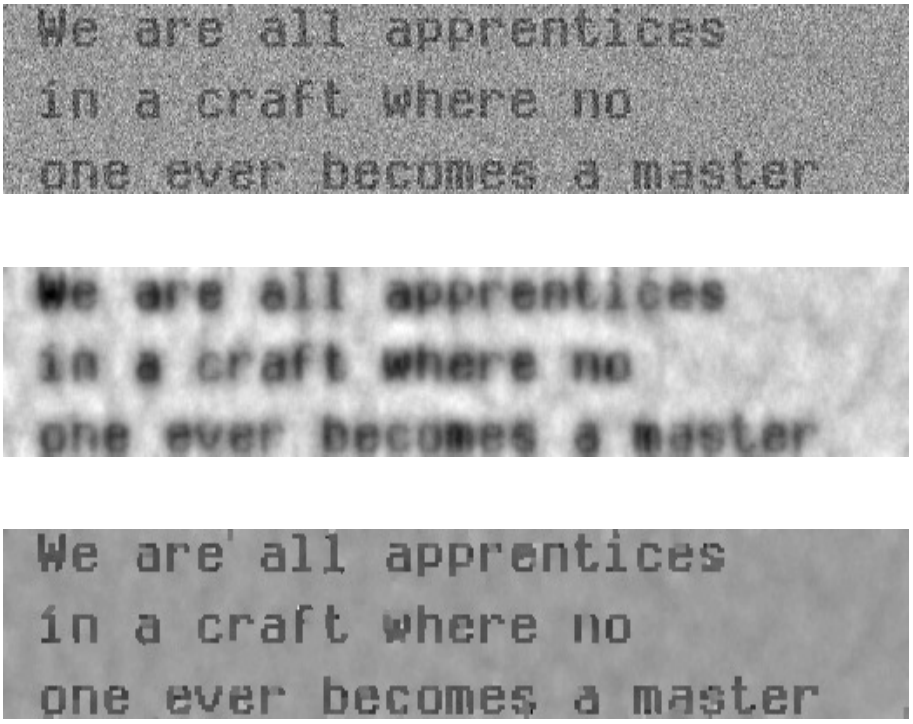
GEOPH 531

Inverse Problems

M D Sacchi



University of Alberta, Winter 2023

Contents

Chapter 1

Introduction

For the most part, the study of the earth's subsurface is *indirect*. In other words, surface measurements infer the subsurface's structures and properties. Geophysicists use basically different methods to obtain subsurface properties. The seismic method (exploration and global seismology) uses variations in the traveltime and amplitude of propagating waves to infer the elastic properties of the subsurface. Potential methods depend on the distortion of a scalar or vector potential field caused by some perturbation of a suitable physical property in the subsurface (Gravity and Magnetic methods). Then we have electrical and electromagnetic methods where currents induced in the earth (natural or human-made currents) are used to estimate the subsurface electrical conductivity.

Geophysical inverse methods aim to successfully translate measurable quantities acquired on the earth's surface into a *picture* of the subsurface geology. The latter depends on understanding the nature of the measurements, developing a mathematical theory to link properties to observations and finally, on the vital link between theoretical mathematical physics aspects of geophysics and geological reality. This course is mainly concerned with understanding inverse theory's physical and mathematical aspects and its application to applied and global geophysical problems. Practical applications of the material discussed in this course encompass content taught in the following classes: Geoph 326, Geop 325, Geoph 421, Geop 436, Geop 424 and Geop 438.

1.1 Forward and Inverse Problems

We first adopt an abstract formulation of the inverse problem. The formulation will become apparent when you develop numerical solutions to an inverse problem. We designate d the data that have been acquired and that provides indirect information about the subsurface. We will call the model one would have liked to observe m. We cannot measure properties everywhere inside the earth, and this is the main reason for m being unknown. We only have access to information about these properties in an indirect way. The latter can be represented by a simple mathematical expression

$$d = \mathcal{F}[m] \quad \textit{The Forward Problem}\,. \tag{1.1}$$

The symbol $\mathcal{F}$ represents the operator that maps the *model* that one would have liked to know into the *data* that one was able to measure. For instance, if m represents subsurface density and d measurements of the vertical component of the gravity field (after applying several corrections), then $\mathcal{F}$ represents Newton's gravitational law.

We summarize the main points of this discussion:

$\mathcal{F}$: Mathematical description of the physical process under study

m: Unknown distribution of physical properties

d: Data

It is essential to mention that one only observes data in some places. Data are measured at fixed positions in time and space. These are positions where one can deploy an instrument to acquire data. For instance, a gravimeter will obtain data every 25 metres, and seismometers will measure teleseismic waves produced by earthquakes at stations that are 100s of kilometres apart. Hence, we stress that observations are quantities taken at a finite number of points in time and/or space

$$d_j\,, j = 1, \ldots N \qquad \text{Finite number of observations}$$

where N indicates the number of observations. Equation 1.1 is now expressed as follows

$$d_j = \mathcal{F}_j[m]\,, j = 1 \ldots N \tag{1.2}$$

We can now discuss one interesting aspect of the inverse theory. We have a finite number of numbers on the left-hand side of equation 1.2. On the right-hand side of 1.2, we have a property expressed as a function m. For instance, if the property is density in the earth's interior $m = \rho(x, y, z)$. The argument x, y, z stresses that density exists everywhere in the earth's interior. Equation 1.2 shows an important degree of *indeterminacy*. Given N observations, we would like to estimate the function m (a function is represented by an ∞ number of points). Clearly, there is probably not enough information to recover m from d unless something is said about the m[1]. In addition, noise (or errors) will add extra complexity to our problem

$$d_j^{obs} = \mathcal{F}_j[m] + e_j\,, j = 1 \ldots N\,. \tag{1.3}$$

The finite set of observations can be written in vector form[2]

[1] We could assume, in some case, that m exhibits some regularity. For instance, m is a smooth function.

[2] Bold lower and upper case fonts are used to indicate vectors and matrices, respectively. For instance, $\mathbf{a}$ is a vector, and $\mathbf{A}$ is a matrix. When I write on the whiteboard, I will use $\vec{a}$ and $\underset{\sim}{A}$.

$$\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix} . \tag{1.4}$$

The problem can now be written as follows

$$\mathbf{d} = \mathbf{F}[m] + \mathbf{e} . \tag{1.5}$$

where $\mathbf{d}$ is the $N \times 1$ vector of observations. Similarly, $\mathbf{F}$ represents the operator that maps the model (a function) m into $\mathbf{d}$ and $\mathbf{e}$ is the $N \times 1$ vector representing observational errors.

Let us consider equation 1.5 and the following

1. Given the data $\mathbf{d}$
2. Given the physical equations describing problem $\mathbf{F}$
3. Details about the experiment
4. "Some knowledge" about the noise term $\mathbf{e}$. For instance, its strength and/or distribution.

What is the distribution of physical properties m ?. In other words, is it possible to construct an expression that estimates m from $\mathbf{d}$. Let's assume that an *inversion formula* exists, then one can write

$$\hat{m} = \mathbf{F}^{-1}\mathbf{d} . \tag{1.6}$$

Consider for a moment that $\mathbf{F}^{-1}$ is the inverse operator you could derive from $\mathbf{F}$. One could ask several interesting questions about equation . For instance,

1. Is there a solution $\hat{m}$? (*existence*)
2. Is the solution $\hat{m}$ unique ? (*uniqueness*)
3. Is the solution stable ? (*stability* [3])

We will return to these points when we discuss regularization methods for inverse problems.

[3] Imagine for a second that you have obtained a solution $\hat{m} = \mathbf{F}^{-1}\mathbf{d}$. Then, perturb the data $\mathbf{d} \rightarrow \mathbf{d} + \delta\mathbf{d}$ to obtain $m + \delta m = \mathbf{F}^{-1}[\mathbf{d} + \delta\mathbf{d}]$. The solution is unstable if, for a small data perturbation $\delta\mathbf{d}$, the model perturbation δm is large.

1.1.1 Linear and non-linear inverse problems

The operator $\mathcal{F}$ can be linear or non-linear. If $\mathcal{F}$ is linear, the following is true. Given m_1 and m_2 such that

$$d_1 = \mathcal{F}[m_1]$$

and

$$d_2 = \mathcal{F}[m_2]$$

the problem is linear if

$$\mathcal{F}[\alpha m_1 + \beta m_2] = \alpha \mathcal{F}[m_1] + \beta \mathcal{F}[m_2]$$

for arbitrary constants α and β. Otherwise, the problem is non-linear. We will first concentrate on linear inverse problems. The last part of the course will discuss non-linear inverse problems.

1.2 Examples of linear inverse problems

1.2.1 Fredholm integral equation of the first kind (FIE)

Many forward problems can be written down via an FIE of the following form

$$d(u) = \int_a^b K(u,x)\, m(x)\, dx\,. \tag{1.7}$$

Seismic convolution and deconvolution

Equation 1.7 represents the classical convolution problem. Consider convolution in the time domain with the following variables.

$d(u) \rightarrow s(t)$ The seismogram or seismic trace

$K(u,x) \rightarrow w(t-\tau)$ The seismic source wavelet

$m(x) \rightarrow r(\tau)$ The reflectivity in terms of two-way travel-time τ

The seismic trace can be expressed via the following convolution integral

$$s(t) = \int_a^b w(t-\tau)\, r(\tau)\, d\tau\,. \tag{1.8}$$

Expression 1.8 represents convolution[4]. The inverse problem entails retrieving $r(\tau)$ from $d(t)$. The inverse problem for this particular example is called *deconvolution* (Ulrych and Sacchi, 2006). Notice that I have made a critical assumption. I have assumed that the seismic source wavelet w is known. This is often a problem because the seismic wavelet is unknown or partially known. When both reflectivity and wavelet are unknown, we have a complicated problem called *Blind Deconvolution* (Kazemi and Sacchi, 2014).

Astronomical deblurring

Spatial convolution arises in image processing in areas such as astronomy (Bertero et al., 2009). Telescopes are not perfect devices that often introduce blurring or distortion to astronomers trying to capture images. The astronomical data $d(x, y)$ (a photographic or digital image of the sky) can be written downs via a 2D convolution integral

$$d(x, y) = \int \int b(x - u, y - v) i(u, v)\, du\, dv \tag{1.9}$$

where $b(x, y)$ represents the instrument's blurring kernel or impulse response (optical telescope or radio telescope). The astronomical image that one would have liked to measure is represented by $i(x, y)$. If the desired image were an impulsive image of the form $i(x, y) = \delta(x, y) = \delta(x)\delta(y)$, then $d(x, y) = b(x, y)$. The latter explains the name *impulse response* for $b(x, y)$.

Vertical Seismic Profile

Consider a Vertical Seismic Profile (Lizarralde and Swift, 1999). In this case, the traveltime is given by

$$t(z) = \int_0^z s(u) du\,. \tag{1.10}$$

The unknown $u(z) = \frac{1}{v(z)}$ is the reciprocal of the media velocity $v(z)$, and $t(z)$ is the traveltime for a wave that travels from the surface of the earth (vertical propagation) to a point z in the interior of the earth. It is easy to show that equation 1.10 can be written as an FIE

$$t(z) = \int_0^\infty s(u) H(z - u) du \tag{1.11}$$

where $H(u)$ is the Heaviside function.

[4]Often written in short form as follows $s(t) = w(t) * r(t)$

Gravity prospection

The vertical component of the reduced gravity anomaly is proportional to subsurface density perturbations $\Delta\rho(x, y, z)$ (Li and Oldenburg, 1998)

$$\Delta g_z(x_o, y_o, z_o) = \gamma \int_V \Delta\rho(x, y, z) \frac{(z - z_i)}{[(x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2]^{3/2}} \, dx \, dy \, dz \tag{1.12}$$

where (x_o, y_o, z_o) is the point where the anomaly has been measured and γ is Newton's gravitational constant.

Road map for this course

How do we construct a solution for an inverse problem? Is the solution unique? Can we characterize its degree of non-uniqueness? All these questions will depend not only on the characteristics of the operator $\mathbf{F}$ but also on the type of data one is considering. For instance, we could have the following scenarios:

A Infinite amount of accurate data

B Finite number of accurate data

C Finite number of inaccurate (noisy) data (This is the real-life scenario!)

I will leave A for theorists[5] and focus on B and C. Part 2 of this course will deal with issues that fall under B. We will introduce the concept of *regularization* of inverse problems and discuss methods to obtain stable solutions to linear inverse problems. In Part 3, we will introduce noise to the data and investigate regularization methods for inaccurate data.

1.3 Brief digression

Consider the FIE

$$d(u) = \int_X K(u, x) \, m(x) \, dx \, . \tag{1.13}$$

We can call u the output variable and x the integration variable. Equation 1.13 resembles a matrix-vector multiplication. Imagine for a second the multiplication of a vector $\mathbf{m}$ with a matrix $\mathbf{A}$. Consider $\mathbf{m}$ is an $M \times 1$ vector and $\mathbf{A}$ an $N \times M$ matrix. Hence,

$$\mathbf{d} = \mathbf{A}\mathbf{m} \tag{1.14}$$

[5] I am not aware of any geophysical problem where there is an infinite amount of data

is a vector of length $N \times 1$. We can rewrite expression 1.14 via indicial notation as follows

$$d_i = \sum_{j=1}^{M} A_{i,j}\, m_j \,. \tag{1.15}$$

We can now interpret the FIE as the continuous form of the matrix-vector multiplication. The latter is quite simple to understand if one considers the following analogies:

$x \rightarrow j$

$u \rightarrow i$

$\int_X \rightarrow \sum_j$

$d(u) \rightarrow \mathbf{d}$

$m(x) \rightarrow \mathbf{m}$

$K(u,x) \rightarrow \mathbf{A}$

Problems given by equation 1.13 will become computer algorithms given by expression 1.14. Remember that matrix-vector multiplication can be easily implemented via a code like this one

```
for i=1:N
 d(i) = 0;
  for j=1:M
   d(i) = d(i) + A(i,j) * m(j);
 end
end
```

The code to evaluate matrix-vector multiplication has two loops. The internal integration loop runs over j (or x) and the external look runs over the output variable i (or u). We will return to this critical analogy when considering the *adjoint* of the FIE operator and the multiplication of a vector with the transpose of the matrix $\mathbf{A}$. For the multiplication of a vector with the transpose of $\mathbf{A}$ (which is equivalent to integrating with an adjoint operator), the integration variable becomes the output variable, and the output variable becomes the integration variable. This concept is fundamental when writing computer codes to evaluate a forward linear operator and its associated adjoint operator. Stay tuned!!

Chapter 2

Inversion of Fredholm Integral equation

2.1 Solution of the FIE for a continuous model $m(x)$ and a finite number of accurate observations

We will continue with the study of continuous linear inverse problems and discuss the solution of the Fredholm Integral equation of the first kind

$$d(u) = \int_a^b K(u,x)\, m(x)\, dx \tag{2.1}$$

when only a finite number of observations are provided. We will also simplify the problem by assuming noise-free data (accurate data). Consider the case where the independent variable of the observation field $d(u)$ is given by $u_j\,, j = 1, \ldots, N$. Equation 2.1 becomes

$$d(u_j) = \int_a^b K(u_j,x)\, m(x)\, dx\,. \tag{2.2}$$

which can be simplified in the following manner

$$d_j = \int_a^b K_j(x)\, m(x)\, dx\,, \quad j = 1, \ldots, N\,. \tag{2.3}$$

Our task is to try to retrieve $m(x)$ (a function) from d_j (a finite number of observations).

2.2 Model averages via Dirichlet condition

We will present a simple attempt to estimate a quantity that represents m. First, we consider the retrieval of information about m via an average of the form

$$< m(x_0) >= \sum_{j=1}^{N} \alpha_j(x_0) d_j \,. \tag{2.4}$$

The data are linear in the model $m(x)$. Hence, one could consider that an average representing $m(x)$ at a point x_0 can be expressed via a linear combination of observations d_j. The coefficients $\alpha_j(x_0)\,, j = 1, \ldots, N$ are our new unknowns. We now substitute 2.3 in equation 2.4

$$\begin{aligned} < m(x_0) >= & \sum_{j=1}^{N} \alpha_j(x_0) \int_a^b K_j(x)\, m(x)\, dx \\ = & \int_a^b \sum_{j=1}^{N} \alpha_j(x_0) K_j(x)\, m(x)\, dx \\ = & \int_a^b A(x_0, x) m(x) dx \end{aligned} \tag{2.5}$$

where the averaging kernel $A(x_0, x)$ is given by

$$A(x_0, x) = \sum_{j=1}^{N} \alpha_j(x_0) K_j(x) \,. \tag{2.6}$$

Consider the highly hypothetical case where

$$A(x_0, x) = \delta(x - x_0) \,. \tag{2.7}$$

In the case mentioned above, it is easy to show that

$$\begin{aligned} < m(x_0) >= & \int_a^b A(x_0, x)\, m(x) dx \\ = & \int_a^b \delta(x - x_0)\, m(x) dx \\ = & m(x_0) \,. \end{aligned} \tag{2.8}$$

In other words, I have estimated from the data d_j the value $m(x)$ at $x = x_0$. One can repeat the process for a large number of points x_0 and, in this way, estimate the unknown $m(x)$. Don't feel

happy about the latter. I could construct an averaging function of the form $A(x_0, x) = \delta(x - x_0)$. We shall see that this is not possible. Let's see how one can, at least, attempt to turn $A(x_0, x)$ into something that resembles $\delta(x - x_0)$. We will try to find a delta-like $A(x_0, x)$ by finding the coefficients $\alpha_j(x_0)$ that minimize the following cost function [1]

$$\begin{aligned} S_D &= \int_a^b [A(x_0, x) - \delta(x - x_0)\,]^2 dx \\ &= \int_a^b [\sum_{j=1}^{N} \alpha_j(x_0) K_j(x) - \delta(x - x_0)\,]^2 dx\,. \end{aligned} \tag{2.9}$$

The constants $\alpha_j(x_0)$ are computed by evaluating the minimum of S_D

$$\frac{\partial S_D}{\partial \alpha_k(x_0)} = 0\,, \quad k = 1 \ldots N\,. \tag{2.10}$$

After a few mathematical steps, it is easy to show that S_D is minimized by the solution of the following linear system of equations

$$\sum_{j=1}^{N} \Gamma_{i,j} \alpha_j(x_0) = K_i(x_0)\,, \quad i = 1 \ldots N \tag{2.11}$$

with

$$\Gamma_{i,j} = \int_a^b K_i(x) K_j(x)\, dx\,. \tag{2.12}$$

Equation 2.11 can be written in a matrix-vector form as follows

$$\mathbf{\Gamma \alpha}(x_0) = \mathbf{g} \tag{2.13}$$

where $\mathbf{\Gamma}$ is the $N \times N$ matrix with elements given by $\Gamma_{i,j}$ (equation 2.12), and

$$\boldsymbol{\alpha}(x_0) = \begin{pmatrix} \alpha_1(x_0) \\ \alpha_2(x_0) \\ \vdots \\ \alpha_N(x_0) \end{pmatrix} \tag{2.14}$$

[1] The function S_D is the l_2 norm of the difference between $A(x_0, x)$ and $\delta(x - x_0)$. By minimizing S_D, we minimize the difference between $A(x_0, x)$ and $\delta(x - x_0)$ and therefore, this is an attempt to make $A(x_0, x)$ to look like $\delta(x - x_0)$.

and

$$\boldsymbol{g}(x_0) = \begin{pmatrix} K_1(x_0) \\ K_2(x_0) \\ \vdots \\ K_N(x_0) \end{pmatrix} \tag{2.15}$$

Finally, one can estimate the coefficients $\alpha_j(x_0)$ via the solution

$$\boldsymbol{\alpha}(x_0) = \boldsymbol{\Gamma}^{-1}\mathbf{g}\,. \tag{2.16}$$

Once the last system is solved, we compute the average $< m(x_0) >= \sum_{j=1}^{N} \alpha_j(x_0)\, d_j$ and the averaging kernel $A(x_0, x) = \sum_{j=1}^{N} \alpha_j(x_0) K_j(x)$ for a point of interest x_0. As you might imagine, this technique will not produce highly resolved results. The ideal averaging kernel should resemble a delta function centred at $x = x_0$. The kernels $K_j(x)$ are smooth functions. Constructing a delta function from a finite sum of smooth functions is not simple.

2.3 Determination of Earth's density distribution from two observations

We will discuss a classical problem in introductory courses to inverse problems (Gubbins, 2006). Assume an ideal earth model where the density is a function that depends on depth (radius) $\rho(r)$. Consider that two observations are available to sense the density distribution of Earth. These observations could have been obtained via astronomical measurements

$$M = 4\pi \int_0^a \rho(r)\, r^2\, dr \qquad \text{Earth's Mass} \tag{2.17}$$

$$I = \frac{8\pi}{3} \int_0^a \rho(r)\, r^4\, dr \quad \text{Earth's Moment of Inertia} \tag{2.18}$$

$a = 6371$ Km indicates the radius of the Earth.

$$M = 5.974 X 10^{24} kg$$

$$\frac{I}{a^2} = 1.975 X 10^{24} kg$$

Expressions 2.17 and 2.18 can be written down as follows after letting $x = \frac{r}{a}$

$$d_1 = \int_0^1 \rho(x) K_1(x)\, dx \tag{2.19}$$

$$d_2 = \int_0^1 \rho(x) K_2(x)\, dx \tag{2.20}$$

or

$$d_j = \int_0^1 m(x) K_j(x)\, dx\,, \quad j = 1, 2 \tag{2.21}$$

with $m(x) = \rho(x)\,, x = [0, 1]$. In other words, the problem has been reduced to an FIE. The Dirichlet condition discussed in the preceding section can be used to estimate, for instance, the average density at $x_0 = \frac{1}{2}$ ($r = \frac{a}{2}$).

$$< \rho(0.5) >= 7435 kg/m^3$$

The averaging kernels $A(x_0, x)$ for $x_0 = 0.4, 0.5, 0.6$ are displayed in Figure 2.1. It is clear that $< \rho(x_0) >$ is a low-resolution average because it results from the integration of a broad kernel that spans the whole Earth.

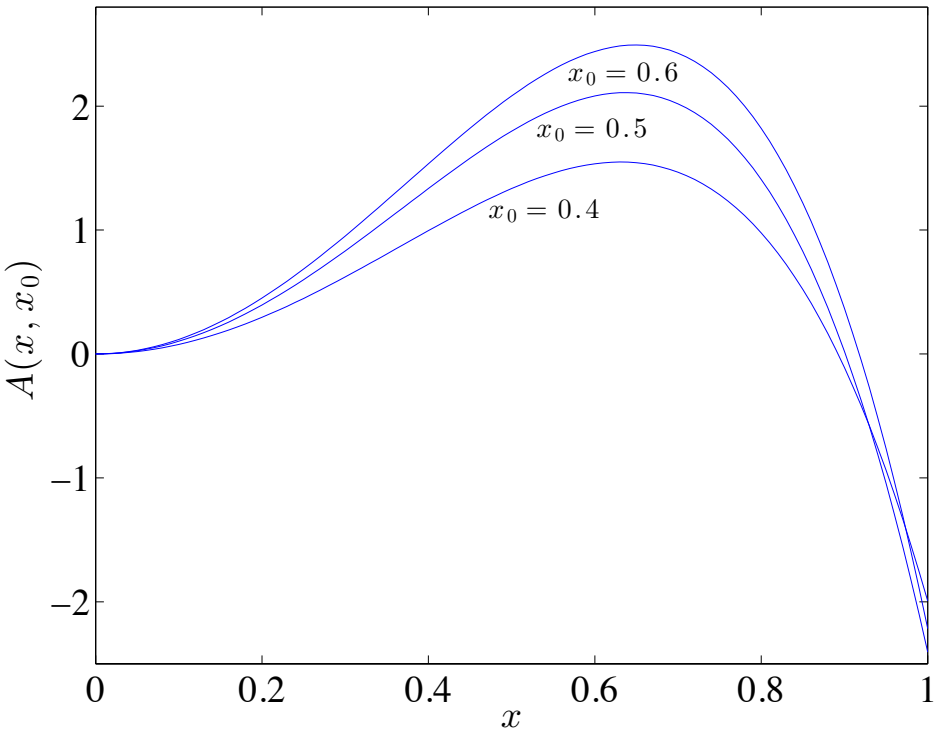


Figure 2.1: Averaging Dirichlet kernel for the Earth's density problem.

2.4 Inversion of $m(x)$ via the minimum norm solution

The data themselves are insufficient to estimate $m(x)$ in equation 2.3. We need an extra constraint. We will assume that among all possible solutions that solve the FIE, one can pick the minimum norm solution (we will extensively use this concept when dealing with discrete inverse problems). The minimum norm solution imposes regularity to the solution $m(x)$ and can be obtained by solving the following problem

$$\begin{aligned} \hat{m}(x) &= \arg\min_{m(x)} [\int_a^b m(x)^2\, dx\,] \\ \text{subject to } \; d(u_j) &= \int_a^b K_j(x)\, m(x)\, dx\,, \quad j = 1 \ldots N \end{aligned} \tag{2.22}$$

The problem is solved using the method of Lagrange multipliers by minimizing an augmented cost function J of the form

$$J(m) = \int_a^b m(x)^2\, dx \; + \; \sum_j \lambda_j [\, d_j - \int_a^b K_j(x)\, m(x)\, dx\,] \tag{2.23}$$

We will use variational calculus to estimate the minimum of J. First, we consider a perturbation to the model given by

$$m(x) \rightarrow m(x) + \delta m(x)\,.$$

We assume that $m(x)$ is a solution that minimizes $J(m)$. We can write[2]

$$\begin{aligned} J(m+\delta m) =& \int_a^b (m(x)+\delta m(x))^2\, dx \; + \; \sum_j \lambda_j [\, d_j - \int_a^b K_j(x)\, (m(x)+\delta m(x))\, dx\,] \\ =& J(m) + \int_a^b [2m(x) + \sum_{j=1}^N \lambda_j K_j(x)]\, \delta m(x)\, dx \end{aligned} \tag{2.24}$$

Therefore, δJ is given by

$$\delta J(m) = \int_a^b [2m(x) + \sum_{j=1}^N \lambda_j K_j(x)]\, \delta m(x)\, dx\,. \tag{2.25}$$

At the minimum, the perturbation of the cost function $\delta J = 0$ (Figure 2.2) for any small δm. Therefore,

$$m(x) = -\frac{1}{2} \sum_{j=1}^N \lambda_j K_j(x)\,. \tag{2.26}$$

[2]Consider $\delta m^2 = 0$.

The latter can also be written down in terms of the product of two vectors

$$m(x) = -\frac{1}{2}\boldsymbol{g}(x)^T\boldsymbol{\lambda} \tag{2.27}$$

We now introduce the last expression into equation 2.3

$$\begin{aligned} d_i =& -\frac{1}{2}\sum_{j=1}^{N}\lambda_j\int_a^b K_i(x)\,K_j(x)\,dx \\ =& -\frac{1}{2}\sum_{j=1}^{N}\lambda_j\Gamma_{i,j} \end{aligned} \tag{2.28}$$

where $\Gamma_{i,j}$ are the elements of the Gram matrix. The last equation can be written in matrix-vector form

$$\mathbf{d} = -\frac{1}{2}\boldsymbol{\Gamma}\boldsymbol{\lambda} \tag{2.29}$$

from where we can compute the vector of Lagrange multipliers

$$\boldsymbol{\lambda} = -2\boldsymbol{\Gamma}^{-1}\mathbf{d}\,. \tag{2.30}$$

The solution is given by

$$\begin{aligned} \hat{m}(x) =& -\frac{1}{2}\boldsymbol{g}(x)^T\boldsymbol{\lambda} \\ =& \boldsymbol{g}(x)^T\boldsymbol{\Gamma}^{-1}\mathbf{d} \end{aligned} \tag{2.31}$$

with

$$\boldsymbol{g}(x)^T = (K_1(x), K_2(x), \ldots, K_N(x))\,.$$

2.5 Minimum norm solution: Earth's density example

In the case of the Earth's density estimation example, one can use the afore-described procedure and obtain the following result

$$m(x) = \rho(x) = 40808x^2 - 44263x^4\,, \quad [Kg/m^3]\,.$$

Figure 2.4 portrays the minimum norm solution for the density. The result is quite peculiar. The density at the centre of the Earth is $\rho(0) = 0$. In addition, the density at the surface of the Earth $\rho(1) = -3455\,[kg/m^3$ (negative!!). The minimum norm solution has produced a non-physical result

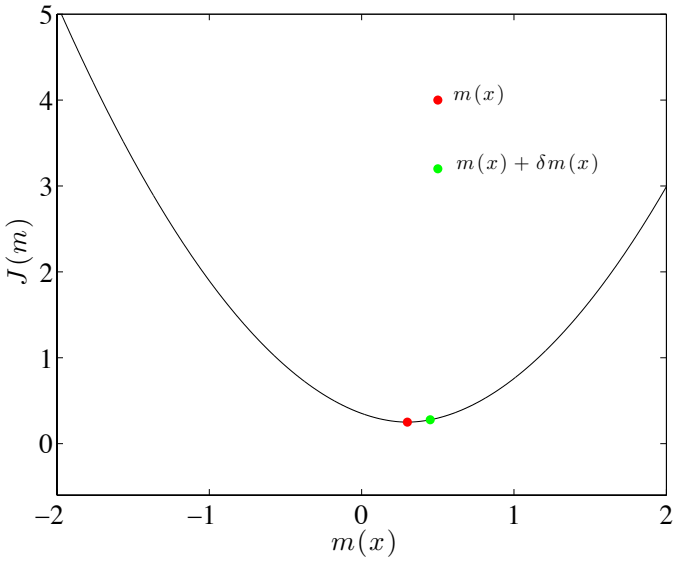


Figure 2.2: At the minimum, the perturbation $\delta J(m)$ is zero for any arbitrary small $\delta m(x)$.

that includes negative density values. What went wrong?. Well, nothing went wrong; we have constructed a solution fitting the data, our goal. The solution does not make sense because we have not used the appropriate model constraint. Evidently, the minimum norm solution for this particular problem was inadequate. A more realistic constraint would enforce a solution where the density is strictly positive.

Another potential remedy for avoiding non-physical solutions is minimizing the density perturbation norm rather than the density itself. Consider you believe that the density can be approximated by an initial solution given by $m_0(x) = \rho_0(x)$. In this case, one can compute a minimum perturbation solution. In other words, we apply the constraint to a new variable given by $\eta(x) = m(x) - m_0(x)$. This is equivalent to minimizing the function

$$\begin{aligned} \hat{\eta}(x) &= \arg\min_{\eta(x)} \left[\int_a^b \eta(x)^2 \, dx \right] \\ \text{subject to } \; d(u_j) &= \int_a^b K_j(x) \left(\eta(x) + m_0(x) \right) dx \,, \quad j = 1 \ldots N \end{aligned} \tag{2.32}$$

The procedure described above to compute the minimum norm solution can be adapted to estimate $\eta(x)$. For this particular case, I have selected the following *a priori* density model

$$m_0(x) = \rho_0(x) = 8200 - 5400x \,, \quad [Kg/m^3] \,.$$

In other words, we assume that the density increases linearly with depth x. We also assumed a

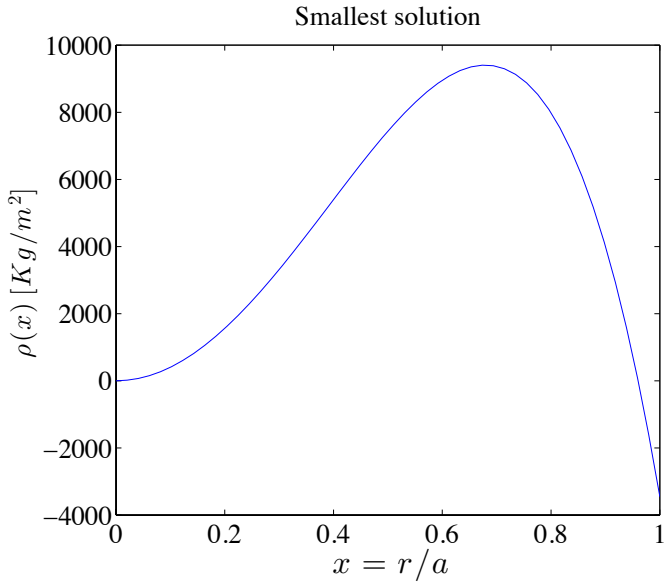


Figure 2.3: Minimum norm solution (smallest solution) for the density distribution in the interior of the Earth. The solution honours two observations (Earth's mass and moment of inertia). Notice that the density becomes negative at the surface. This is a mathematical solution that honours the two observations. This is a non-physical solution.

surface value of $2800\,kg/m^3$. In this case, the minimum norm perturbation yields

$$\eta(x) = 14353x^2 - 16911x^4\,.$$

Hence,

$$\begin{aligned}\rho(x) =& \rho_0(x) + \eta(x)\\ =& 8200 - 5400x + 14353x^2 - 16911x^4\,.\end{aligned}$$

The density at the Earth's centre is $8200\,kg/m^2$, and now the density at the surface is $242\,kg/m^2$. The density is too low to be a realistic result, but at least it is nonnegative. The solution (minimum norm perturbation) is portrayed in Figure 2.4.

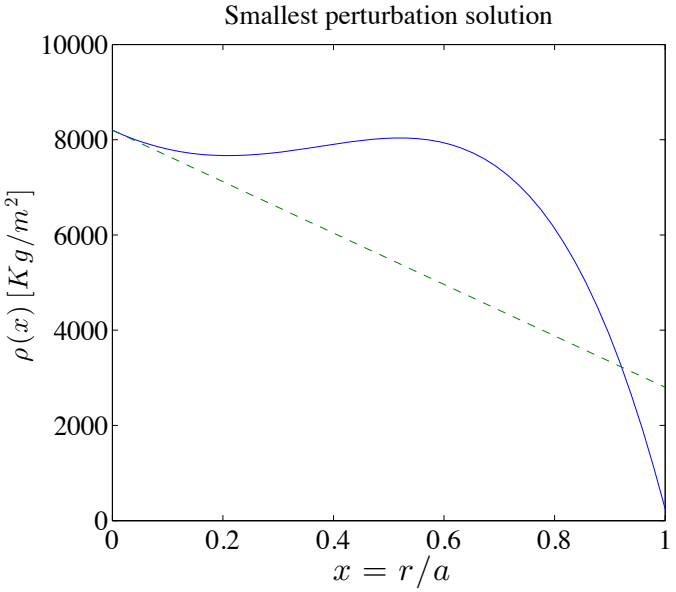


Figure 2.4: Solid line: minimum deviatory solution (smallest perturbation) for the density distribution in the interior of the Earth. The solution honours two observations (Earth's mass and moment of inertia). Dashed line: a priori model of the density $\rho_0(x)$ used for this example.

Chapter 3

Inverse Problems: The discrete case with accurate data

We have discussed the linear inverse problem as the solution of linear integral operator. We will now consider scenarios with discrete models. In order words, a continuous model is discretized to form a vector of model parameters. Let us assume that we have a finite amount of data d_j, $j = 1, \ldots N$, the unknown model m after discretization using layers or cells is given by the vector $\mathbf{m}$ with elements m_i, $i = 1, \ldots M$. The expression of the forward model is given by

$$\mathbf{d} = \mathbf{G}\mathbf{m} \tag{3.1}$$

where $\mathbf{G}$ is a $N \times M$ matrix that arises after discretizing the Kernel of the linear forward problem. If $M > N$, many solutions solve the system of equations. In other words, the system is *underdetermined*.

3.1 Minimum norm solution

An underdetermined problem has an infinite number of solutions. We will retrieve a minimum norm model. The problem is posed as follows:

Find a model $\mathbf{m}$ that minimizes the function

$$\mathbf{m}^T\mathbf{m} \tag{3.2}$$

subject to data constraints

$$\mathbf{d} = \mathbf{G}\mathbf{m}\,. \tag{3.3}$$

We notice that model norm $\|\mathbf{m}\|_2^2 = \mathbf{m}^T\mathbf{m}$, is actually, the squared l_2 norm of the model. By definition, the l_2 norm is given by

$$l_2 = \|\mathbf{m}\|_2 = \sqrt{\mathbf{m}^T\mathbf{m}}\,.$$

Each observation in equation 3.3 provides one constraint. Therefore, we have N constraints. The constrained minimization problem is solved using the method of Lagrange multipliers. In this case, we minimize the following cost function

$$J = \mathbf{m}^T\mathbf{m} + \boldsymbol{\lambda}^T(\mathbf{d} - \mathbf{G}\mathbf{m}) \tag{3.4}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers λ_i, $i = 1, \ldots, N$. Evaluating the derivative of J with respect to the unknowns $\mathbf{m}$ and $\boldsymbol{\lambda}$ and setting them to zero yields the following two equations

$$\begin{aligned} 2\mathbf{m} + \mathbf{G}^T\boldsymbol{\lambda} &= 0 & (3.5)\\ \mathbf{G}\mathbf{m} - \mathbf{d} &= \mathbf{0}\,. & (3.6) \end{aligned}$$

You can play for a while with the last two equations to obtain the expression for the minimum norm solution

$$\mathbf{m}_{MN} = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{d}\,. \tag{3.7}$$

Note that we have assumed that $\mathbf{G}\mathbf{G}^T$ is invertible. In other words, $rank(\mathbf{G}\mathbf{G}^T) = N$. Once we have computed the solution $\mathbf{m}_{MN}$, the solution can be used to predict the data

$$\begin{aligned} \mathbf{d}^{pred} &= \mathbf{G}\mathbf{m}_{MN} \\ &= \mathbf{G}\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{d} \\ &= \mathbf{d}\ . \end{aligned} \tag{3.8}$$

This result was expected, given that the solution was designed to fit the data.

3.1.1 Example

We assume that our data d_j can be written as a linear integral operator of the form

$$d_j = d(r_j) = \int_0^L e^{-\alpha(x-r_j)^2} m(x)dx\,. \tag{3.9}$$

Note that r_j can indicate the position of an observation point where we have a sensor. We can discretize the above expression using the trapezoidal rule

$$d(r_j) = \sum_{k=0}^{M-1} \Delta x\, e^{-\alpha(r_j - x_k)^2}\ \ m(x_k)\,,\ \ j = 0 \ldots N-1\,. \tag{3.10}$$

Further, we assume the measurement points r_j corresponds N observation distributed in $[0, L]$, then

$$r_j = j \,.\, L/(N-1)\,, \;\; j = 0 \ldots N-1\,.$$

The above system can be written down as follows

$$\mathbf{d} = \mathbf{G}\mathbf{m} \tag{3.11}$$

In our example, we assume a true solution given by the profile $\mathbf{m}$ portrayed in Figure 3.1a. The corresponding data are displayed in Figure 3.1b. For this toy problem, I have chosen the following parameters:

$$\alpha = 0.8, \quad L = 100., \quad N = 20, \quad M = 50$$

The following script is used to compute the Kernel $\mathbf{G}$ and to generate the observations $\mathbf{d}$. Figure 3.1c shows the minimum norm solution. Similarly, Figure 3.1d shows the predicted data. Notice that the solution fits the data (as it should) but it does not resemble the true profile given in Figure 3.1a. The minimum norm solution is too oscillatory and out next task is to device a regularization technique to minimize the oscillatory behaviour observed in Figure 3.1a.

Script 1

```
% Define model m
 M = 50;
 m = zeros(M,1);
 m(10:14,1) = 1.;
 m(15:26,1) = -.3;
 m(27:34,1) = 2.1;

% Discrete kernel G
 N = 20;
 L = 100;
 alpha = .8
 x = (0:1:M-1)*L/(M-1);
 dx = L/(M-1);
 r = (0:1:N-1)*L/(N-1);
 for j=1:M
 for k=1:N
 G(k,j) = dx*exp(-alpha*abs(r(k)-x(j))^2);
 end
 end

% Compute data
 d = G*m;
```

The MATLAB script for the minimum norm solution is given by:

Function min_norm_sol.m

```
function [m_sol,d_pred] = min_norm_sol(G,d);
%
% Given a discrete Kernel G and the data d, we compute the
% minimum norm solution of the inverse problem d = Gm.
% m_sol: solution (minimum norm solution)
% d_pred: predicted data
%

 m_sol = G'*inv(G*G')*d;
 d_pred = G* m_sol
```

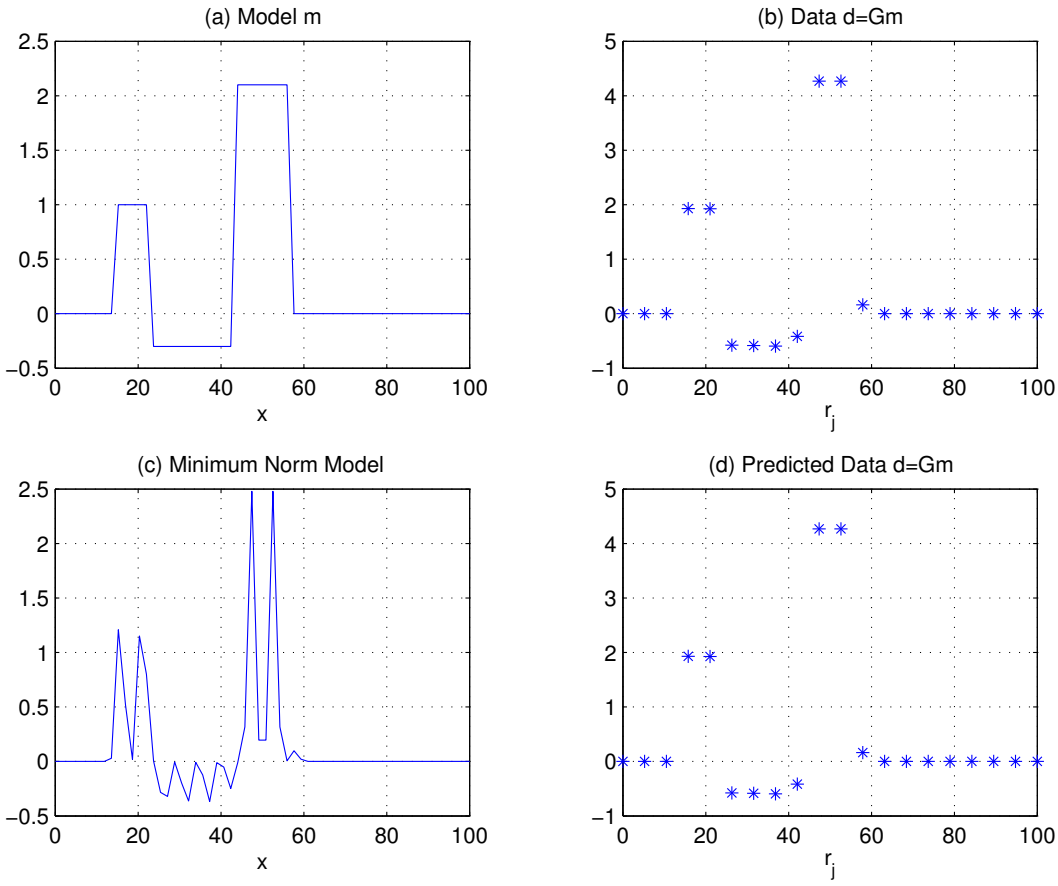


Figure 3.1: a) Model. d) Data. c) Minimum norm solution. d) Predicted data.

In the following script, I present one way to display our results.

Script 3

```
% This is the recipe I used to prepare Figures 1 a-d.

 figure(1); clf;

 subplot(221);
 plot(x,m);
 title('(a) Model m');
 xlabel('x') ;grid;

 subplot(222);
 plot(r,d,'*');
 title('(b) Data d=Gm');
 xlabel('r_j'); grid

 subplot(223);
 plot(x,m_est); t
 title('(c) Minimum Norm Model ');
 xlabel('x'); grid;

 subplot(224);
 plot(r,d,'*');
 title('(d) Predicted Data d=Gm');
 xlabel('r_j');grid;
```

3.2 Weighted minimum norm solution

The minimum norm solution can become too oscillatory. We introduce a weighting function into the model norm to alleviate this problem. We define the following weighted norm

$$||\mathbf{W}\,\mathbf{m}||_W^2 = \mathbf{m}^T\,\mathbf{W}^T\,\mathbf{W}\,\mathbf{m}\,. \tag{3.12}$$

This new norm is minimized subject to data constraints $\mathbf{Gm} = \mathbf{d}$. It is easy to show that the minimum weighted norm solution is given by

$$\mathbf{m}_{MWN} = \mathbf{Q}\,\mathbf{G}^T(\mathbf{G}\,\mathbf{Q}\,\mathbf{G}^T)^{-1}\mathbf{d} \tag{3.13}$$

where

$$\mathbf{Q} = (\mathbf{W}^T\,\mathbf{W})^{-1} \tag{3.14}$$

This solution is called the minimum weighted norm solution. In Figures 3.2 and 3.2, I computed the minimum norm weighted solution using $\mathbf{W} = \mathbf{D}_1$ and $\mathbf{W} = \mathbf{D}_2$, that is the first and second derivative operators, respectively.

Smoothing with first-order derivatives is accomplished by choosing the following operator (for $M = 5$)

$$\mathbf{D}_1 = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.15}$$

Similarly, one could select second-order derivatives to include smoothing in the solution. For $M = 5$ we have

$$\mathbf{D}_2 = \begin{pmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

Remember that there is freedom in selecting the operator at the boundaries (to be discussed in class).

The following MATLAB script was used to compute the solution with 1D first order derivative smoothing (the trick I used to calculate the first-order derivative is not very transparent - ask me if you don't understand)

function min_norm_sol_d1.m

```
function [m_sol,d_pred] = min_norm_sol(G,d);

% Minimum norm solution with smoothing using D1
 [N,M] = size(G);
 W = convmtx([1,-1],M);
 D1 = W(1:M,1:M);
 Q = inv(D1'*D1);
 m_sol= Q*G'*inv(G*Q*G')*d;
 d_pred = G*m_sol;
```

Similarly, you can use second-order derivatives

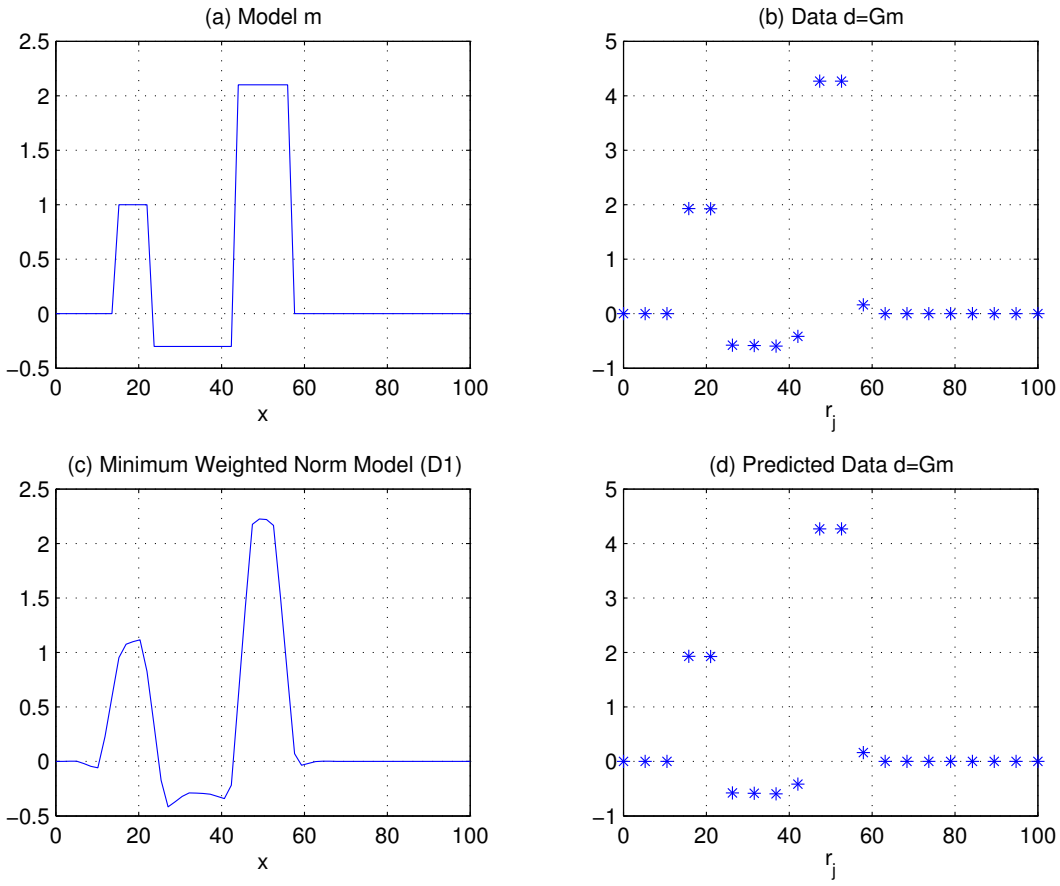


Figure 3.2: a) True model. b) Data (observations). c) Inversion using the minimum weighted norm solution. The smoothing operator is $\mathbf{D}_1$ (First order derivative). d) Predicted data.

function min_norm_sol_d2.m

```
function [m_sol,d_pred] = min_norm_sol(G,d);
% Minimum norm solution with smoothing using D1
 [N,M] = size(G);
 W = convmtx([1,-2,1],M);
 D2 = W(1:M,1:M);
 Q = inv(D2'*D2);
 m_sol = Q*G'*inv(G*Q*G')*d;
 d_pred = G*m_sol;
```

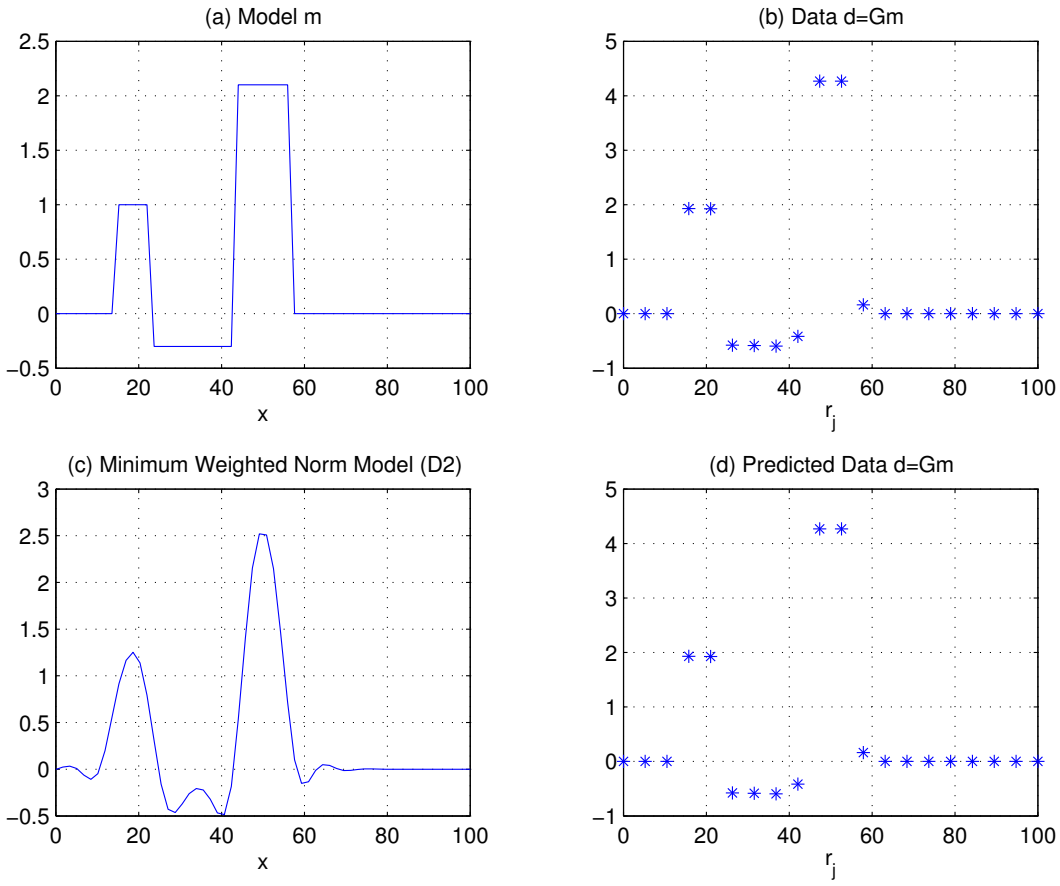


Figure 3.3: a) True model. b) Data (observations). c) Inversion using minimum weigher norm solution. The smoothing operator is $\mathbf{D}_2$ (Second order derivative). d) Predicted data.

The derivative operators $\mathbf{D}_1$ and $\mathbf{D}_2$ behave like high pass filters. If you are not convinced, please compute the amplitude spectrum of a random time series after and before applying one of these operators.

Question: Why is it that when one wants to find a smooth solution, we must use weights that behave like a high-pass filter?

Chapter 4

Linear Inverse Problems: The Discrete case with inaccurate data

4.1 Least squares problems with quadratic regularization

In this section, we will explore the solution of the discrete inverse problem in a situation where the data are contaminated with errors. Attempting to fully fit the data is not a good idea. In fact, rather than trying a perfect fit for each observation as we did in Chapter 2, we will fit the data with some tolerance. In the previous Chapter, we worked with the *Accurate Data Problem* and defined the minimum norm solution via the following problem

$$\text{Minimize } \mathbf{m}^T\mathbf{m} \text{ [This is the model norm]}$$

$$\text{Subject to } \mathbf{d} - \mathbf{G}\mathbf{m} = \mathbf{0}\,.$$

When data are contaminated with errors, the exact fitting goal must be replaced by the following goal

$$\mathbf{G}\mathbf{m} - \mathbf{d} \approx \mathbf{0}\,.$$

A model like the above can also be written as

$$\mathbf{d} = \mathbf{G}\mathbf{m} + \mathbf{e}\,,$$

where $\mathbf{e}$ indicates the error or noise vector. This quantity is unknown. The minimum norm solution, in this case, becomes the solution to the following optimization problem

$$\text{Minimize } J = \mathbf{m}^T\mathbf{m}$$

$$\text{Subject to } \; ||\mathbf{d} - \mathbf{G}\mathbf{m}||_2^2 = \epsilon$$

It is clear that now rather than having one constraint per observation, we have a single global constraint for all the observations. Notice in the previous equation, we have used the l_2 norm (squared) as a measure of distance for the errors in the data; we will see that this also implies that the errors are considered to be distributed according to the normal distribution (Gaussian errors). Before continuing with the analysis, we recall

$$\|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 = \|\mathbf{e}\|_2^2$$

which in matrix/vector notation can also be expressed as

$$\|\mathbf{e}\|_2^2 = \mathbf{e}^T\mathbf{e}\,.$$

We now minimize the cost function J given by

$$\begin{aligned} J &= \mu\,\text{Model Norm} + \text{Misfit} \\ &= \mu\,\mathbf{m}^T\mathbf{m} + \mathbf{e}^T\mathbf{e} \\ &= \mu\,\mathbf{m}^T\mathbf{m} + (\mathbf{d} - \mathbf{G}\mathbf{m})^T(\mathbf{d} - \mathbf{G}\mathbf{m}) \end{aligned} \tag{4.1}$$

The solution is now obtained by minimizing J with respect to the unknown $\mathbf{m}$. This requires some algebra, and I will give you the final solution[1]

$$\begin{aligned} \tfrac{d\,J}{d\mathbf{m}} &= 0 \\ &= (\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})\mathbf{m} - \mathbf{G}^T\mathbf{d} = \mathbf{0}\,. \end{aligned} \tag{4.2}$$

The minimizer is given by[2]

$$\mathbf{m}_\mu = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})^{-1}\mathbf{G}^T\mathbf{d}\,. \tag{4.3}$$

This solution is often called the *damped least-squares solution*. Notice that the solution structure looks like the solution we obtained when we solve a least squares problem. A simple identity permits one to make equation (4.3) resemble the minimum norm solution (see Part 3)

$$\text{Identity} \quad (\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})^{-1}\mathbf{G}^T = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T + \mu\mathbf{I})^{-1}\,. \tag{4.4}$$

[1] This is also done in assignment 1

[2] Also called *minimum norm least-squares solution*.

Therefore, equation (4.3) can be re-expressed as

$$\mathbf{m}_\mu = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T + \mu\mathbf{I})^{-1}\mathbf{d}\,. \tag{4.5}$$

It is important to note that the previous expression reduces to the minimum norm solution for exact data when $\mu = 0$. However, the identity given above is only valid for $\mu > 0$ to guarantee the invertibility of $(\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})$ and $(\mathbf{G}\mathbf{G}^T + \mu\mathbf{I})$.

4.2 The tradeoff paramter μ

The importance of *infamous* parameter μ can be seen from the cost function J

$$J = \mu\,\text{Model Norm} + \text{Misfit} \tag{4.6}$$

- Large μ means more weight (importance) is given to minimize the model norm rather than the misfit.
- Small μ means that the misfit is the leading term in the minimization; the model norm becomes less critical.
- You can think that we are trying to simultaneously achieve two *goals*:

  Model Norm Reduction (Stability, we do not want highly oscillatory solutions)

  Misfit Reduction (We want to honour our observations)

We will explore that these two goals cannot be simultaneously achieved, which is why we often call μ a *tradeoff parameter*. The parameter μ receives different names according to the scientific background of the user

1. Statisticians: Hyper-parameter, tradeoff parameter, Ridge regression parameter,
2. Mathematicians: Regularization parameter, Penalty parameter, Stabilization parameter, Damping parameter, Tikhonov regularization parameter
3. Signal Processing: Tradeoff parameter, Pre-whitening parameter

Finding μ can be a big deal for large-scale inverse problems, and heuristic methods can generally be used. For instance, one can visualize data residuals for evidence of over or under-fitting.

4.3 Trade-off diagrams

A tradeoff diagram displays the model norm versus the misfit for different values of μ. Consider a solution for a given parameter μ

$$\mathbf{m}_\mu = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{I})^{-1}\mathbf{G}^T\mathbf{d} \tag{4.7}$$

This solution can be used to compute the model norm

$$\text{Model Norm}_\mu = {\mathbf{m}_\mu}^T\mathbf{m}_\mu \tag{4.8}$$

and the misfit

$$\text{Misfit}_\mu = (\mathbf{G}\mathbf{m}_\mu - \mathbf{d})^T(\mathbf{G}\mathbf{m}_\mu - \mathbf{d}) \tag{4.9}$$

by varying μ, one can compute a curve showing Model Norm$_\mu$ vs Misfit$_\mu$. Sometimes, the mathematical literature calls this curve an *L-curve*.

Example

We use the toy example in Chapter 3 to examine the tradeoff parameter's influence on our inverse problem's solution. But first, I will add noise to the synthetic data. Below is the script that shows how to contaminate accurate data with noise:

add_noise.m

```
%Add noise to synthetic data
 dc = G*m;
 s = 0.1;
 d = dc + s*randn(size(dc));
```

The Minimum Norm least-squared solution is obtained with the following script:

function dls_1.m

```
function [m_est,d_pred] = dls_1(G,d,mu);
%Damped LS solution with GG'
 I = eye(N);
 m_est = G'*((G*G'+mu*I)\d);
 d_pred = G*m_est;
```

Remember the identity (equation 4.4). The solution can also be written in the following form:

function dls_2.m

```
function [m_est,d_pred] = dls_2(G,d,mu);
%Damped LS solution using  G'G
 I = eye(M);
 m_est = (G'*G+mu*I)\ (G'*d);
 d_pred = G*m_est;
```

Inversion of the noisy data using the least squares minimum norm solution for $\mu = 0.05$ and $\mu = 5.$ are portrayed in Figures 4.3 and 4.3, respectively. Notice that a small value of μ leads to data over-fitting. In other words, we are attempting to reproduce the noisy observations as if they were accurate. This is not a good idea since over-fitting can force the creation of non-physical features in the solution, such as highly oscillatory solutions that become difficult to interpret. The tradeoff curve for this problem is given by Figure 4.3.

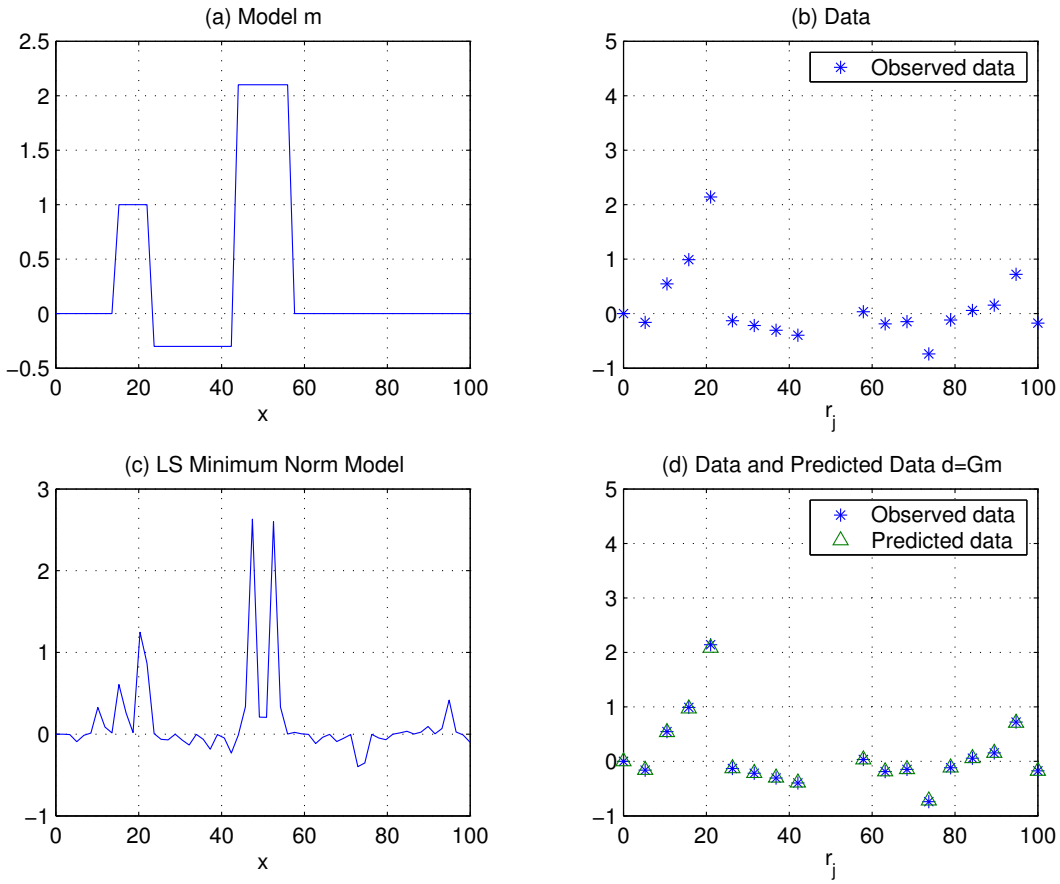


Figure 4.1: a) Model. d) Data. c) LS Minimum norm least-squares solution. d) Predicted data and observed data. In this example, I used a small tradeoff parameter $\mu = 0.005$. Notice that the predicted data reproduce quite well the noisy observations. Not a good idea since you want to avoid fitting the noise.

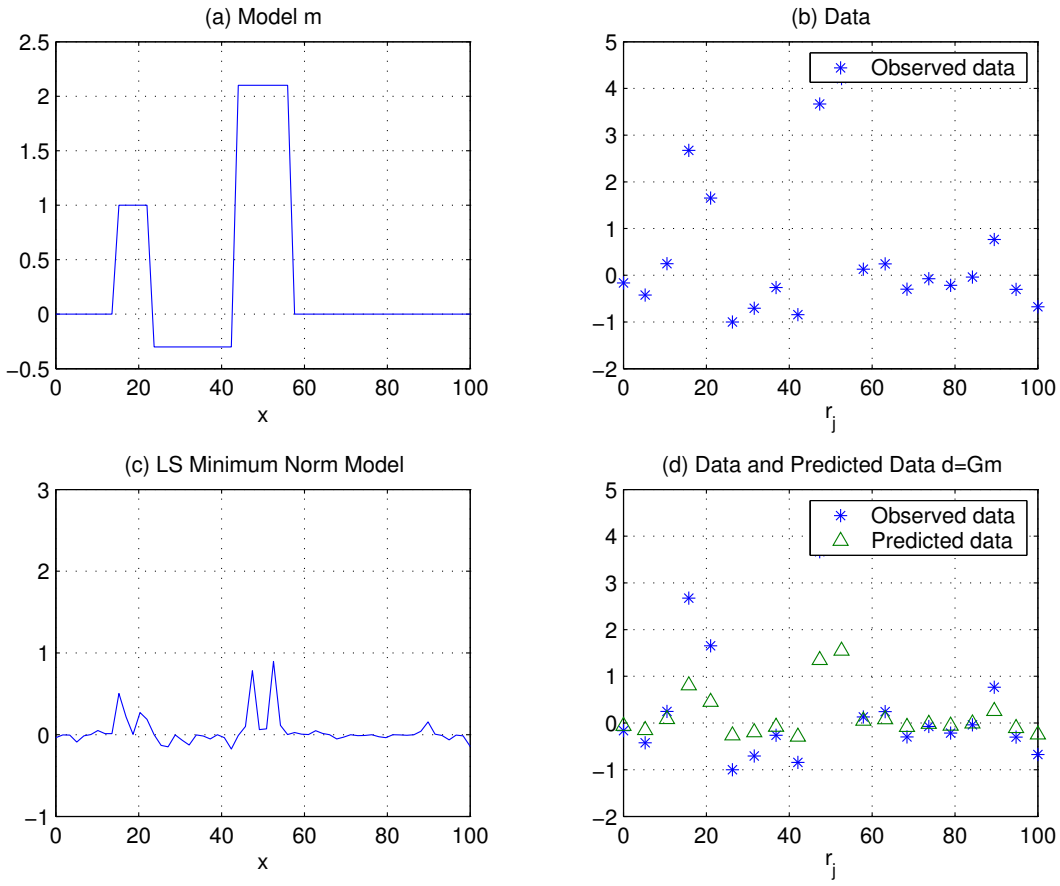


Figure 4.2: a) Model. d) Data. c) LS Minimum norm least-squares solution. d) Predicted data and observed data. In this example, I used a small tradeoff parameter $\mu = 5$. Notice that the predicted data does not try to reproduce the noisy observations.

Script for tradeoff curve

```
% Compute tradeoff curves
% mu is varied in logarithmic scale
%
mu_min= log10(0.5);
mu_max = log10(100);
Nmu = 11;

for k=1:Nmu;
 mu(k) = mu_min + (mu_max-mu_min)*(k-1)/(Nmu-1);
 mu(k) = 10^mu(k);
 m_est = G'*((G*G'+mu(k)*I)\d);
 dp = G*m_est;
 Misfit(k) = (dp-d)'*(dp-d);
 Model_Norm(k) = m_est'*m_est;
end;

% **** Make a Very Fancy Plot and Impress your Friends *****
 figure(1); clf;

 plot(Model_Norm,Misfit,'*'); hold on;
 plot(Model_Norm,Misfit  );

 for k=1:Nmu
 say = strcat('  \mu=',num2str(mu(k)));
 text(Model_Norm(k),Misfit(k),say);
 end;

 xlabel('Model Norm')
 ylabel('Misfit'); grid
```

4.4 Smoothing in the presence of noise

We consider adding a weighting matrix to the regularization term (the model norm)

$$\begin{aligned} J &= \mu \,\text{Model Norm} + \text{Misfit} \\ &= \mu\,(\mathbf{Wm})^T(\mathbf{Wm}) + \mathbf{e}^T\mathbf{e} \\ &= \mu\,\mathbf{m}\mathbf{W}^T\mathbf{W}\mathbf{m} + (\mathbf{Gm}-\mathbf{d})^T(\mathbf{Gm}-\mathbf{d}) \end{aligned} \tag{4.10}$$

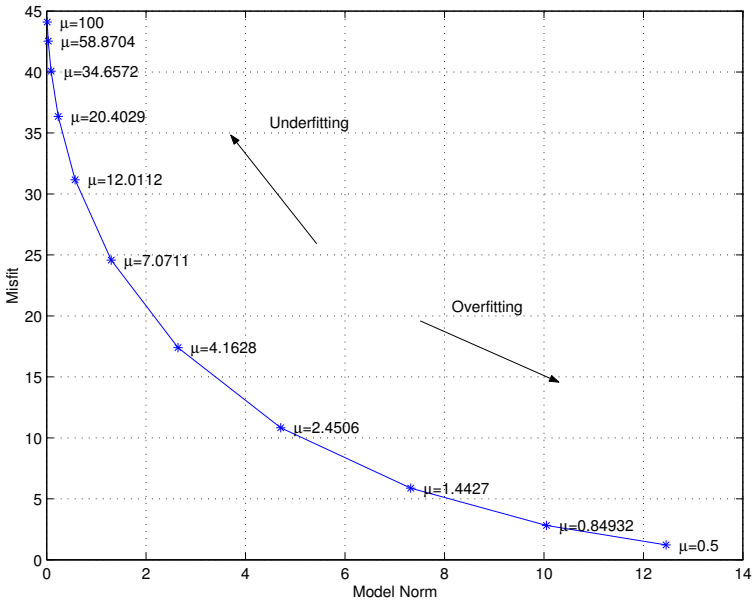


Figure 4.3: Tradeoff diagram for our toy example. Which value of μ would you pick?

where $\mathbf{W}$ can be a matrix of first or second-order derivatives, as shown before. The minimization of J leads to the least-squares minimum weighted norm solution

$$\begin{aligned} \frac{d\,J}{d\mathbf{m}} &= 0 \\ &= (\mathbf{G}^T\mathbf{G} + \mu\mathbf{W}^T\mathbf{W})\mathbf{m} - \mathbf{G}^T\mathbf{d} = \mathbf{0}\,. \end{aligned} \tag{4.11}$$

or,

$$\mathbf{m}_\mu = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{W}^T\mathbf{W})^{-1}\mathbf{G}^T\mathbf{d} \tag{4.12}$$

function dls_d1.m

```
function [m_est,d_pred,Misfit,Model_norm] = dls_d1(G,d,mu);
% Damped LS solution with first derivative regularization
%
 [N,M] = size(G);
 W = convmtx([1,-1],M);
 W1 = W(1:M,1:M);
 m_est = (G'*G+mu*W1'*W1)\(G'*d);
 d_pred = G*m_est;
 Model_Norm = m_est'*m_est;
 Misfit = (d-d_pred)'*(d-d_pred);
```

4.5 Recipe to compute μ

Consider that the variance of the noise is known and given by σ^2. In seismology, you could use data before the arrival of the P wave to analyze ambient noise and have a rough estimate of σ^2. First, you can use the solution $\mathbf{m}_\mu$ to estimate the predicted data

$$\mathbf{d}_\mu^{pred} = \mathbf{G}\mathbf{m}_\mu$$

the error can be estimated as well

$$\mathbf{e}_\mu = \mathbf{d}_\mu^{pred} - \mathbf{d}\,.$$

One can now estimate

$$\hat{\sigma}^2 = \frac{||\mathbf{e}_\mu||_2^2}{N}$$

(the estimator of the variance of the noise) for different values of μ. The optimum μ is the one that yields $\hat{\sigma}^2 \approx \sigma^2$. In other ways, one selects μ such that the estimated data error has a variance close to the assumed noise variance. This procedure assumes we know the variance of the noise in the data, which could be a problem in many situations. Other tradeoff parameter selection methods exist that do not require assuming a known noise variance (e.g. the generalized cross-validation method).

4.6 Edge Preserving Regularization (EPR)

Smoothing tends to blur edges. How can we preserve edges or structural boundaries in our inverted models? One solution entails adopting non-quadratic regularization functions. For instance, we could minimize a quadratic misfit in conjunction with a non-quadratic regularization term. In this particular case, we minimize the following cost function

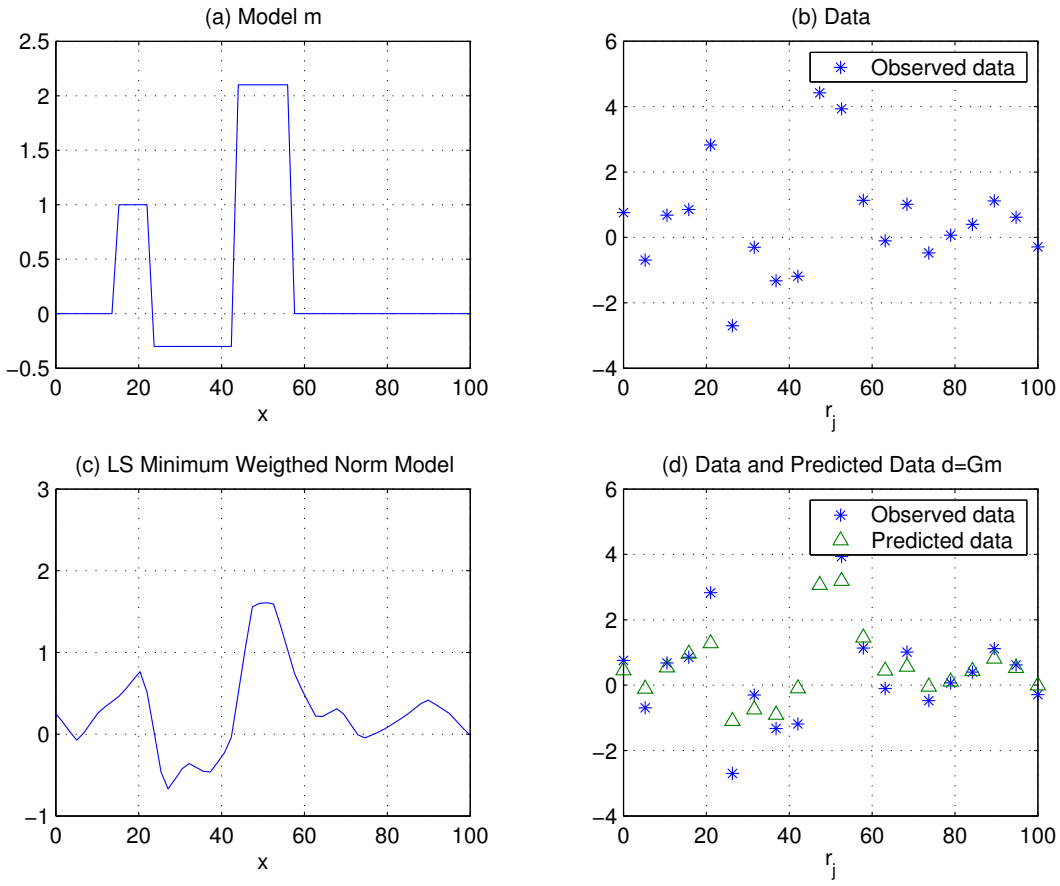


Figure 4.4: a) Model. d) Data. c) Least-squares Minimum Weighted norm solution. d) Predicted data and observed data. In this example, I used a tradeoff parameter $\mu = 5$. Notice that the predicted data does not try to reproduce the noisy observations. In this example the model norm is $||\mathbf{Wm}||_2^2$ where $\mathbf{W}$ is the first order derivative operator $\mathbf{D}_1$. This example was computed with the script `function dls_d1.m`

$$J_{EP} = ||\mathbf{Gm} - \mathbf{d}||^2 + \mu\Phi(\mathbf{m})$$

Where Φ is an edge-preserving regularization function of the form

$$\Phi(\mathbf{m}) = \sum_i \ln(1 + (\mathbf{D}_1\mathbf{m}/\delta)_i^2)\,.$$

Notice that now the regularization term (Φ) is non-quadratic[3].

[3]We will study non-quadratic regularization in detail in a forthcoming chapter

The EP solution involves the iterative solution of the following system of equations

$$\mathbf{m}^k = (\mathbf{G}^T\mathbf{G} + \mu\mathbf{Q}^{k-1})^{-1}\,\mathbf{G}^T\mathbf{d}$$

where $\mathbf{Q}$ is a diagonal matrix that depends on $\mathbf{m}$. The index k indicates iteration number.

Figure 4.6 shows a profile $\mathbf{m}$ and associated data $\mathbf{d}$ computed using the toy example in the MATLAB script. Figure 4.6 compares classical quadratic regularization with first and second-order derivative operators and EP regularization. It is clear that the edge-preserving solution shows more resemblance to the original profile.

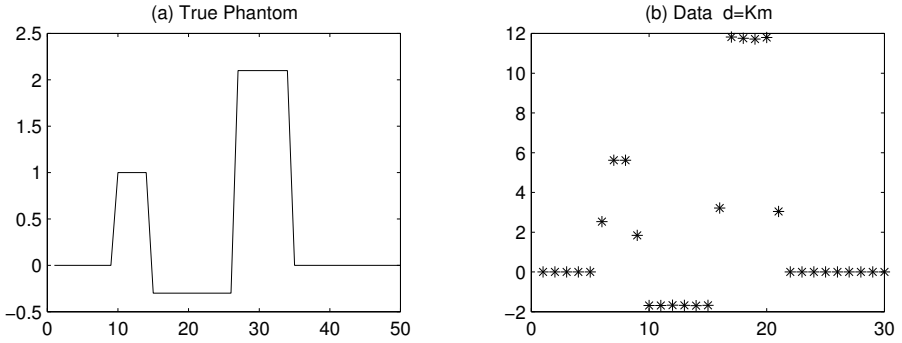


Figure 4.5: a) True profile with edges. d) Data to test EP regularization inversion.

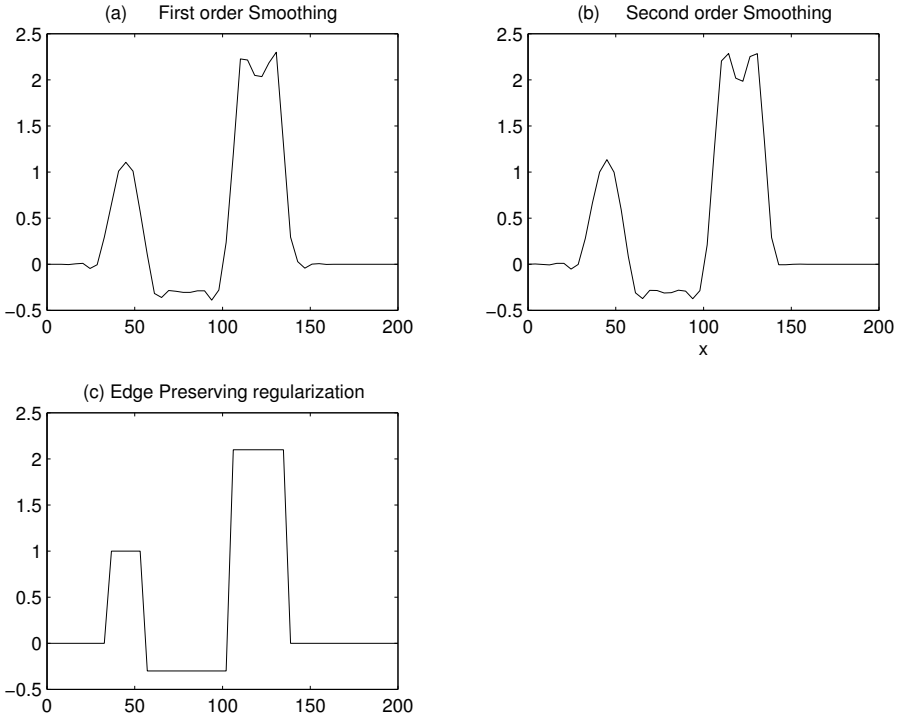


Figure 4.6: a) and b) Smooth solution obtained with quadratic regularization (first and second order derivative smoothing) c) Edge preserving solution.

Chapter 5

Robust regression and Sparsity

Non-Robust regression (LS fitting)

We consider a simple overdetermined problem of the form

$$\mathbf{d} - \mathbf{G}\mathbf{m} = \mathbf{n}\,, \tag{5.1}$$

where $M < N$ and the rank of the matrix is M (overdetermined problem). The least-squares solution assumes that errors (noise term) are Gaussian. If we further assume that the errors are uncorrelated, the maximum likelihood solution for $\mathbf{m}$ is given by the least-squares estimator

$$\mathbf{m}_{ls} = \arg\min_{m} \|\mathbf{G}\mathbf{m} - \mathbf{d}\|_2^2\,. \tag{5.2}$$

with solution given by

$$\mathbf{m}_{ls} = (\mathbf{G}^T\mathbf{G})^{-1}\,\mathbf{G}^T\mathbf{d}\,. \tag{5.3}$$

In Figure 5.1 (right), we see the problem when we attempt to fit a line using the least-squares method (L_2 regression). In this example, the data were contaminated with outliers. In other words, you may have Gaussian errors for most samples, but some samples are contaminated with erratic or gross errors[1].

In this case, the data are contaminated by outliers that distort the error distribution. When using least-squares fitting, these outliers are taken as *good* observations and the resulting regression is biased by the presence of outliners. A remedy for this problem is to use a robust regression algorithm. In other words, a procedure that deliberately attempts to ignore the presence of outliers.

[1] significant isolated errors often called outliers

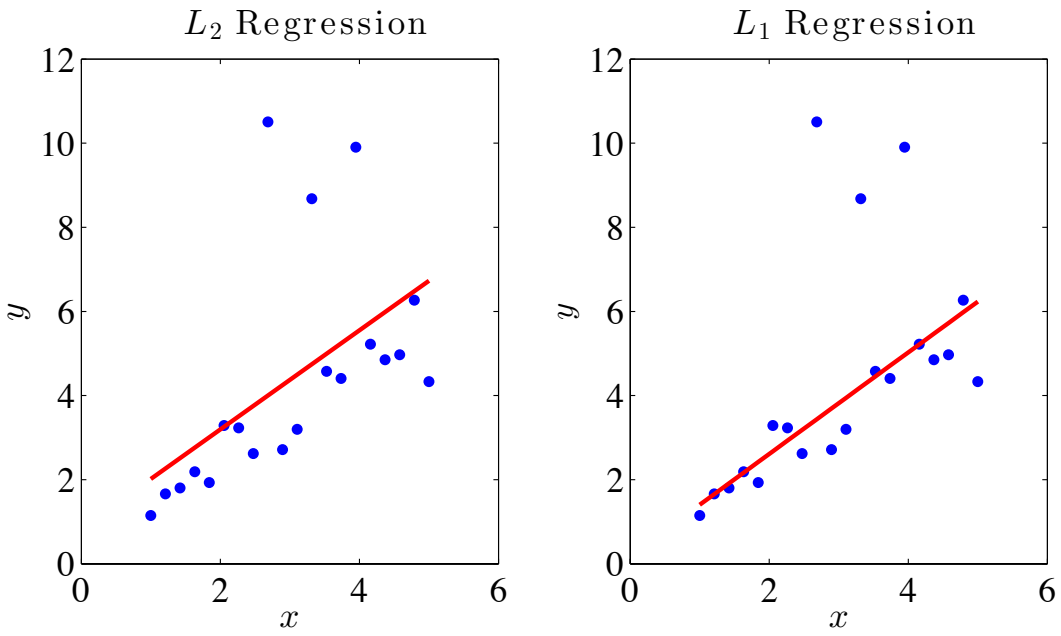


Figure 5.1: L_2 (non-robust) versus L_1 (robust) regression.

Robust regression

To diminish the influence of outliers in the solution of the linear regression problem, we propose to solve

$$\mathbf{m}_r = \arg\min_m \|\mathbf{Gm} - \mathbf{d}\|_1^1 . \tag{5.4}$$

The solution entails finding $\mathbf{m}$ that minimizes the sum of the absolute values of the error (the L_1 norm of the error. A simple procedure is to use the Iterative Reweighed Least-Squares (IRLS) algorithm to minimize equation 5.4. In other words, we minimize

$$\frac{\partial}{\partial \mathbf{m}} \|\mathbf{Gm} - \mathbf{d}\|_1^1 = \mathbf{G}^T\mathbf{WGm} - \mathbf{G}^T\mathbf{Wd} \tag{5.5}$$

$$= 0 \tag{5.6}$$

where $\mathbf{W}$ is a diagonal matrix with elements given by

$$W_{ii} = \frac{1}{|e_i|} \tag{5.7}$$

where e_i is the i-th element of the error vector $\mathbf{e} = \mathbf{Gm} - \mathbf{d}$. The resulting system of equations is non-linear because the solution is part of the matrix of weights $\mathbf{W}$. In general, to avoid division by zero, we can rewrite

$$W_{ii} = \frac{1}{|e_i| + \epsilon} \tag{5.8}$$

The IRLS method iteratively solves the problem by a simple algorithm:

$$\mathbf{m}^{[\nu]} = (\mathbf{G}^T\mathbf{W}^{[\nu-1]}\mathbf{G})^{-1}\mathbf{G}^T\mathbf{W}^{[\nu-1]}\mathbf{d}\,. \tag{5.9}$$

Where $^{[\nu]}$ indicates iteration. The algorithm MATLAB script is below.

L_1 Regression via IRLS

```
m=cm0;
Iter_Max = 10;
epsilonc=1.e-4;
for k = 1:Iter_Max;
 dp = G*m;
 e = d - dp;
 W = diag( 1./(abs(e)+epsilon) );
 m = (A'*W*A)\A'*W*d;
end
```

You can use other functions to measure errors. So far, we have seen the quadratic cost function $\|\mathbf{e}\|_2^2$ and now the the l_1 norm $\|\mathbf{e}\|_1$. However, other functions, such as the Huber or Cauchy norm could be used. These two are not strictly norms; they are cost or loos functions less sensitive to outliers than the quadratic function.

Chapter 6

The two canonical operators for linear inverse problems

The solution to large-scale linear inverse problems often boils down to iterative methods that require two simple operations

$$\begin{aligned} \text{Matrix vector multiplication:} \quad & \mathbf{y} = \mathbf{A}\mathbf{x} \\ \text{Transpose of Matrix times vector multiplication:} \quad & \hat{\mathbf{x}} = \mathbf{A}^T\mathbf{y} \end{aligned}$$

where $\mathbf{A} \in R^{N\times M}, \mathbf{x} \in R^{M\times 1}, \mathbf{y} \in R^{N\times 1}$ and $\hat{\mathbf{x}} \in R^{M\times 1}$. The two canonical operations are represented via a diagram in Figure 6.1.

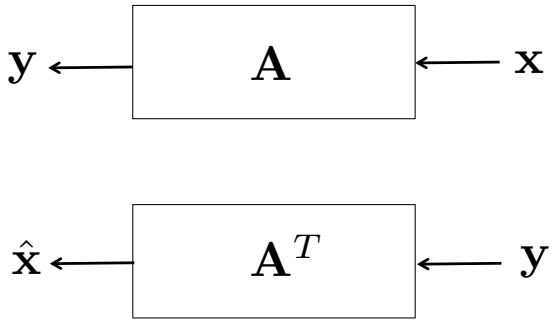


Figure 6.1: Basics operations for solving a linear inverse problem via iterative methods.

A programmer will immediately try to bundle the two operations in one simple function with the

diagram provided in Figure 6.3.

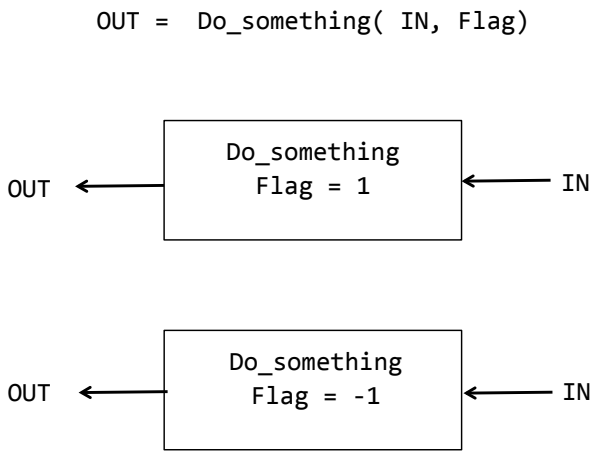


Figure 6.2: Basics operations for solving a linear inverse problem via iterative methods.

$$y_i = \sum_{j=1}^{M} A_{i,j} x_j, \ i = 1, N \, . \tag{6.1}$$

As always, we need a computer code to perform the last sum:

Matrix times Vector Multiplication

```
y= zeros(N,1)        % Allocate a vector of zeros
  for i=1:N
    for j=1:M
      y(i) = y(i) + A(i,j)*x(j)
    end
  end
```

Similarly, the multiplication of the transpose of the matrix times a vector can be evaluated via another sum

$$\hat{x}_j = \sum_{i=1}^{N} A_{i,j} y_i, \ j = 1, M \tag{6.2}$$

As always, we need a computer code to perform the sum:

Transpose Matrix times Vector Multiplication

```
xhat= zeros(M,1)        % Allocate a vector of zeros
  for i=1:N
    for j=1:M
      xhat(j) = xhat(j) + A(i,j)*y(i)
    end
  end
```

As a preparation for more complicated linear operators, I would suggest that you consider the two codes above as one function (See Figure 6.3)

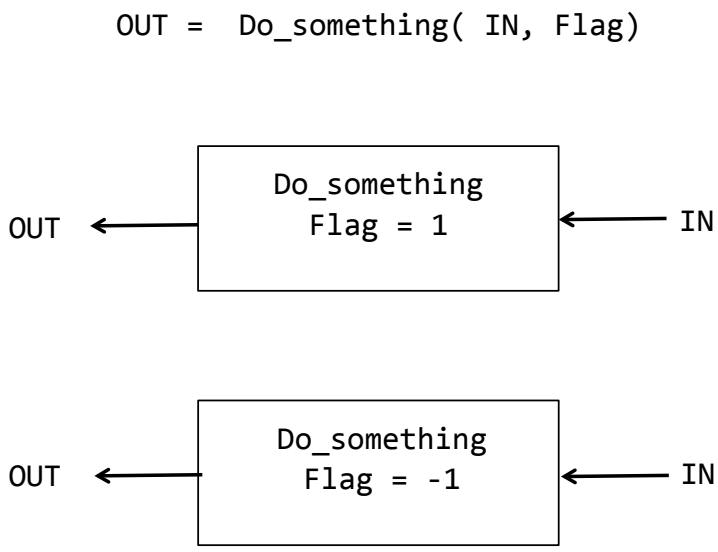


Figure 6.3: Basic operations for solving a linear inverse problem via iterative methods.

Matrix times a vector

```
Function OUT = do_something(IN, A, Flag);

[N,M] = size(A);

if Flag ==  1; OUT = zeros(N,1); end;
if Flag == -1; OUT = zeros(M,1); end;

    for i = 1:N
     for j = 1:M
        if Flag == 1;  OUT(i) = OUT(i) + A(i,j)*IN(j); end;
        if Flag ==-1;  OUT(j) = OUT(j) + A(i,j)*IN(i); end;
      end
    end
```

It is easy to show that the number of operations that are needed to multiply a matrix times a vector is given by

```
Number of sums = M N
Number of products = N M
```

Therefore, matrix-vector multiplication has complexity $\mathcal{O}(MN)$.

Large sparse system of equations

A sparse matrix is a matrix where most elements are equal to zero. The tomographic matrix is a good example of a sparse matrix. The discrete derivative operator we have adopted to estimate smooth solutions is another example of a sparse matrix. Consider the following simple example

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & -1 & 0 \end{pmatrix} . \tag{6.3}$$

The matrix can also be stored in the following form

$$\begin{array}{llll} A_{1,2} \rightarrow & i(1) = 1 & j(1) = 2 & a(1) = 1 \\ A_{1,4} \rightarrow & i(2) = 1 & j(2) = 4 & a(2) = 2 \\ A_{3,3} \rightarrow & i(3) = 2 & j(3) = 3 & a(3) = -1 \end{array} \tag{6.4}$$

Instead of saving $A_{i,j}, i = 1, 2\,, j = 1, 4$, we have saved the non-zero elements of the matrix in the sparse format:

$$i(k), j(k), g(k), k = 1, \quad K$$

where K is the number of non-zero elements of the matrix $\mathbf{A}$. I have not saved much in this example because I used a small matrix for illustrative purposes. However, consider the case of large matrices with most elements equal to zero and then the savings in memory and computing time will become significant. Below is a script showing how one can evaluate matrix-times-vector multiplications when the matrix is sparse.

Sparse Matrix times a vector

```
 Function OUT = do_something_sparse(IN, i, j, a, K, N, M, Flag);

%  Flag =    1: Matrix times vector
%           -1: Transpose of Matrix times vector

 if Flag ==  1; OUT = zeros(N,1); end;
 if Flag == -1; OUT = zeros(M,1); end;

  k=1:K
    if Flag == 1;  OUT(i(k)) = OUT(i(k)) + a(k)*IN(j(k)); end;
    if Flag ==-1;  OUT(j(k)) = OUT(j(k)) + a(k)*IN(i(k)); end;
  end
```

In the sparse matrix-times-vector multiplication case the total number of operations is given by

```
Number of sums = K
Number of products = K
```

The cost of multiplying a sparse matrix times a vector is $K < M \times N$.

Iterative solutions to linear inverse problems

Consider the problem of minimizing the following quadratic cost function $\Phi = ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2$. We consider that the problem is over-determined, and the form $\mathbf{A}^T\mathbf{A}$ is a positive-definite matrix[1]. The solution that minimizes Φ is found by the solution of the following linear system of equations

$$\mathbf{A}^T\mathbf{A}\,\mathbf{x} = \mathbf{A}^T\mathbf{y} \tag{6.5}$$

[1] Φ is strictly convex with a single minimum.

which is given by the least squares solution

$$\mathbf{x}_{ls} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}\,. \tag{6.6}$$

Consider the case where the matrix $\mathbf{A}$ is large. Moreover, consider the case where you cannot form product $\mathbf{A}^T\mathbf{A}$ and solve the system of equations 6.5 to obtain $\mathbf{x}_{ls}$. In this case, one could consider minimizing Φ via an optimization method, such as the steepest descent method. The idea is to find the minimum cost function Φ by taking steps toward the steepest descent direction. The steepest descent direction at point $\mathbf{x}'$ is given

$$-\mathbf{g}(\mathbf{x}') = -\left.\frac{\partial\Phi}{\partial\mathbf{x}}\right|_{\mathbf{x}'} = -\text{gradient of } \Phi$$

where it is easy to show that

$$\begin{aligned}\mathbf{g}(\mathbf{x}') &= \mathbf{A}^T(\mathbf{A}\mathbf{x}' - \mathbf{y}) \\ &= \mathbf{A}^T\mathbf{r}\end{aligned} \tag{6.7}$$

where $\mathbf{r}$ is the vector of residuals.

Therefore, the cost function Φ can be minimized by taking successive steps in the direction $-\mathbf{g}(\mathbf{x})$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \beta\,\mathbf{g}\left(\mathbf{x}^k\right)$$

where k indicates iteration (or step) and the parameter β is the step length that is computed to ensure that $\Phi(\mathbf{x}^k) < \Phi(\mathbf{x}^{k-1})$.

The steepest descent algorithm.

```
% Make a matrix (A), true x (xt), and data (y)
  A  = randn(8,3);
  xt = ones(3,1);
  y = A*xt;
  beta = 0.1;

 % Initial guess
  x = rand(3,1);
  r = A*x-y;
  J = r'*r;
  tol = 0.001;
  Max = 30;
  k = 1;
 while (J>tol \& k<Max);
   r = A*x-y;
   g = A'*r;
   x = x-beta*g;
   J = r'*r;
   k = k +1;
end;
```

The output of the program in terms of iteration number is given below

k	x(1)	x(2)	x(3)
2	1.0780	1.2201	1.1756
3	0.8824	1.1241	1.0751
4	0.9369	1.1148	1.0827
5	0.9386	1.0936	1.0687
6	0.9492	1.0791	1.0601
7	0.9564	1.0665	1.0516
8	0.9630	1.0562	1.0443
9	0.9685	1.0476	1.0378
10	0.9732	1.0403	1.0323
11	0.9772	1.0342	1.0275
12	0.9806	1.0290	1.0235
13	0.9835	1.0246	1.0200
14	0.9860	1.0209	1.0170
15	0.9881	1.0178	1.0144
16	0.9899	1.0151	1.0123

```
1.        1.        1.        --> True answer
```

Conjugate Gradient (CG) Method

The CG method is more elegant than the steepest descent method when dealing with linear problems. The CG method finds the minimum of $\Phi = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ via a series of steps. We first choose an initial solution $\mathbf{x}_0$ and compute $\mathbf{s}_0 = \mathbf{y} - \mathbf{A}\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}_0)$, and $\mathbf{q}_0 = \mathbf{A}_{\mathbf{p_0}}$. Then

$$\begin{aligned}
\alpha_{k+1} &= \mathbf{r}_k^T\mathbf{r}_k / \mathbf{q}_k^T\mathbf{q} \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_{k+1}\mathbf{p}_k \\
\mathbf{s}_{k+1} &= \mathbf{s}_k - \alpha_{k+1}\mathbf{q}_k \\
\mathbf{r}_{k+1} &= \mathbf{A}^T\mathbf{s}_{k+1} \qquad (*) \\
\beta_{k+1} &= \mathbf{r}_k^T\mathbf{r}_k / \mathbf{r}_k^T\mathbf{r} \\
\mathbf{p}_{k+1} &= \mathbf{r}_{k+1} + \beta_{k+1}\mathbf{p}_k \\
\mathbf{q}_{k+1} &= \mathbf{A}\mathbf{p}_{k+1} \qquad (**)
\end{aligned}$$

where $k = 0, 1, 2, ..$ is the iteration number.

Theoretically, the minimum is found after M iterations (steps). We will generally stop in $M' < M$ iterations and be happy with an approximate solution. It is clear that all the computational burden of minimizing J' using CG is in the lines marked $*$ and $**$. Notice an important feature of the CG method, the operation marked with $*$ and $**$ can be executed in implicit form. For instance, there are situations where we do not have an explicit matrix $\mathbf{A}$, but we do have access to liner operators that mimic the application of $\mathbf{A}$ and $\mathbf{A}^T$ to vectors of size $M \times 1$ and $N \times 1$, respectively. In Chapter 7, we will see an example of the latter.

CG method on augmented systems

In general, one would like to find the minimum of problems given by

$$J = ||\mathbf{G}\mathbf{m} - \mathbf{d}||^2 + \mu||\mathbf{W}\mathbf{m}||^2 . \tag{6.8}$$

Can we recycle the CG method to minimize $\Phi = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$ and adapt it to minimize equation 6.8?. Yes, you can. Consider the system

$$\mathbf{A} = \begin{pmatrix} \mathbf{G} \\ \sqrt{\mu}\mathbf{W} \end{pmatrix} \tag{6.9}$$

$$\mathbf{y} = \begin{pmatrix} \mathbf{d} \\ \mathbf{0}_M \end{pmatrix} \tag{6.10}$$

then, J and Φ have the same minimizer [2]. Therefore, when minimizing the cost J we replace $\mathbf{A}$ and $\mathbf{y}$ by equations (6.9) and (6.10) and use CG to minimize the cost J. The following script is a CG vanilla algorithm [3].

[2] Please, prove it!

[3] Jonathan Richard Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994. [`http://www-2.cs.cmu.edu/~jrs/jrspapers.html`]

testcg.m

```
% Make an Example (y = Ax)
N = 20;
M = 10;
A=randn(N,M);
x = ones(M,1);
y = A*x;

% Try to get back x from y

% Initialization
x =zeros(M,1);
s=y-A*x;
p=A'*s;
r=p;
q=A*p;
old = r'*r;
max_iter = 10
% CG loop
for k=1:max_iter
 alpha = (r'*r)/(q'*q);
 x= x +alpha*p;
 s = s-alpha*q;
 r= A'*s;
 new = r'*r;
 beta = new/old;
 old = new;
 p = r + beta*p;
 q = A*p;
end
```

Chapter 7

Fourier reconstruction as a linear inverse problem

Consider a 1D signal $f(t)$ and it Fourier representation in terms of $F(\omega)$

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int f(t) e^{-i\omega t}\, dt \tag{7.1}$$

The inverse Fourier transform cam represent the signal $f(t)$ as a function of $F(\omega)$ via the following expression

$$f(t) = \frac{1}{\sqrt{2\pi}} \int F(\omega) e^{i\omega t}\, d\omega\,. \tag{7.2}$$

The last equation is also a Fredholm integral equation of the first kind with a solution given by equation 7.1. Consider the more interesting case where the signal is only partially known at time samples t_j $j = 1 \ldots N$. In this case, one can write the following expression

$$f(t_j) = \frac{1}{\sqrt{2\pi}} \int F(\omega) e^{-i\omega t_j}\, d\omega\,. \tag{7.3}$$

The inverse problem, in this case, involves retrieving $F(\omega)$ from the partially observed signal $f(t_j)$. Equation 7.1 is not a solution of 7.3 because to compute $F(\omega)$ via equation 7.3 one needs to know $f(t)$ for all t.

The Fourier reconstruction problem pertains to estimating $F(\omega)$ from incomplete data $f(t)$ and then adopting the estimated function $F(\omega)$ to recover $f(t)$ for all t via expression 7.2. We will abandon Fourier integrals for practical purposes and present the problem using a discrete formulation that uses the Discrete Fourier Transform (DFT).

Discrete Fourier Transform

A time series (or a spatial series) is a collection of consecutive signal samples. We first consider signals where the samples are acquired at a constant rate every Δt seconds. Spatial signals are collected every Δx meter. I shall identify a time series via a column vector $\mathbf{s} = [s_0, s_1, s_2, \ldots s_{M-1}]^T$ where s_j corresponds to the sample j of the signal $s(t)$ acquired at time $t = (j-1)\Delta t$. The discrete signal $\mathbf{s}$ can be expressed in terms of its Fourier coefficients via the following expression

$$s_k = \frac{1}{\sqrt{M}} \sum_{l=0}^{M-1} c_l \, e^{i2\pi lk/M} \, . \tag{7.4}$$

The latter is also called the IDFT (Inverse Discrete Fourier Transform). The coefficients $c_l, l = 0, \ldots M-1$ are the Fourier coefficients required to represent the signal s_k. The last equation can be written in matrix-vector form

$$\mathbf{s} = \mathbf{F}\,\mathbf{c} \tag{7.5}$$

where $\mathbf{c}$ is the vector of Fourier coefficients $\mathbf{c} = [c_0, c_1, c_2, \ldots c_{M-1}]^T$ and $\mathbf{F}$ is the $M \times M$ Fourier matrix with elements given

$$F_{k,l} = \frac{1}{\sqrt{M}} e^{i2\pi lk/M} \, , \quad k, l = 0 \ldots M-1$$

Given the discrete signal $\mathbf{s}$, how do we compute the vector of Fourier coefficients $\mathbf{c}$?. The answer is quite simple. The square matrix $\mathbf{F}$ is orthonormal. In other words, $\mathbf{F}^{-1} = \mathbf{F}^H$ (the inverse is equal to its Hermitian transpose). Therefore,

$$\mathbf{c} = \mathbf{F}^H \, \mathbf{s} \, . \tag{7.6}$$

The latter can be written in element form

$$c_l = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} s_k \, e^{-i2\pi lk/M} \, , \quad l = 0 : M-1 \, . \tag{7.7}$$

To summarize our quick review:

1. A discrete signal can be represented by Fourier coefficients via expression 7.4.
2. The Fourier coefficients are computed using formula 7.7.
3. In addition, we have seen that the operations often called DFT and IDFT are nothing else than matrix-vector multiplication with a special orthonormal matrix $\mathbf{F}$ and its Hermitian transpose, respectively.

The two formulas are written again

IDFT (*Synthetics operator*):

$$s_k = \frac{1}{\sqrt{M}} \sum_{l=0}^{M-1} c_l \, e^{i2\pi lk/M} \longrightarrow \mathbf{s} = \mathbf{F}\mathbf{c}$$

DFT (*Analysis operator*):

$$c_l = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} s_k \, e^{-i2\pi lk/M} \longrightarrow \mathbf{c} = \mathbf{F}^H \mathbf{s} \, .$$

Sampling

Consider an example where you would like to acquire a signal of length $M = 8$, $\mathbf{s} = [s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7]^T$. However, there was a problem in the acquisition system, and you could only record the samples $[s_0, s_1, s_2, s_5, s_7]$ ($N = 5$ observations). In other words, your system could not acquire s_3, s_4 and s_6. This can be written in the following form

$$\underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_5 \\ s_7 \end{pmatrix}}_{\mathbf{d}^{obs}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}} \underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{pmatrix}}_{\mathbf{s}} \tag{7.8}$$

We will call the left-hand side vector $\mathbf{d}^{obs}$ (Observations) to stress these are the data that have been observed. Similarly, we will call $\mathbf{T}$ the sampling matrix. Our sampling problem reduces to solving the following system of equations

$$\mathbf{d}^{obs} = \mathbf{T}\,\mathbf{s} \, . \tag{7.9}$$

The naïve solution ≡ The minimum norm solution

We will try to use tricks we have learned in this course to recover $\mathbf{s}$ from $\mathbf{d}^{obs}$. First, we will propose to estimate the minimum norm solution. In other words, we seek a solution that minimizes the model norm subject to data constraints

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \{ \|\mathbf{s}\|_2^2 \} \quad \text{subject to} \quad \mathbf{Ts} = \mathbf{d}^{obs} \tag{7.10}$$

We have already explored the minimum norm solution with exact data constraints in Chapter 3 of this course. The answer is given by

$$\hat{\mathbf{s}} = \mathbf{T}^T(\mathbf{T}\mathbf{T}^T)^{-1}\mathbf{d}^{obs} \tag{7.11}$$

One could easily prove the following interesting properties of the sampling operator

$$\mathbf{T}\mathbf{T}^T = \mathbf{I}, \qquad \mathbf{T}^T\mathbf{T} \neq \mathbf{I}\,.$$

Therefore, equation 7.11 reduces to

$$\hat{\mathbf{s}} = \mathbf{T}^T\mathbf{d}^{obs}\,. \tag{7.12}$$

We can go back to our initial example given by equation 7.8 and show the final solution

$$\underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ 0 \\ 0 \\ s_5 \\ 0 \\ s_7 \end{pmatrix}}_{\hat{\mathbf{s}}} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}^T} \underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_5 \\ s_7 \end{pmatrix}}_{\mathbf{d}^{obs}} \tag{7.13}$$

The resulting solution should come as no surprise to us. The missing data have been replaced by zeros because we have used the minimum norm constraint.

Second attempt: Inversion of the minimum norm Fourier coefficients that represent the data

We will try again to adopt the minimum norm solution to reconstruct our data. However, rather than inverting directly for the data ($\mathbf{s}$), we propose to invert the Fourier coefficients ($\mathbf{c}$) representing the data. In this case, one can write down

$$\begin{aligned} \mathbf{d}^{obs} &= \mathbf{Ts} \quad \text{Data constraint} \\ \mathbf{s} &= \mathbf{Fc} \quad \text{Fourier representation of the ideal data (Equation 5)} \end{aligned}$$

The last two equations can be concatenated into one single expression as follows

$$\mathbf{d}^{obs} = \mathbf{TFc}\,.$$

the new unknown of our problem is $\mathbf{c}$ (the vector of Fourier coefficients). We will attempt to solve the problem using the minimum norm solution for the unknown $\mathbf{c}$, and then, we propose to utilize the estimated vector of Fourier coefficients to reconstruct the signal. We repeat the minimum norm solution but in terms of the unknown vector of Fourier coefficients $\mathbf{c}$

$$\begin{aligned}\hat{\mathbf{c}} =& \arg\min_{\mathbf{c}}\{ \|\mathbf{c}\|_2^2\} \quad \text{subject to} \quad \mathbf{TFc} = \mathbf{d}^{obs} \\ \hat{\mathbf{s}} =& \mathbf{F}\hat{\mathbf{c}} \quad \text{Data reconstruction via Fourier Synthesis}\end{aligned}$$

The solution is given by

$$\hat{\mathbf{c}} = \mathbf{F}^H\mathbf{T}^T(\mathbf{TFF}^H\mathbf{T}^T)^{-1}\mathbf{d}^{obs}\,. \tag{7.14}$$

However, $\mathbf{FF}^H = \mathbf{I}$ (orthonormality of the DFT matrix), and $\mathbf{TT}^T = \mathbf{I}$. Therefore,

$$\hat{\mathbf{c}} = \mathbf{F}^H\mathbf{T}^T\mathbf{d}^{obs}\,. \tag{7.15}$$

Inserting the last expression in the Fourier synthesis equation

$$\begin{aligned}\hat{\mathbf{s}} =& \mathbf{F}\hat{\mathbf{c}} \\ =& \mathbf{FF}^H\mathbf{T}^T\mathbf{d}^{obs} \\ =& \mathbf{T}^T\mathbf{d}^{obs}\,.\end{aligned} \tag{7.16}$$

The solution equals the minimum norm solution given (equation 7.13). In other words, we have not gained much by inverting Fourier coefficients and using them to reconstruct the signal. We have again obtained the naïve interpolation solution where missing data are replaced by unimaginative zeros.

Reconstruction using sparse Fourier coefficients with exact data constraints

We failed to reconstruct our signal because we have adopted a minimum norm constraint. We will consider an alternative solution by adopting the sparsity assumption. The unknown data are non-sparse because $\mathbf{s}$ is a smooth signal or wavefield. However, a smooth signal can be represented via a sparse superposition of harmonics. In other words, we will assume that the ideal signal $\mathbf{s}$ can be defined via a sparse vector of Fourier coefficients $\mathbf{c}$ (Sacchi and Ulrych, 1996; Sacchi et al., 1998). The problem is restated as follows

$$\begin{aligned}\hat{\mathbf{c}} =& \arg\min_{\mathbf{c}}\{ \|\mathbf{c}\|_1\} \quad \text{subject to} \quad \mathbf{TFc} = \mathbf{d}^{obs} \\ \hat{\mathbf{s}} =& \mathbf{F}\hat{\mathbf{c}} \quad \text{Data reconstruction via Fourier Synthesis}\end{aligned}$$

where now $\|\mathbf{c}\|_1 = \sum_l |c_l|$ is the l_1 norm of the vector of Fourier coefficients[1]. We designate $\mathbf{G} = \mathbf{TF}$. Using the method of Lagrange multipliers, we obtain the minimum l_1 norm solution with exact constraints[2]

$$\hat{\mathbf{c}} = \mathbf{Q}\mathbf{G}^H(\mathbf{G}\mathbf{Q}\mathbf{G}^H)^{-1}\mathbf{d}^{obs} \tag{7.17}$$

where $\mathbf{Q}$ is a diagonal matrix with elements given by $\mathbf{Q}_{i,i} = |c_i|$. In general, we use $\mathbf{Q}_{i,i} = |c_i| + \epsilon$ where ϵ is a small number. The sparse solution of the Fourier coefficients is retrieved via an iterative algorithm often called Iteratively Reweighed Least-Squares (IRLS) (Burrus et al., 1994; Sacchi et al., 1998; Daubechies et al., 2010). If we designate the solution at iteration ν by $\mathbf{c}^{[\nu]}$, the IRLS solver can be written down via the following simple iteration

$$\mathbf{Q}^{[0]} = \mathbf{I} \tag{7.18}$$

$$\begin{aligned}
&\text{do } \nu = 1, \nu_{max} \\
&\qquad \mathbf{c}^{[\nu]} = \mathbf{Q}\mathbf{G}^H(\mathbf{G}\mathbf{Q}^{[\nu-1]}\mathbf{G}^H)^{-1}\mathbf{d}^{obs} \\
&\qquad \mathbf{Q}^{[\nu]} = \text{diag}\left(|\mathbf{c}^{[\nu]}| + \epsilon\right) \\
&\text{end do}
\end{aligned} \tag{7.19}$$

Figure 7.1a shows the complete (true) data before decimation. The incomplete data (decimated data) is shown in Figure 7.1b. Figure 7.1c is the reconstructed data where I have adopted the IRLS algorithm to solve equation 7.17 Figure 7.1d is the reconstruction error. Similarly, Figure 7.2 portrays the absolute value of the Fourier coefficients versus iteration ν. It is clear that the IRLS algorithm has converged to a sparse solution where the signal is represented by only three Fourier complex harmonics.

[1] Note: $\|\mathbf{c}\|_1 = \sum_l |c_l| = \sum_l \sqrt{c_l c_l^*}$

[2] $\mathbf{G}^H = \mathbf{F}^H\mathbf{T}^T$.

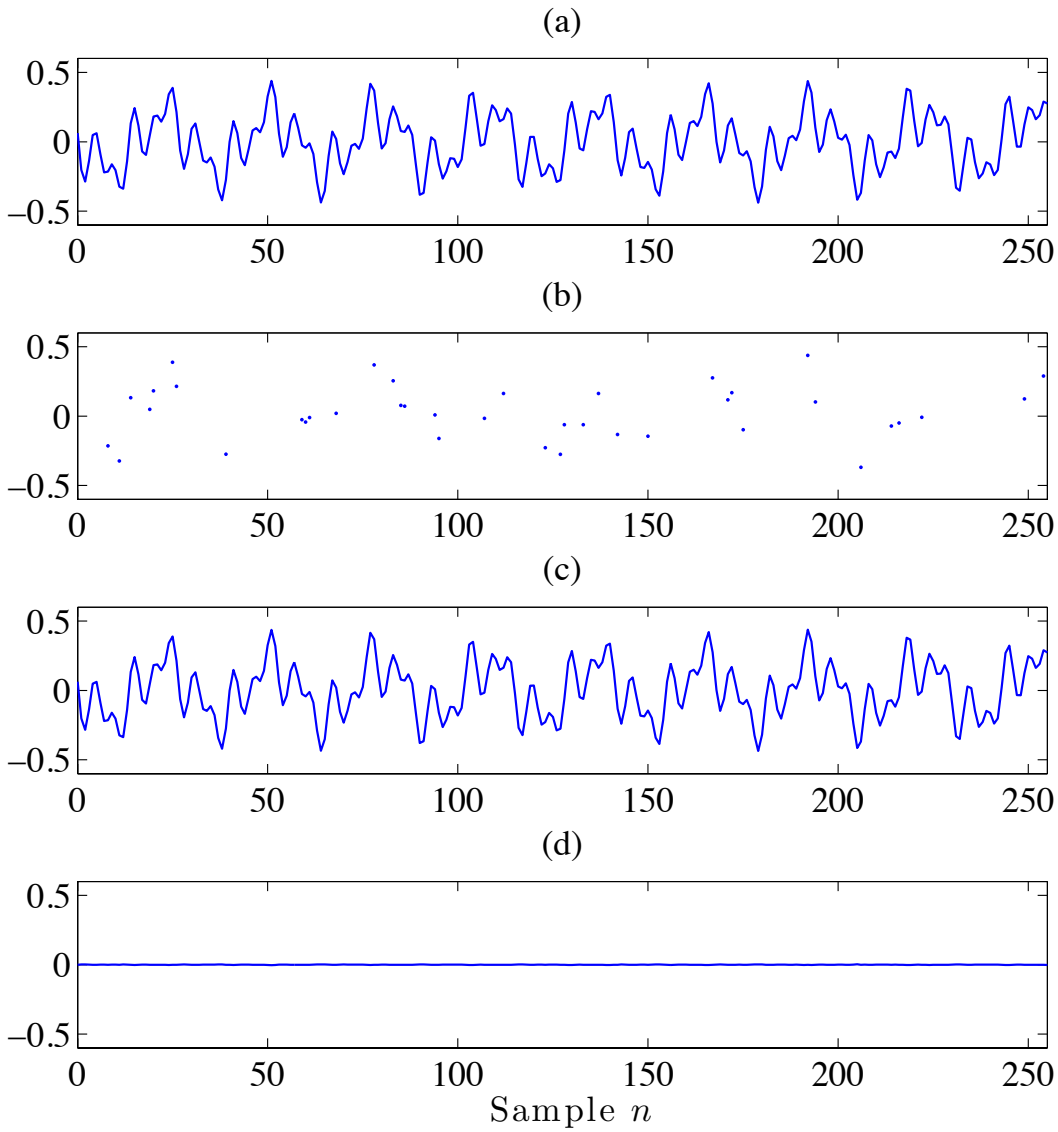


Figure 7.1: a) Ideal signal. b) Signal after sampling. c) Reconstructed signal via sparse inversion of its Fourier coefficients. d) Reconstruction error

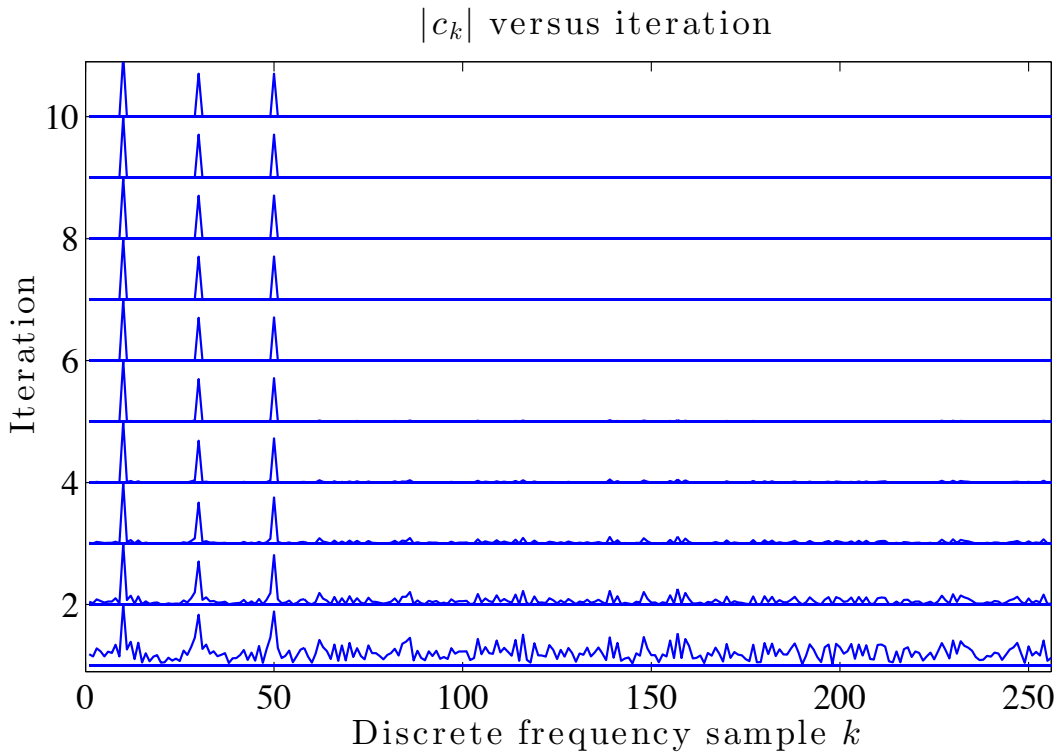


Figure 7.2: The iteratively reweighted least-squares (IRLS) algorithm is used to estimate the sparse Fourier coefficients that honour the observations. This figure shows the absolute value of the Fourier coefficients that are estimated via IRLS versus iteration.

The following simple code was used for preparing Figures 7.1 and 7.2.

```
 close all;
 N  = 256 ;
 F = (1/sqrt(N))*dftmtx(N)';  % Fourier Matrix (DFT)

 c = zeros(N,1);                    % c are DFT coefficients
 c(10,1) = (1+i*3);
 c(30,1) = (-1+i*2);
 c(50,1) = (1+i*2);

 t = [0:1:N-1];
 s = F*c;                               % The signal

% Sampling Operator
 T = eye(N);
 tobs = t;

% Random sampling
for k = N:-1:1
    p = rand(1,1);
       if p>0.15;
         T(k,:)=[];
          tobs(k)=[];
         end
     end

 dobs = T*s;                    % Sampled data

 G = T*F;                         % Operator to invert

% IRLS to estimate c

 c = zeros(N,1);
  for k = 1:10
    q = abs(c)+0.001;
    Q = diag(q);
    u = (G*Q*G')\(dobs);
    c = Q*G'*u;
 end;

dpred = F*c;                    % The reconstructed data
```

$l_1 - l_2$ reconstruction: Noisy observations case

We consider the reconstruction problem in the presence of data that contains noise. In this case, we will estimate the sparse Fourier coefficients via

$$\hat{\mathbf{c}} = \arg\min_{c} \{ \|\mathbf{d}^{obs} - \mathbf{G}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_1 \}, \tag{7.20}$$

where I remind the reader that $\mathbf{G} = \mathbf{TF}$. The parameter λ is the trade-off parameter of the problem. The IRLS algorithm also gives the solution but is written in a form that reminds us of the least-squares solution with damping. Taking the derivatives of the cost function with respect to the unknown Fourier coefficients and, after setting the gradient to zero, yields the following expression

$$(\mathbf{G}^H\mathbf{G} + \lambda\mathbf{P})\mathbf{c} = \mathbf{G}^H\mathbf{d}^{obs} \tag{7.21}$$

where $\mathbf{Q}$ is the diagonal matrix with elements given by $P_{i,i} = (|c_i| + \epsilon)^{-1}$. The IRLS problem generally involves inverting a matrix in each iteration followed by an update of the diagonal matrix $\mathbf{P}$. For instance, one can write the following iteration

$$\begin{aligned} &\text{do } \nu = 1, \nu_{max} \\ &\quad \mathbf{c}^{[\nu]} = (\mathbf{G}^H\mathbf{G} + \lambda\mathbf{P}^{[\nu-1]}))^{-1}\mathbf{G}^H\mathbf{d}^{obs} \\ &\quad \mathbf{P}^{[\nu-1]} = \text{diag}\left(\frac{1}{|\mathbf{c}^{[\nu-1]}| + \epsilon}\right) \\ &\text{end do} \end{aligned} \tag{7.22}$$

The algorithm provided above will definitely work, but it will be quite expensive because one needs to invert a matrix in each iteration. Let us try to understand the two canonical operators $\mathbf{G} = \mathbf{TF}$ and $\mathbf{G}^H = \mathbf{F}^T\mathbf{T}^T$ that should be part of a fast solution. First, consider the DFT matrix and how one can apply it via the `fft`:

Forward/Synthesis: $\mathbf{Fc} \equiv$ `ifft(c)`

Adjoint/Analysis: $\mathbf{F}^H\mathbf{s} \equiv$ `fft(s)`

Now we also incorporate sampling and its transpose operator:

Forward/Synthesis: $\mathbf{TFc} \equiv$ `T * ifft(c)`

Adjoint/Analysis: $\mathbf{F}^H\mathbf{T}^T\mathbf{s} \equiv$ `fft( T'*s)`

M	IRLS with Matrices	IRLS-CGLS with FFT/IFFT
256	0.088	0.0091
512	0.470	0.0195
1024	2.528	0.0493

Table 7.1: Times in seconds for the vanilla IRLS that uses the DFT matrix and the IRLS that calls CGLS with fast matrix-times-vector products that are implemented via FFTs and IFFTs.

Multiplication of a $M \times M$ DFT matrix with a vector takes M^2 operations. On the other hand, evaluating the same multiplication via the `fft` takes $M \log_2 M$ operations. It is clear that an iterative solver for the equation 7.22 will permit us to exploit fast Matrix-times-vector products via the `fft`. The solution in equation 7.22 can be obtaining by CGLS the (Conjugate Gradients Least Squares) method by simply noticing that 7.22 is the solution to the minimum of the cost function

$$J^{[\nu]} = \|\mathbf{G}\mathbf{c}^{[\nu]} - \mathbf{d}^{obs}\|_2^2 + \lambda \|\mathbf{W}^{[\nu-1]}\mathbf{c}\|_2^2 \tag{7.23}$$

where the matrix of diagonal weights is given by $\mathbf{W}^{[\nu-1]} = (\mathbf{P}^{[\nu-1]})^{1/2}$. In essence, you can adopt CGLS to solve for $\mathbf{c}^{[\nu]}$ in 7.22, and all matrix-times-vector products in the CGLS algorithm can be computed via `fft` and `ifft` functions. The latter is denominated matrix-free optimization or optimization with implicit form operators.

One could have adopted methods that are more sophisticated than IRLS. For instance, Beck and Teboulle (2009) proposed a technique called FISTA (Fast iterative shrinkage-thresholding algorithm) which only requires matrix-times-vector and matrix-transpose-time-vector multiplications which could be executed in implicit form.

I have compared the computational times of a reconstruction problem that entails inverting M frequencies via a vanilla IRLS code that uses matrices and for one that uses CGLS with fast multiplications via `fft` and `ifft`. The times are given in seconds and are in the following table:

Bibliography

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

M. Bertero, P. Boccacci, G. Desidera, and G. Vicidomini. Image deblurring with Poisson data: from cells to galaxies. *Inverse Problems*, 25(12):123006, 2009.

C. Burrus, J. Barreto, and I. Selesnick. Iterative reweighted least-squares design of FIR filters. *IEEE Transactions on Signal Processing*, 42(11):2926–2936, 1994. doi: 10.1109/78.330353.

I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010. doi: https://doi.org/10.1002/cpa.20303. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.20303`.

D. Gubbins. *Time Series Analysis and Inverse Theory for Geophysicists*. Cambridge University Press, 2006.

N. Kazemi and M. D. Sacchi. Sparse multichannel blind deconvolution. *Geophysics*, 79(5):V143–V152, 2014.

Y. Li and D. W. Oldenburg. 3D inversion of gravity data. *Geophysics*, 63(1):101–119, 1998.

D. Lizarralde and S. Swift. Smooth inversion of VSP traveltime data. *Geophysics*, 64(3):659–661, 1999.

M. D. Sacchi and T. J. Ulrych. Estimation of the discrete fourier transform, a linear inversion approach. *Geophysics*, 61(4):1128–1136, 1996.

M. D. Sacchi, T. J. Ulrych, and C. J. Walker. Interpolation and extrapolation using a high-resolution discrete fourier transform. *Signal Processing, IEEE Transactions on*, 46(1):31–38, 1998.

T. J. Ulrych and M. D. Sacchi. *Information-Based Inversion and Processing with Applications*. Elsevier Science, 1st edition, 2006.