$$

4.3 Determination of Regularization Parameter α

First of all, we suppose there is no error in our observed data d. Then the ill-poseness is becuase G has no inverse.For this case, we will lost some information of estimation. We'd better assume continuity or some other properties to find out the solution of singular points. This case, however, we haven't seen in our facing question, so we don't want to research it.

The second case,which is our most important and difficult one,is that we have some data which have errors. For this case,we know that this question is ill-posed, for it doesn't satisfy stability condition whatever.
We have some conclusions in this case.
The best we can hope for is some estimate d^δ of d satisfying

$$\left\|d^\delta - d\right\| \leqslant \delta \tag{4.30}$$

where δ is a known bound on the measurement error.Then we have

$$\left\|m_\alpha^\delta - m_\alpha\right\| \leqslant \frac{\delta}{\alpha} \tag{4.31}$$

According to condition **??**, any a priori choice $\alpha = \alpha(\delta)$ of the regularization parameter satisfying $\frac{\delta^2}{\alpha^2(\delta)} \longrightarrow 0$ as $\delta \longrightarrow 0$ leads to a regular algorithm for the solution of $Gm = d$. Although this asymptotic result may be theoretically satisfying, it would seem that a choice of the regularization parameter that is based on the actual computations performed, that is, an a posteriori choice of the regularization parameter would be more effective in practice.
There are many ways to find out regularization parameter α, We have some useful methods as below.

4.3.1 Discrepancy Principle of Morozov

One such a posteriori strategy is the discrepancy principle of Morozov. The idea of the strategy is to choose the regularization parameter so that the size of the residual $||Gm_\alpha^\delta - d^\delta||$ is the same as the error level in the data:

$$\left\|Gm_\alpha^\delta - d^\delta\right\| = \delta \tag{4.32}$$

4.3.2 Engl's Criterien

Engl's idea[**?**] is to find an α which make the following formula minimum.

$$\varphi(\alpha) = \frac{\left\|Gm_\alpha^\delta - d^\delta\right\|}{\alpha} \tag{4.33}$$

4.3.3 Arcangeli Criterien

Arcangeli Criterien[**?**]is that αsatisfying

$$\left\|Gm_\alpha^\delta - d^\delta\right\| - \frac{\delta}{\alpha} = 0 \tag{4.34}$$

4.3.4 Tikhonov Criterien

Tikhonov claims[**?**] that α has optimal value satisfying

$$\alpha_{\text{opt}} = \min_{\alpha \neq 0} \left\{ \left\| \alpha \frac{dm_\alpha}{d\alpha} \right\| \right\} \tag{4.35}$$

4.3.5 Hanke's L-Curves

Let $\rho = \log \|Gm_\alpha - d\|, \theta = \log \|m_\alpha\|$,then the curvature of $\rho - \theta$ curve is defined

$$c(\alpha) = \frac{\rho'\theta'' - \rho''\theta'}{\left((\rho')^2 + (\theta')^2\right)^{3/2}} \tag{4.36}$$

Hanke uses the αwhich has maximum $c(\alpha)$ as the optimal one[**?**].

5 Iterative Methods

5.1 Introduction

SVD-based pseudoinverse and Tikhonov regularization solutions become impractical when we consider larger problems in which G has thousands of rows and columns. Storing all of the elements in a large G matrix can require a great deal of memory.
If the majority of the elements in the G matrix are 0, then G is a sparse matrix , and we can save storage by only storing the nonzero elements of G and their locations. The density of G is the percentage of nonzero elements in the matrix. Dense matrices contain enough nonzero elements that sparse storage schemes are not efficient.

5.2 Iterative Methods

5.2.1 Kaczmarz's Algorithm

Algorithm 1

Given a system of equations $\boldsymbol{Gm} = \boldsymbol{d}$.

1. Let $\boldsymbol{m}^{(0)} = \mathbf{0}$.
2. For $i = 0, 1, ..., m$,let

$$m^{(i+1)} = m^{(i)} - \frac{G_{i+1,\cdot} m^{(i)} - d_{i+1}}{G_{i+1,\cdot} G_{i+1,\cdot}^T} G_{i+1,\cdot}$$

where $G_{i+1,\cdot}$ be the $i+1$-th row of G

3. If the solution has not yet converged, go back to step 2.

5.2.2 Algebraic Reconstruction Technique(ART)

Algorithm 2

Given a system of equations $\boldsymbol{Gm} = \boldsymbol{d}$ arising from a tomography problem.

1. Let $\boldsymbol{m}^{(0)} = \mathbf{0}$.
2. For $i = 0, 1, ..., m$,let N_i be the number of cells touched by ray path i.
3. For $i = 0, 1, ..., m$,let L_i be the length of ray path i.
4. For $i = 0, 1, ..., m-1$, $j = 1, 2, \ldots, n$,let

$$m_j^{(i+1)} = \begin{cases} m_j^{(i)} + \frac{d_{i+1}}{L_{i+1}} - \frac{q_{i+1}}{\mathrm{lN}_{i+1}} & \text{cell } j \text{ in ray path } i+1 \\ m_j^{(i)} & \text{cell } j \text{ not in ray path } i+1 \end{cases} \tag{5.37}$$

5. If the solution has not yet converged, let $\boldsymbol{m}^{(0)} = \boldsymbol{m}^{(m)}$and go back to step 4.Otherwise, return the solution $\boldsymbol{m} = \boldsymbol{m}^{(m)}$.

5.2.3 Selective Internal Radiation Therapy(SIRT)

Algorithm 3

Given a system of equations $\boldsymbol{Gm} = \boldsymbol{d}$ arising from a tomography problem.

1. Let $\boldsymbol{m}^{(0)} = \mathbf{0}$.
2. For $j = 0, 1, ..., n$,let K_i be the number of ray path that pass through cell i.
3. For $i = 0, 1, ..., m$,let L_i be the length of ray path i.
4. For $i = 0, 1, ..., m$,let N_i the number of cells touched by ray path i.
5. let $\Delta\boldsymbol{m} = \mathbf{0}$.
6. For $i = 0, 1, ..., m-1$, $j = 1, 2, \ldots, n$,let

$$\Delta m_j = \Delta m_j + \begin{cases} \frac{d_{i+1}}{L_{i+1}} - \frac{q_{i+1}}{\mathrm{lN}_{i+1}} & \text{cell } j \text{ in ray path } i+1 \\ 0 & \text{cell } j \text{ not in ray path } i+1 \end{cases} \tag{5.38}$$

7. For $j = 1, 2, ...n$,let

$$m_j = m_j + \frac{\Delta m_j}{K_j} \tag{5.39}$$

8. If the solution has not yet converged, go back to step 5. Otherwise, return the current solution.

5.2.4 The Conjugate Gradient(CG) Method

Algorithm 4

Given a positive definite and symmetric system of equations $\mathrm{Ax} = b$, and an initial solution x_0 , let $\beta_0 = 0$, p_{-1}=0, r_0=$b - Ax_0$, and $k = 0$. Repeat the following steps until convergence.

1. If $k > 0$, let$\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$.
2. Let $p_k = r_k + \beta_k r_{k-1}$.
3. Let $\alpha_k = \frac{r_k^T r_k}{p_k^T \mathrm{Ap}_k}$.
4. Let $x_{k+1} = x_k + \alpha_k p_k$.
5. Let $r_{k+1} = r_k - \alpha_k \mathrm{Ap}_k$.
6. let $k = k + 1$.

5.2.5 Conjugate Gradient Least Squares (CGLS) Method

Algorithm 5

Given a positive definite and symmetric system of equations $\boldsymbol{Gm} = \boldsymbol{d}$, and an initial solution x_0 , let $\beta_0 = 0$, p_{-1}=0,$\boldsymbol{m}_0 = \boldsymbol{0}$, $\boldsymbol{s}_0 = \boldsymbol{d}$, $\boldsymbol{r}_0$=$G^T \boldsymbol{s}_0$, and $k = 0$. Repeat the following steps until convergence.

1. If $k > 0$, let$\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}$.
2. Let $p_k = r_k + \beta_k r_{k-1}$.
3. Let $\alpha_k = \frac{r_k^T r_k}{(\mathrm{G}\mathrm{p}_k)^T (\mathrm{G}\mathrm{p}_k)}$.
4. Let $x_{k+1} = x_k + \alpha_k p_k$.
5. Let $r_{k+1} = r_k - \alpha_k \mathrm{G}\mathrm{p}_k$.
6. let $k = k + 1$.

6 Applications in Our Project

we suppose that x is average distributed i.e. for each x_i,we have $x_{i+1} - x_i = \Delta x$.

6.1 Given P(x), Find Q(x),and without Input Errors

Let $e^{\int P(x)\,\mathrm{dx}} = m\,(x)$,then for the linear first order ode,we have

$$m\,(x)\,y\,(x) = \int_{x_0}^{x} m\,(x)\,Q\,(x)\,\mathrm{dx} \tag{6.40}$$

Equation (39) is an **Volterra Integral Equations of the First Kind**. We can change it as a **Fredholm Integral Equations of the First Kind**.
Let

$$\hat{m}\,(x, x_i) \quad = \quad \begin{cases} m\,(x) & x \leqslant x_i \\ 0 & x > x_i \end{cases} \tag{6.41}$$

Then,we have

$$m\,(x_i)\,y\,(x_i) = \int_{x_0}^{x_n} \hat{m}\,(x, x_i)\,Q\,(x)\,dx \tag{6.42}$$

we use quadrature methods to discretize above system,

$$m\,(x_i)\,y\,(x_i) = \sum_{j=1}^{n} \hat{m}\,(x_j, x_i)\,Q\,(x_j)\,\Delta x, i = 1, 2, \ldots, n \tag{6.43}$$

As we can see, we can find that operator $G$is non-sigular,and have a inverse,for there's no error,so it is well-posed.
In fact,we can write it as matrix

$$\begin{pmatrix} z\,(x_1) \\ z\,(x_2) \\ \cdots \\ \cdots \\ \cdots \\ z\,(x_n) \end{pmatrix} = \begin{pmatrix} m\,(x_1) & 0 & \ldots & \ldots & \ldots & 0 \\ m\,(x_1) & m\,(x_2) & 0 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & 0 & \ldots & \ldots \\ \ldots & \ldots & \ldots & m\,(x_i) & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ m\,(x_1) & m\,(x_2) & \ldots & \ldots & m\,(x_{n-1}) & m\,(x_n) \end{pmatrix} \begin{pmatrix} Q\,(x_1) \\ Q\,(x_2) \\ \cdots \\ \cdots \\ \cdots \\ Q\,(x_n) \end{pmatrix} \tag{6.44}$$

where $z\,(x_i) = m\,(x_i)\,y\,(x_i)\,/\Delta x$.
Because m(x) is an exponent function of $P(x)$, $m(x) > 0$ is always right. operator Ghas a unique inverse G^{-1}. We can solve it by find G^{-1}.

6.2 Given Q(x), Find P(x),and without Input Errors

Similarly, we suppose $e^{\int P(x)\,\mathrm{dx}} = m(x)$,and also get

$$m(x)\,y(x) = \int_{x_0}^{x} m(x)\,Q(x)\,\mathrm{dx} \tag{6.45}$$

However,This is no longer **Volterra Integral Equations of the First Kind**,and it's the second kind of VIEs. Let

$$\hat{Q}(x, x_i) \quad = \quad \begin{cases} Q(x) & x \leqslant x_i \\ 0 & x > x_i \end{cases} \tag{6.46}$$

Hence,we can get a well-posed second kind Fredholm Integral Equations.

$$m(x_i)\,y(x_i) = \int_{x_0}^{x_n} m(x)\,\hat{Q}(x_j, x_i)\,dx \tag{6.47}$$

we use quadrature methods to discretize above system,

$$m(x_i)\,y(x_i) = \sum_{j=1}^{n} \hat{Q}(x_j, x_i)\,m(x_j)\,\Delta x, i = 1, 2, \ldots, n \tag{6.48}$$

i.e.

$$\frac{1}{\Delta x} m(x_i) = \sum_{j=1}^{n} \frac{\hat{Q}(x_j, x_i)}{y(x_i)} m(x_j), i = 1, 2, \ldots, n \tag{6.49}$$

Let $\frac{\hat{Q}(x_j, x_i)}{y(x_i)} = G_{i,j}$,then we have

$$Gm = \frac{1}{\Delta x} m \tag{6.50}$$

From above equation,we get m is an eigenvector with eigenvalue $\frac{1}{\Delta x}$ of G.
Sadly,it's easy to know that there n different eigenvalues of G,if $Q(x_i)$ are different.Then,we will get n solutions.It's seemed that the question is ill-posed as we have done above,but actualy,we only get one solution if we fix Δx.
However,Δx is certained,if we get (x_i, y_i). So we should discuss separately by the Value of Δx.

- if $\frac{1}{\Delta x}$ is one of the eigenvalues of G,then we can calculate m by it.
- if $\frac{1}{\Delta x}$ is not one of the eigenvalue of G,which is more common.First of all, we choose most closed one Q_i(In fact, we can find that every Q_i is one of the eigenvalues of G),that satisfying

$$\min_{i=\{1,2,\ldots,n\}} \left(\frac{1}{\Delta x} - Q_i \right) := \alpha \tag{6.51}$$

Now we solve equation

$$(G + \alpha I)\, m = \frac{1}{\Delta x} m \tag{6.52}$$

We can solve it ,and get an m_α which is most closed solution.

6.3 Given P(x), Find Q(x),and Have Input Errors

Similarly with none errors case,we will get

$$y^{\delta}(x_i) = \sum_{j=1}^{n} \frac{\hat{m}(x_j, x_i)\,\Delta x}{m(x_i)} Q(x_j) \tag{6.53}$$

thus we have $G_{i,j} = \frac{\hat{m}(x_j, x_i)\Delta x}{m(x_i)}$,and we have a new form

$$y^{\delta} = GQ \tag{6.54}$$

by Tikhonov Regularization,we have minimum norm solution

$$Q_\alpha^\delta = \left(G^*G + \alpha^2 I\right)^{-1} G^* y^\delta \tag{6.55}$$

by Arcangeli Criterien,we choose

$$\alpha = \frac{\left\|GQ_\alpha^\delta - y^\delta\right\|}{\delta} \tag{6.56}$$

$$= \frac{\left\|G\left(G^*G + \alpha^2 I\right)^{-1} G^* y^\delta - y^\delta\right\|}{\delta} \tag{6.57}$$

6.4 Given Q(x), Find P(x),and Have Input Errors

Similarly with none errors case,we will get

$$y^\delta(x_i)\, m^\delta(x_i) = \sum_{j=1}^{n} \hat{Q}(x_j, x_i)\, m^\delta(x_j)\, \Delta x, i = 1, 2, \ldots, n \tag{6.58}$$

$$y^\delta(x)\, m^\delta(x) = \begin{pmatrix} \sum_{j=1}^n \hat{Q}(x_j, x_1)\, m^\delta(x_j)\, \Delta x \\ \cdots \\ \cdots \\ \sum_{j=1}^n \hat{Q}(x_j, x_n)\, m^\delta(x_j)\, \Delta x \end{pmatrix} \tag{6.59}$$

Suppose there is no errors in data y,then $y^\delta(x) = y(x)$,by none error case,we have get a solution $m(x)$,which also satisfying

$$y(x)\, m(x) = \begin{pmatrix} \sum_{j=1}^n \hat{Q}(x_j, x_1)\, m(x_j)\, \Delta x \\ \cdots \\ \cdots \\ \sum_{j=1}^n \hat{Q}(x_j, x_n)\, m(x_j)\, \Delta x \end{pmatrix} \tag{6.60}$$

Make $\mathrm{substraction}$ between (59)&(58),then we get

$$y^\delta(x)\left(m(x) - m^\delta(x)\right) = \begin{pmatrix} \sum_{j=1}^n \hat{Q}(x_j, x_1)\left(m(x_j) - m^\delta(x_j)\right)\Delta x \\ \cdots \\ \cdots \\ \sum_{j=1}^n \hat{Q}(x_j, x_n)\left(m(x_j) - m^\delta(x_j)\right)\Delta x \end{pmatrix} \tag{6.61}$$

Let $k(x) = m(x) - m^\delta(x)$,then $k(x)$ also fixes equation (59) (changes m into k).We have claims that if Δx is stable,then we can only find one solution. Thus we cannot find another $m(x)$ if we suppose $y^\delta(x) = y(x)$.i.e $m(x) = m^\delta(x)$,or $m(x) = \mathbf{0}$.
From above discussion,we know that if we do not make a regularization of this second kind of Fredholm equation, we won't clean away or even decrease the data errors.

7 MATLAB interpolation approach to specific parameter estimation problems

7.1 Constant Parameter Scenario: Multiple Shooting method

We can see that it is not easy to estimate our parameter $p(x)$ and $q(x)$ in our first order linear equation. However, if the form of $p(x)$ and $q(x)$ are given, like $\sin \alpha x$, in which α is to be determined, then we can use MATLAB to accomplish this mission. In addition, if we only need to consider one parameter, say α in $\sin \alpha x$, then we even can endure some error since we will use least square method. In MATLAB, we will use function `fminsearch` to determine our goal function SSE (Sum-Square-of-errors) To visualize our task, we can now assume that our differential equation is:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \sin \alpha t + \cos \beta y \tag{7.62}$$

First, we should produce a series of accurate data, and plus our noise in it for the second case. We assume that our noise, which is a Random Variable, satisfy $noise \sim N(0, 0.01^2)$ (Gaussian White Noise) .In MATLAB it can be expressed as :

```
noisenormrnd(MU,SIGMA,m,n)
```

where m,n is the size of matrix. We first solve our problem assuming that our observation data is with no noise.

7.1.1 No Noise applied

Now we produce our accurate data, which will be used as our observed data.
1.determine our dy fuction (destination.m)

```
function dy = destination(t,y,flag,alpha,beta)
dy=sin(alpha*t)+cos(beta*y);
```

2.use ODE45 to solve the equation and plot the accurate data(command line)

```
y0=3;alpha=4;beta=5;N=1000;
tspan=linspace(0,10,N);
[t ymodelt]=ode45('destination',tspan,y0,[],alpha,beta);
[t ymodel]=ode45('destination',tspan,y0,[],alpha,beta);
plot(t,ymodel);
```

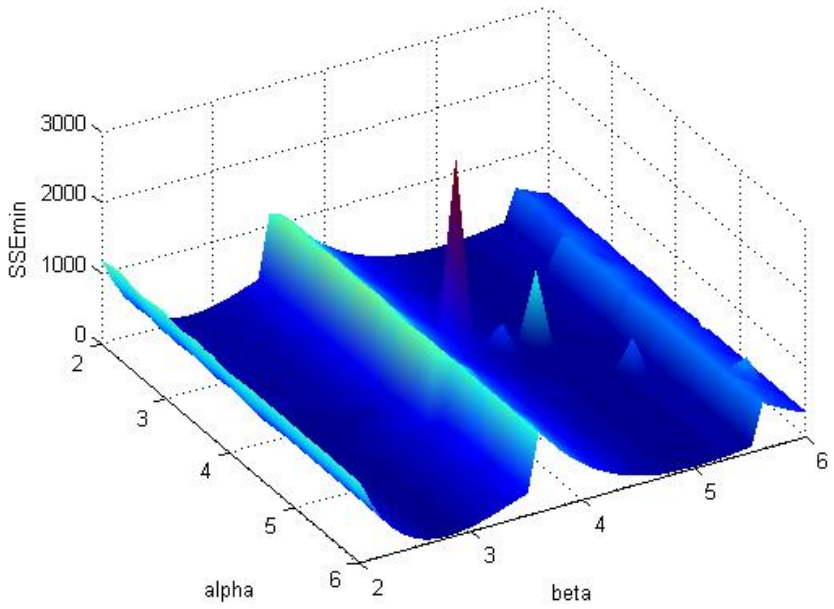
Fig 4.1
3.In order to solve for alpha, beta using `fminsearch`, we need to write a function which can calculate the error of our estimation and the data given above. Lets call such function SSE_1.m(Sum-Square-of-errors)

```
function val=SSE_1(k,tspan,yobs)
alpha=k(1);
beta=k(2);
tspan=tspan';
y0=3;
[t ymodel]=ode45('destination',tspan,y0,[],alpha,beta);
resid=ymodel-yobs;
val=resid'*resid;
```

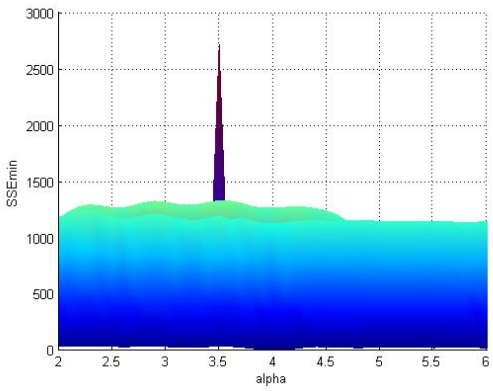
4.We can now plot alpha-beta-SSE plot simply using SSE_1:

```
x=2:0.1:6;
y=2:0.1:6;
[X Y]=meshgrid(x,y);
for m=1:41
    for n=1:41
        Z(m,n)= SSE_1([X(m,n) Y(m,n)],tspan, ymodelt);
    end
end
```

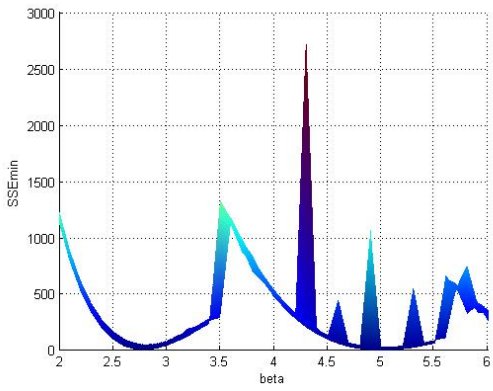
```
grid on;
surf(X,Y,Z);
shading interp;
xlabel('alpha');ylabel('beta'); zlabel('SSE');
view([60 50]);
```



(a) SSE plot



(b) SSE plot (α)



(c) SSE plot (β)

Figure 7.1: SSE plot respect to α and β

We can clearly see that when $\alpha \approx 4$ and $\beta \approx 2.8$ and 4.9 the SSE gets its minimum. For more precise calculation, we should use `fminsearch` to get α and β. Recall that we now have `tspan`, `ymodelt` and we should give MATLAB initial estimation of alpha and beta. Then we can use `fminsearch` to directly solve the problem. We shall now guess that $\alpha = 4$ and $\beta = 2.8$:

```
>> k=[4 2.8];
>> [k SSEmin]=fminsearch('SSE_1',k,[],tspan,ymodelt)
```

The result is:

```
k =
    4.0426    2.7839
SSEmin =
```

```
1.5513
>> k=[4.0426 2.7839];
>> [k SSEmin]=fminsearch('SSE_1',k,[],tspan,ymodelt)
k =
    4.0426    2.7838
SSEmin =
    1.5513
```

The process of iteration indicate that point `[4.0426 2.7839]` is really the local minimum. Dont forget that we have another approximation $\alpha = 4$ and $\beta = 4.9$:

```
>> k=[4 4.9]
>> for m=1:6
[k SSEmin]=fminsearch('SSE_1',k,[],tspan,ymodelt);
k
SSEmin
end

k =
    4.0001    4.9988
SSEmin =
   5.8807e-04
k =
    4.0004    5.0002
SSEmin =
   1.6218e-04
k =
    4.0004    5.0002
SSEmin =
   1.5774e-04
```

Such iteration indicates that `k=[4.0004 5.0002]` is the final answer given by MATLAB. We now compare the SSE of two local minimum: `1.5774e-04` when `k=[4.0004 5.0002]` and 1.5513 when `k=[4.0426 2.7839]`. It is obvious that we should choose `k=[4.0004 5.0002]`, which means that the approximation for α and β using numerical method is given by $\alpha = 4.0004$ and $\beta = 5.0002$, which indeed close to the exact value $\alpha = 4.0000$ and $\beta = 5.0000$.

7.1.2 Noise applied

We shall now consider the case that the observed data is with noise. To see whether noise is actually distributed as normal distribution, we call draw a graph:

```
noise=normrnd(0,0.01,1000,1);
[a,b]=hist(noise);
bar(b,a/sum(a))
```

1.add our noise and plot the data again in Figure **??** and Figure **??**

```
for i=1:1000
    ymodelt(i)=ymodelt(i)+noise(i);
end
plot(t,ymodelt);
hold on;
plot(t,ymodel,'r');
xlabel('t');ylabel('y');
hold off;
```

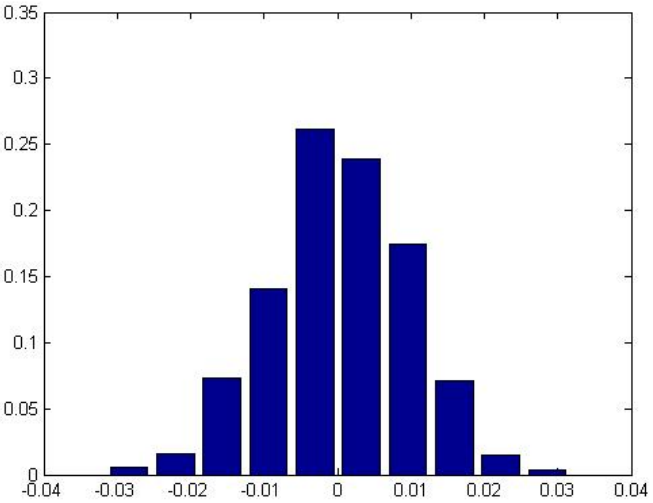
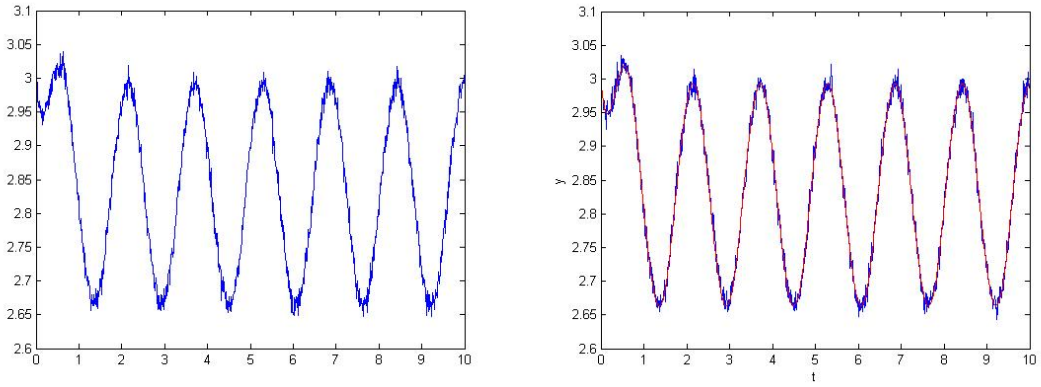


Figure 7.2: Distribution of noise



(a) Observed Data

(b) Comparison of data with and without noise

Figure 7.3: Data with noise

2.We shall first eliminate our noise in order to process our data using FFT (Fast Fourier Transformation)(Figure **??**) and IFFT (Inverse Fast Fourier Transform). After FFT and IFFT, the magnitude of our data will shrink, so we should amplify our data.

```
z= fft(ymodelt);%FFT
po=1:1:1000;
plot(po,z);
xlabel('relative frequency');ylabel('amplitute');
```

3.We can see that we should eliminate the high frequency part and do the IFFT. Note that in the Figure below, the red plot is our data after IFFT and the green plot is our origin data.

```
filter(1:50)= 1;%filter magnitude
filter(51:N)=0.00;%
for i=1:N
    thispy(i)= filter(i)*z(i);%filter
end
Z=ifft(thispy,N);%IFFT
plot(t,Z, 'r-');
hold on;
plot(t,ymodelt, 'g-');
```

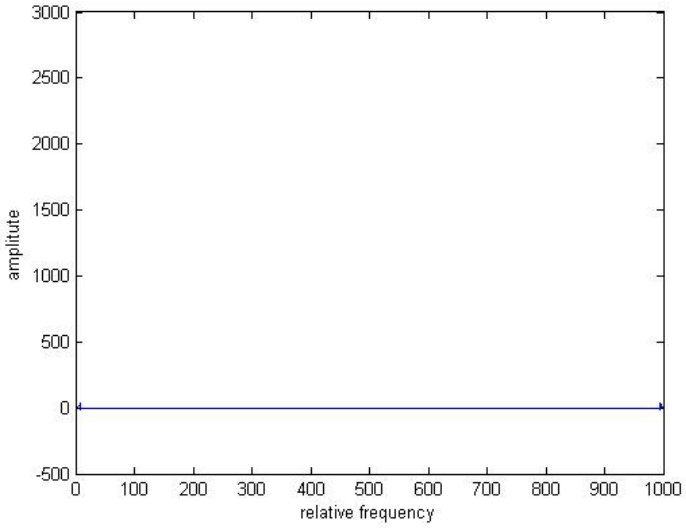


Figure 7.4: FFT of Observed Data

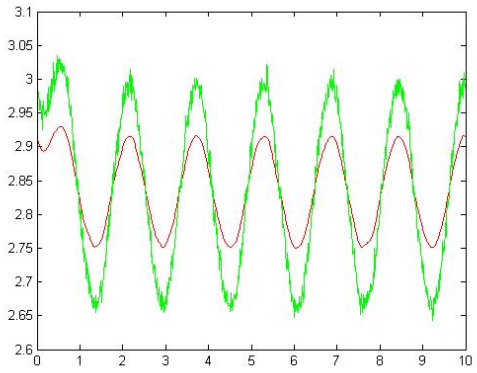


Figure 7.5: Data After FFT and IFFT

4.Amplification of our data:

```
Z=abs(Z);%eliminate imaginary number
ymodel_average = sum(ymodel)/length(ymodel);
Z_average = sum(Z)/length(Z);
averageDelta = ymodel_average - Z_average;
ymodel_max=max(ymodel);
ymodel_min=min(ymodel);
for i=1:N
    Z(i)=Z(i)+ averageDelta;%move Z in order to meet the average of original data
end
Z_max=max(Z);
Z_min=min(Z);
Amplify=min((ymodel_max- ymodel_average)/( Z_max -  ymodel_average), (ymodel_min-
ymodel_average)/( Z_min -  ymodel_average));
for i = 1 : N
    Z(i)= ymodel_average+(Z(i)- ymodel_average)* Amplify;
end
plot(t, Z);
hold off;
```

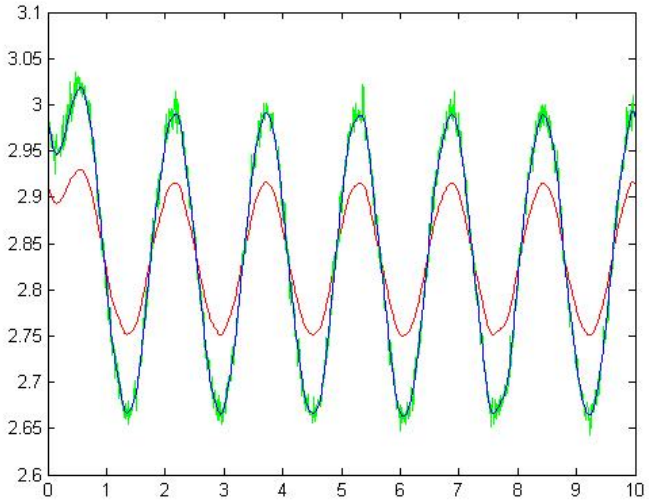


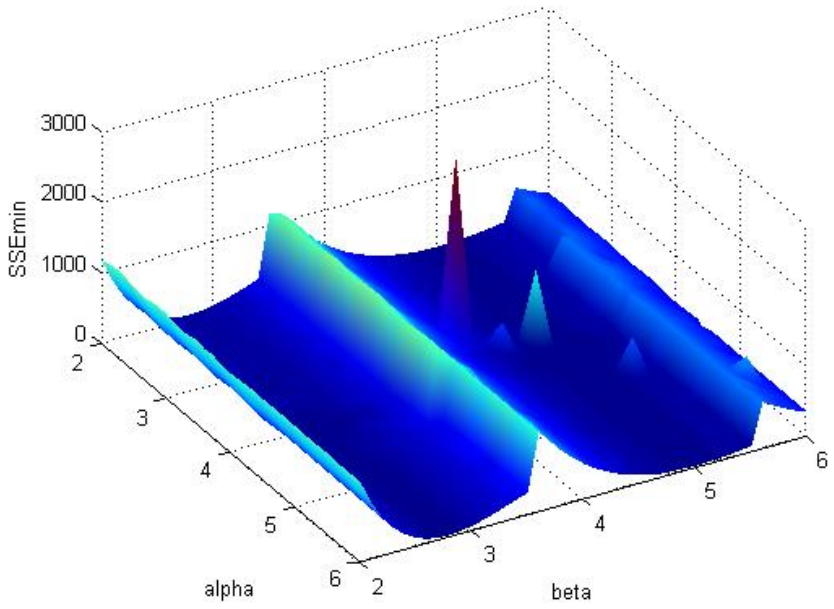
Figure 7.6: Data after amplification (Blue)

5.We can see that our datas noise is virtually eliminated. After this procedure, the process of getting SSE is the same.

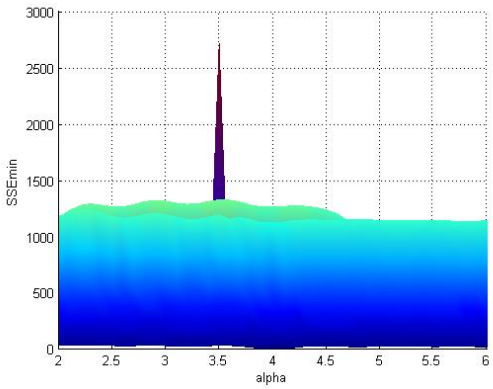
```
ymodelt=Z';
clear Z;
x=2:0.1:6;
y=2:0.1:6;
[X Y]=meshgrid(x,y);
for m=1:41
    for n=1:41
        Z(m,n)= SSE_1([X(m,n) Y(m,n)],tspan, ymodelt);
    end
end
grid on;
surf(X,Y,Z);
shading interp;
xlabel('alpha');ylabel('beta'); zlabel('SSE');
view([60 50]);
```

6.It is not astonishing that the SSE figures (Figure **??**) before and after the noise is added are nearly the same. Such result indicates that FFT and IFFT really work in this scenario. And what we should do now is merely repeat fminsearch to get the best approximation of α and β

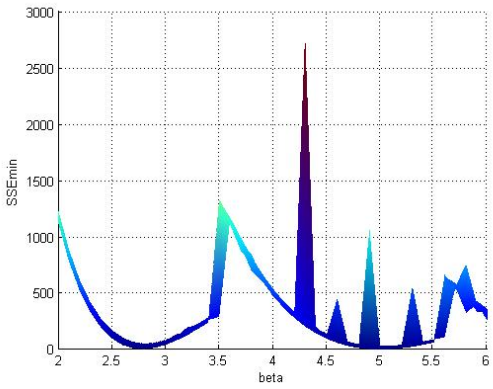
```
>> k=[4 2.8];
>> for m=1:2
    [k SSEmin]=fminsearch('SSE_1',k,[],tspan,ymodelt);
    k
    SSEmin
end
k =
    4.0450    2.7829
SSEmin =
    1.7462
k =
    4.0450    2.7829
SSEmin =
    1.7462
```



(a) SSE plot



(b) SSE plot (α)



(c) SSE plot (β)

Figure 7.7: SSE plot respect to α and β

```
>> k=[4 4.9];
>> for m=1:6
    [k SSEmin]=fminsearch('SSE_1',k,[],tspan,ymodelt);
    k
    SSEmin
end
k =
    4.0001    4.9988
SSEmin =
    0.0181
k =
    4.0001    4.9988
SSEmin =
    0.0181
k =
    4.0001    4.9987
SSEmin =
```

```
    0.0178
k =
    4.0001    4.9987
SSEmin =
    0.0178
```

Under the initial condition $\alpha = 4, \beta = 2.8$ and $\alpha = 4, \beta = 4.9$, the result of iteration using `fminsearch` indicates that the best approximation of α and β is $\alpha = 4.0001, \beta = 4.9987$

7.1.3 Conclusion

We now conclude our answer:

Table 7.1: Conclusion of α and β

	α	β
exact value	4.0000	5.0000
without noise	4.0004	5.0002
with noise	4.0001	4.9987

Such approximation can meet our precision.

8 Interpolation Methods

suppose we have several points given by the coordinate: $(x_1, y_1); (x_2, y_2); (x_3, y_3).....(x_n, y_n)$. given $x \in [x_{j-1}, x_j]$,how to get the y value corresponding to x ?we can use some functions called interpolate to estimate the y value, and this method is called interpolation.

8.1 Linear Interpolation

this method creates linear function connecting two points $(x_{j-1}, y_{j-1}); (x_j, y_j)$

$$y = y_{j-1} + \frac{(x - x_{j-1})(y_j - y_{j-1})}{(x_j - x_{j-1})}$$

for any x in between, we can get corresponding y through this linear function.

8.2 Polynomial Interpolation

assume we have n points, there exists unique nth order polynomial that can connect these n points, this nth order polynomial is called Polynomial interpolate .
Lagrange polynomial (one method to get this n th order polynomial):

$$P(x) = \sum_{j=1}^{n} \prod_{i=1, i \neq j}^{n} \frac{x - x_j}{x_i - x_j} y_i$$

8.3 Cubic Spline Interpolation

Basically, for each two successive points, spline is a local 3rd order polynomial crossing these points. But we have an additional requirement, that is the 1st and 2nd derivative must be continuous especially at every boundary point.
Assume we have n+1 points,$(x_0, y_0); (x_1, y_1); (x_2, y_2); (x_3, y_3), ..., (x_n, y_n)$. Let $S_i(x)$ be the cubic spline function between $[x_i, x_{i+1})(i = 0, 1, 2, 3....., n-1)$. let

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

where a,b,c,d are unknown parameters. in summery we have 4n unknown parameters , by common sense we need 4n linear equations.

Since for $\forall S_i(x), S(x_i) = y_i, S(x_{i+1}) = y_{i+1}$ we can get 2n linear equations,
$\forall x_i S_{i-1}^{'}(x_i) = S_i^{'}(x_i) i = 1, 2, 3, ..., n-1$ we get another n-1 linear equations.
$\forall x_i S_{i-1}^{''}(x_i) = S_i^{''}(x_i), i = 1, 2, 3, ..., n-1$ we get another n-1 linear equations.
In summary we have 4n-2 equations, so we can have two dimension of freedom. the normal process is setting $S_0^{''}(x_0) = 0, S_{n-1}^{''}(x_{n-1}) = 0$,such that we eliminate the rest dimension of freedom.
finally, in order to get these 4n unknown parameters. our task become to solve n-1 linear equations, that is to solve the reverse of$(n-1) \times (n-1)$matrix.

8.4 Other Interpolation Methods

8.4.1 Rational Functions

this method use the rational function to estimate values.

8.4.2 Trigonometric Interpolation

this method use trigonometric-sum to approximate discrete data. explicitly, the function should be

$$f(x) = a_0 + \sum_{n=1}^{N} a_n cos(nx) + \sum_{n=1}^{N} b_n sin(nx)$$

9 $P(x)$ Estimation by Using MATLAB Interpolation Method

Interpolation is a method of constructing new data points within the range of a discrete set of known data points. In engineering and science, one often has a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable. It is often required to interpolate (i.e. estimate) the value of that function for an intermediate value of the independent variable. This may be achieved by curve fitting or regression analysis.
In our case, we using interpolation to accomplish the parameter estimation problem of differential equation which has the form:

$$\frac{dy}{dx} = P(x)y + Q(x) \tag{9.63}$$

and during the process, a comparison between different MATLAB interpolation methods is also shown in the paper.

9.1 Continue $P(x)$ fitting

9.1.1 A special case study: $Q(x) = 0$

Let's begin with P(x) which is continuous, and we shall first discuss a special case that $Q(x) = 0$.
Now we produce our accurate data, which will be used as our observed data.
1.determine our P(x) function (expinter.m)

```
x=0.1:0.05:7;
y=exp(-cos(x));
```

2.use spline function to interpolate the original data into a more accurate one

```
xi=0.1:0.01:7;                          %
yi=interp1(x,y,xi,'spline');                    %splineyi
```

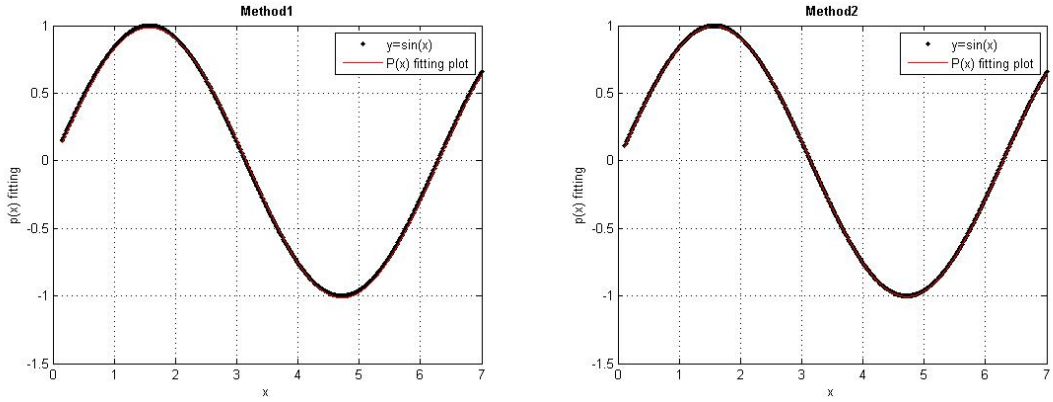
3.In order to solve for $P(x)$, the value of $\frac{dy}{dx}$ is obviously necessary. Here we have two ways to go, one is to get the value of $\frac{dy}{dx}$ every five points (expinter.m), and we need to do interpolation again to get a more accurate data of P(x), while the other one only needs to have one interpolation because it get value of $\frac{dy}{dx}$ every point (expintercubic.m). It is easy for us to give an assumption that the second method is better than the first one, and we shall use $P(x) = \cos x$ as an example to verify this.
The first method:

```
for k=1:(length(y)-1)
    dyi(k)=(yi(5*k)-yi(5*k-1))/(xi(5*k)-xi(5*k-1));
    p(k)=dyi(k)/y(k+1);
end
    x2=0.15:0.05:7;
    x3=0.15:0.01:7;
    pnew=interp1(x2,p,x3,'spline');
    plot(x3,sin(x3),'k.',x3,pnew,'r')
    title('Method1','fontweight','bold')
    xlabel('x','fontsize',10)
    ylabel('p(x) fitting','fontsize',10)
    legend('y=sin(x)','P(x) fitting plot')
    grid on;
    sigma=0;
for t=1:length(x3)
        sigma=sigma+(pnew(t)-sin(x3(t)))^2;
end
    sigma
```

The second method:

```
for k=1:(length(yi)-1)
    dyi(k)=(yi(k+1)-yi(k))/(xi(k+1)-xi(k));
    p(k)=dyi(k)/yi(k+1);
end
    x3=0.11:0.01:7;
    plot(x3,sin(x3),'k.',x3,p,'r')
    title('Method2','fontweight','bold')
    xlabel('x','fontsize',10)
    ylabel('p(x) fitting','fontsize',10)
    legend('y=sin(x)','P(x) fitting plot')
    grid on;
    sigma=0;
for t=1:length(x3)
        sigma=sigma+(p(t)-sin(x3(t)))^2;
end
    sigma
```

Notice that all of the interpolation methods we used in these two methods are `spline`, which ensures the reliability of this comparison.



(a) Method 1 (b) Method 2

Figure 9.8: The fitting plot of Method1 and Method2

And the result of sigma(SSE: sum square of the error) is:

```
>> expinter
sigma =
    0.1390
>> expintercubic
sigma =
    0.0156
```

Obviously, the SSE of the second method is much smaller than the first one, and this is easy to understand because every time you do an interpolation, new error would be made and at last they would composite and become an even larger error.

9.1.2 MATLAB interpolation methods' study when $Q(x) = 0$

So based on the second method, we shall do a comparison between different MATLAB interpolation method:
1.`linear` interpolation
Linear interpolation is a method of curve fitting using linear polynomials. It is quick and easy, but it is not very precise. Another disadvantage is that the interpolant is not differentiable at the point x_k. As we shall see, its SSE is much bigger than the other methods'.

```
yi=interp1(x,y,xi,'linear');                    %linearyi
```

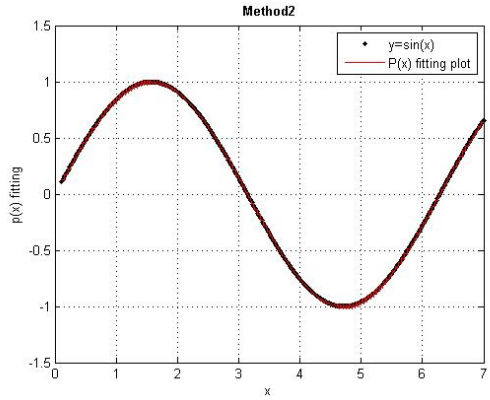
2.`spline` interpolation
Spline interpolation uses low-degree polynomials in each of the intervals, and chooses the polynomial pieces such that they fit smoothly together. The resulting function is called a spline. And compare with linear interpolation, spline interpolation has a smaller error and the interpolant is smoother.

```
yi=interp1(x,y,xi,'spline');                    %splineyi
```

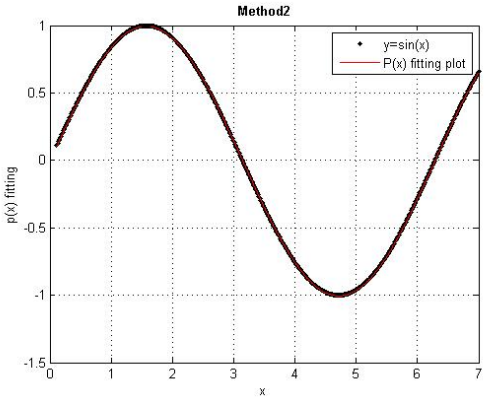
3.`cubic` interpolation
In the mathematical subfield of numerical analysis a cubic Hermite spline (also called cspline), named after Charles Hermite, is a third-degree spline with each polynomial of the spline in Hermite form. The Hermite form consists of two control points and two control tangents for each polynomial. We shall see in the later result, because the Hermite polynomials it use is third-degree, of which the degree is same as `spline` function, the SSE of these two methods have little difference.

```
yi=interp1(x,y,xi,'cubic');                    %cubicyi
```

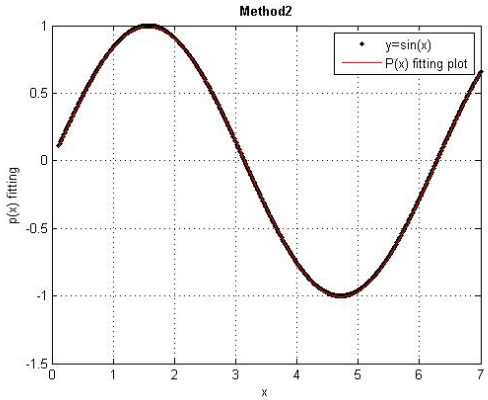
The MATLAB plots here:



(a) Linear interpolation



(b) Spline interpolation



(c) Cubic interpolation

Figure 9.9: Fitting plot of the three methods

And the SSE of these methods are:
1.`linear` interpolation:

```
sigma =
    0.1399
```

2.`spline` interpolation:

```
sigma =
    0.0156
```

3.`cubic` interpolation:

```
sigma =
    0.0173
```

As we see, just like what we assume before, `linear` interpolation has the biggest SSE while `spline` interpolation and `cubic` interpolation have smaller ones and the difference of them is little.(The more detailed analysis of their accuracy level will be shown in the theoretical analysis part of these interpolation method.)

9.1.3 General case: when $Q(x) \neq 0$

However, here we still need to have the condition that $Q(x)$ is known. For such case it is easy to understand that the question is still easy, because we just need to minus the value of $Q(x)$ from $\frac{dy}{dx}$, divide the value of y, and then we would get the $P(x)$ at that point. This is almost the same as the case when $Q(x) = 0$, so we would not give an example.

9.2 Discontinue $P(x)$ fitting

For convenience, we assume $Q(x) = 0$ Now we produce our accurate data, which will be used as our observed data.
1.determine our P(x) function (uncontinue.m)

```
x=-1:0.05:3;
x01=-1:0.05:1;
y01=exp(0.5*x01.^2);
x02=1.05:0.05:3;
y02=exp(-0.5*x02.^2+1);
y=[y01,y02];
```

Which means that $P(x) = x$ on $[-1, 1]$ and $P(x) = -x$ on $(1, 3]$.
2.use spline function to interpolate the original data into a more accurate one

```
xi=-1:0.01:3;                          %
yi=interp1(x,y,xi,'spline');                  %splineyi
```

3.Like before, we need to get the value of $\frac{dy}{dx}$ to solve for $P(x)$, and then plot it out and calculate its SSE. Here the interpolation method we use is `spline` function.
Solve for $P(x)$:

```
for k=1:(length(yi)-1)
    dyi(k)=(yi(k+1)-yi(k))/(xi(k+1)-xi(k));
    p(k)=dyi(k)/yi(k+1);
end
```

Plot it out:

```
    x3=-0.99:0.01:3;
    x4=-0.99:0.01:1;
    x5=1:0.01:3;
    subplot(1,2,1),plot(x3,p,'r.')
    title('Spline fitting of p(x)','fontweight','bold')
    xlabel('x','fontsize',10)
    ylabel('p(x) fitting','fontsize',10);
    subplot(1,2,2),plot(x4,x4,'k',x5,-x5,'k')
    title('Real p(x)','fontweight','bold')
    xlabel('x','fontsize',10)
    ylabel('p(x)','fontsize',10);
```

Solve for SSE:

```
    sigma=0;
for t=1:(length(x4)-1)
    sigma=sigma+(p(t)-x4(t))^2;
end
for t=1:length(x5)
    sigma=sigma+(p(t+length(x4)-1)+x5(t))^2;
end
    sigma
```

And the fitting plot is: And the result of SSE is:

```
>> uncontinue
sigma =
    2.9607
```

From the picture, one can easily figure out the discontinuity of $P(x)$, and can easily told a small interval that the discontinue point in. And then we can divide the discussion of $P(x)$ into two parts, one is from the lower bound of the domain of definition to the discontinuous point, another is from the point to the upper bound, and that come backs to the case that $P(x)$ is continue. Also, it is easy to observe that the SSE here is a little bit large, however if we remove the 10 points around 1 when we are calculating the SSE, we shall see that it would become a very small value.

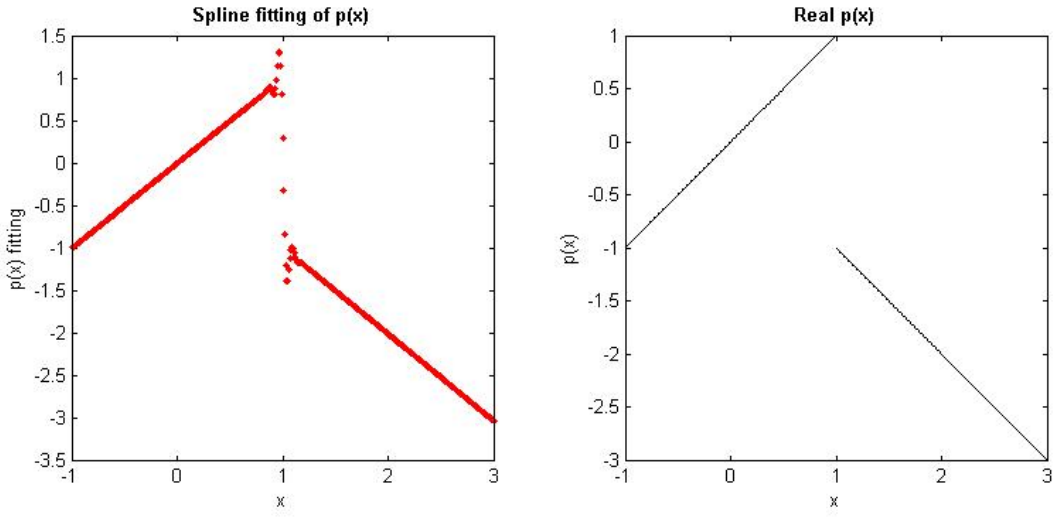


Figure 9.10: Fitting P(x) and the original P(x)

```
for t=1:(length(x4)-5)
    sigma=sigma+(p(t)-x4(t))^2;
end
for t=5:length(x5)
    sigma=sigma+(p(t+length(x4)-1)+x5(t))^2;
end
    sigma
```

And the SSE become:

```
sigma =
    0.4658
```

which is a quite small one. This shows that the fitting result is quite good at the two sides of the function.
So from all of the discussion, we need to conclude what we get:
1.How to use the MATLAB interpolation method solving the problem of parameter estimation.(continue and discontinue)
2.Among all the interpolation method, `linear` function has the lowest accuracy level while `spline` function and `cubic` function are better.

10 Conclusion and Distribution

At the beginning of our article, we posted four questions, and three of them can be answer by such demonstration with MATLAB. They are:
1. How can we obtain parameters from such model (approximation)
Using least square criterion, we calculate the Sum-Square-of-errors (SSE) to obtain our parameter
2. Is the model existing the only one model (uniqueness)?
The answer to such question cannot completely answered by the example accompanied with MATLAB. However, we can sure that there is situation where the parameter of our model is unique, it can be seen in the example Constant Parameter Scenario, where β can be either 2.78 or 5.00 (approximation).
3. Can the model endure little error of the output (stability)?
The answer is surely illustrated by example with MATLAB. We can see that FFT and IFFT is a very useful tool to erase the noise. We pick a noise which is distributed as normal distribution, which simulate the actual scenario of measurement.

Distribution:
W. Zhang: Discritization,SVD ,Methods of Regularization,Iterative method,Applications of Regularization
Q. Wu: Multiple Shooting Method with MATLAB,SVD, redaction

J. Wang: $P(x)$ Estimation with MATLAB (Cont. and DisCont. case)
T. Luo: Interpolation Methods

References

[1] G. Backus and F. Gilbert. Uniqueness in the inversion of inaccurate gross earth data. *Philosophical Transactions for the Royal Society of London. Series A, Mathematical and Physical Sciences*, :123–192, 1970.

[2] G.H. Golub and C. F. Van Loan. *Matrix Computation*. The Johns Hopkins University Press, 1996.

[3] Charles W. Groetsch. *Inverse Problems in the Mathematical Sciences*. Informatica International, Inc., 1993.

[4] And A. Neubauer H.W.Engl, M.Hanke. *Regularization on Inverse Problem*. Kluwer Academic Publishers, 1996.

[5] A. Kirsch. *An introduction to the mathematical theory of inverse problems*, volume 120. Springer, 2011.

[6] C. Lanczos. *Linear Differential Operators*. SIAM, 1996.

[7] Vogel C R. Non-convergence of the l-curve regularization parameter selection method. *Inverse Problems*, :535–547, 1996.

[8] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*. Society for Industrial Mathematics, 2005.

[9] Arsenin V Y Tikhonov A N. *Solutions of Ill-Posed Problems*. John Wiley and Sons, 1977.

[10] Groesch C W. *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*. Pitman Advanced Publishing Program, 1984.

[11] Groesch C W. *Inverse Problems in the Mathematical Sciences*. vieweg Inc.