# *Chapter 3 Regularization Methods for Linear Inverse Problems*

The primary difficulty with linear ill-posed problems is that the inverse image is undetermined due to small (or zero) singular values of $\mathbf{A}$. Actually the situation is a little worse in practice because $\mathbf{A}$ depends on our model of the measurement process and that is typically not precisely known – leading to a slight imprecision in the singular values. Usually that is not significant for the large singular values, but may lead to ambiguity in the small singular values so that we do not know if they are small or zero.

As an introduction to regularization – which is one method for surmounting the problems associated with small singular vectors – we consider a framework for describing the quality of a reconstruction $\hat{\mathbf{f}}$ in an inverse problem.

## 3.1 The Data Misfit and the Solution Semi-Norm

In the last chapter, we considered the linear problem

$$\mathbf{d} = \mathbf{A}\mathbf{f} + \mathbf{n}$$

and focussed on the structure of the operator $\mathbf{A} \in \mathbb{R}^{m\times n}$. As far as the data are concerned, a reconstructed image $\hat{\mathbf{f}}$ is good provided that it gives rise to "mock data" $\mathbf{A}\hat{\mathbf{f}}$ which are close to the observed data. Thus, one of the quantities for measuring the quality of $\hat{\mathbf{f}}$ is the *data misfit* function or the square of the *residual norm*

$$C(\mathbf{f}) = \|\mathbf{d} - \mathbf{A}\mathbf{f}\|^2 . \tag{3.1}$$

However, from our previous considerations, we have seen that choosing $\hat{\mathbf{f}}$ so as to minimize $C(\mathbf{f})$ usually gives a poor reconstruction. If the rank of the operator $\mathbf{A}$ is less than $n$, there are an infinite number of reconstructions, all of which minimize $C(\mathbf{f})$, since the data are not affected by adding to a reconstruction any vector which lies in the null space of $\mathbf{A}$. In the presence of noise, finding the (possibly non-unique) minimum of $C$ is undesirable as it leads to amplification of the noise in the directions of the singular vectors with small singular values. Instead, we usually regard the data as defining a "feasible set" of reconstructions for which $C(\mathbf{f}) \le C_0$ where $C_0$ depends on the level of the noise. Any reconstruction within the feasible set is to be thought of as being consistent with the data.

Since the data do not give us any information about some aspects of $\mathbf{f}$, it is necessary to include additional information which allows us to select from among several feasible reconstuctions. Analytical solutions are available if we choose sufficiently simple criteria. One way of doing this is to introduce a second function $\Omega(\mathbf{f})$ representing our aversion to a particular reconstruction. For example, we may decide that the solution of minimum norm should be chosen from among the feasible set. This can be done by choosing

$$\Omega(\mathbf{f}) = \|\mathbf{f}\|^2 \tag{3.2}$$

Sometimes, we have a preference for reconstructions which are close to some *default solution* $\mathbf{f}^\infty$. This may be appropriate if we have historical information about the quantity. This can be done by choosing

$$\Omega(\mathbf{f}) = \|\mathbf{f} - \mathbf{f}^\infty\|^2 \tag{3.3}$$

More generally, it may not be the norm of $\mathbf{f} - \mathbf{f}^\infty$ which needs to be small, but some linear operator acting on this difference. Introducing the operator $\mathbf{L}$ for this purpose, we can set

$$\Omega(\mathbf{f}) = \|\mathbf{L}(\mathbf{f} - \mathbf{f}^\infty)\|^2 = (\mathbf{f} - \mathbf{f}^\infty)^{\mathrm{t}}\, \mathbf{L}^{\mathrm{t}}\mathbf{L}\,(\mathbf{f} - \mathbf{f}^\infty) \tag{3.4}$$

If the image space is $n$ dimensional and the data space is $m$ dimensional, the matrix $\mathbf{A}$ is of size $m \times n$ and the matrix $\mathbf{L}$ is of size $p \times n$ where $p \leq n$. Typically, $\mathbf{L}$ is the identity matrix or a banded matrix approximation to the $(n-p)$'th derivative. For example, an approximation to the first derivative is given by the matrix

$$\mathbf{L}_1 = \frac{1}{\Delta x}\begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix},$$

while an approximation to the second derivative is

$$\mathbf{L}_2 = \frac{1}{(\Delta x)^2}\begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix}.$$

In other cases, it may be appropriate to minimize some combination of the derivatives such as

$$\Omega(\mathbf{f}) = \alpha_0 \|\mathbf{f} - \mathbf{f}^\infty\|^2 + \sum_{k=1}^{q} \alpha_k \|\mathbf{L}_k(\mathbf{f} - \mathbf{f}^\infty)\|^2,$$

where $\mathbf{L}_k$ is a matrix which approximates the $k$'th derivative, and $\alpha_k$ are non-negative constants. Such a quantity is also the square of a norm, called a *Sobolev norm.*

It is a simple result from linear algebra that any real symmetric positive semidefinite matrix may be factorized into a product of the form $\mathbf{L}^{\mathrm{t}}\mathbf{L}$ where $\mathbf{L}$ is a lower triangular matrix. A constructive proof of this result leads to the so-called *Cholesky factorization* of the square matrix. The Sobolev norm above may thus also be written in the form of (3.4) for a suitable choice of $\mathbf{L}$.

There are many ways of balancing the often conflicting requirements of equations (3.4) and (3.1) and these lead to a variety of regularization methods. We shall discuss two of these below.

## 3.2 Tikhonov regularization

This is perhaps the most common and well-known of regularization schemes. We form a weighted sum of $\Omega(\mathbf{f})$ and $C(\mathbf{f})$ using a weighting factor $\lambda^2$, and find the image $\hat{\mathbf{f}}_\lambda$ which minimizes this sum, i.e.,

$$\hat{\mathbf{f}}_\lambda = \arg\min\left\{\lambda^2 \|\mathbf{L}(\mathbf{f} - \mathbf{f}^\infty)\|^2 + \|\mathbf{d} - \mathbf{A}\mathbf{f}\|^2\right\}. \tag{3.5}$$

This is a whole family of solutions parameterized by the weighting factor $\lambda^2$. We call $\lambda$ the *regularization parameter.* If the regularization parameter is very large, the effect of the data misfit term $C(\mathbf{f})$ is negligible to that of $\Omega(\mathbf{f})$ and we find that $\lim_{\lambda\to\infty} \hat{\mathbf{f}}_\lambda = \mathbf{f}^\infty$. With a large amount of regularization, we effectively ignore the data (and any noise on the data) completely and try to minimize the solution semi-norm which is possible by choosing the default solution. On the other hand, if $\lambda$ is small, the weighting placed on the solution semi-norm is small and the value of the misfit at the solution becomes more important. Of course, if $\lambda$ is reduced to zero, the problem reduces to the least-squares case considered earlier with its extreme sensitivity to noise on the data.

A formal solution to the problem may readily be found. We set

$$\frac{\partial}{\partial f_k}\left\{\lambda^2 (\mathbf{f} - \mathbf{f}^\infty)^{\mathrm{t}} \mathbf{L}^{\mathrm{t}}\mathbf{L}(\mathbf{f} - \mathbf{f}^\infty) + (\mathbf{d} - \mathbf{A}\mathbf{f})^{\mathrm{t}}(\mathbf{d} - \mathbf{A}\mathbf{f})\right\} = 0, \tag{3.6}$$

for $k = 1, 2, ..., n$. This leads to the simultaneous equations

$$2\lambda^2 \mathbf{L}^{\mathrm{t}}\mathbf{L}(\mathbf{f} - \mathbf{f}^\infty) - 2\mathbf{A}^{\mathrm{t}}(\mathbf{d} - \mathbf{A}\mathbf{f}) = 0, \tag{3.7}$$

or

$$\left(\lambda^2\mathbf{L}^{\mathrm{t}}\mathbf{L} + \mathbf{A}^{\mathrm{t}}\mathbf{A}\right)\mathbf{f} = \lambda^2\mathbf{L}^{\mathrm{t}}\mathbf{L}\mathbf{f}^{\infty} + \mathbf{A}^{\mathrm{t}}\mathbf{d}. \tag{3.8}$$

Setting $\lambda = 0$ reduces this system of equations to the normal equations associated with the usual least squares problem. For non-zero values of $\lambda$, the additional term $\lambda^2\mathbf{L}^{\mathrm{t}}\mathbf{L}$ in the matrix on the left-hand side alters the eigenvalues (and eigenvectors) from those of $\mathbf{A}^{\mathrm{t}}\mathbf{A}$ alone. So long that $\left(\lambda^2\mathbf{L}^{\mathrm{t}}\mathbf{L} + \mathbf{A}^{\mathrm{t}}\mathbf{A}\right)$ is non-singular, there is a unique solution. The problem of image reconstruction is thus reduced to solving a (large) system of simultaneous equations with a symmetric positive definite coefficient matrix, and we shall later discuss ways of solving such systems of equations.

## 3.3 Truncated Singular Value Decomposition (TSVD)

Let us suppose that the operator $\mathbf{A}$ has the singular value decomposition

$$\mathbf{A} = \sum_{l=1}^{r} \sigma_l \mathbf{u}_l^{\mathrm{t}} \mathbf{v}_l. \tag{3.9}$$

The truncated singular value decomposition (TSVD) method is based on the observation that for the larger singular values of $\mathbf{A}$, the components of the reconstruction along the corresponding singular vector is well-determined by the data, but the other components are not well-determined. An integer $k \leq n$ is chosen for which the singular values are deemed to be significant and the solution vector $\hat{\mathbf{f}}$ is chosen so that

$$\mathbf{v}_l^{\mathrm{T}}\hat{\mathbf{f}} = \frac{\mathbf{u}_l^{\mathrm{T}}\mathbf{d}}{\sigma_l} \text{ for } l = 1, ..., k. \tag{3.10}$$

The components along the remaining singular-vector directions $\{\mathbf{v}_l\}$ for $l = k+1, ..., n$ are then chosen so that the *total* solution vector $\hat{\mathbf{f}}$ satisfies some criterion of optimality, such as the minimization of a solution semi-norm of the form $\Omega(\mathbf{f}) = \|\mathbf{L}(\mathbf{f} - \mathbf{f}^{\infty})\|^2$ as above. Let us denote by $\mathbf{V}_k$ the $n \times (n-k)$ matrix whose columns are $\{\mathbf{v}_l\}$ for $l = k+1, ..., n$ so that $\mathbf{V}_k$ is the matrix whose columns span the effective null space of $\mathbf{A}$. The reconstruction which has zero projection in this effective null space is

$$\mathbf{f}' = \sum_{l=1}^{k} \left(\frac{\mathbf{u}_l^{\mathrm{T}}\mathbf{d}}{\sigma_l}\right) \mathbf{v}_l \ . \tag{3.11}$$

The desired reconstruction $\hat{\mathbf{f}}$ must be equal to the sum of $\mathbf{f}'$ and a vector which is the superposition of the columns of $\mathbf{V}_k$. This may be written as

$$\hat{\mathbf{f}} = \mathbf{f}' + \sum_{l=k+1}^{n} c_l \mathbf{v}_l = \mathbf{f}' + \mathbf{V}_k \mathbf{c} \tag{3.12}$$

for a $n-k$ element column vector $\mathbf{c}$. The solution semi-norm of this reconstruction is

$$\Omega\left(\hat{\mathbf{f}}\right) = \left\|\mathbf{L}\left(\hat{\mathbf{f}} - \mathbf{f}^{\infty}\right)\right\|^2 = \|\mathbf{L}(\mathbf{f}' + \mathbf{V}_k\mathbf{c} - \mathbf{f}^{\infty})\|^2 = \|\mathbf{L}(\mathbf{f}' - \mathbf{f}^{\infty}) + (\mathbf{L}\mathbf{V}_k)\mathbf{c}\|^2$$

The vector $\mathbf{c}$ which minimizes this semi-norm is

$$\mathbf{c} = -(\mathbf{L}\mathbf{V}_k)^{\dagger}\mathbf{L}(\mathbf{f}' - \mathbf{f}^{\infty})$$

where the dagger represents the Moore-Penrose inverse. i.e., for any matrix $\mathbf{A}$, we define $\mathbf{A}^{\dagger} = (\mathbf{A}^{\mathrm{t}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{t}}$. This gives an explicit expression for the truncated singular value decomposition solution, namely

$$\hat{\mathbf{f}} = \mathbf{f}' - \mathbf{V}_k(\mathbf{L}\mathbf{V}_k)^{\dagger}\mathbf{L}(\mathbf{f}' - \mathbf{f}^{\infty}) \tag{3.13a}$$

where $\mathbf{f}'$ is given by (3.11) above.

Note that some authors use the terminology *truncated singular value decomposition* to refer to the special case where $\mathbf{L}$ is chosen to be the identity matrix, and call the general case derived above the *modified truncated singular value decomposition.*

## 3.4 Filter factors

In any regularization scheme, there is a *regularization parameter* which is a quantity that can be adjusted in order to change the degree of regularization of the solution. For values of this parameter at one end of its range, the solution is usually smoother, more similar to the default solution and less affected by noise on the data whereas for values of this parameter at the other end, the solution can be very sensitive to noise as it is primarily determined by the requirement of minimizing the data residual. In the case of Tikhonov regularization, the parameter is the quantity $\lambda$ while in the case of the TSVD method, it is the choice of $k$ at which the singular values are deemed to be negligible.

It is useful to be able to look at the range of solutions which result as the regularization parameter is varied. This can always be done by recomputing the solution from scratch for each value of the parameter, but this is computationally very intensive as we often need to invert a large matrix for each choice of the regularization parameter. An advantage of studying the singular value decomposition is that it provides a convenient way of investigating the family of regularized solutions without having to reinvert large matrices. We shall thus re-examine the regularization methods described above in terms of the singular value decomposition.

Tikhonov regularization can be analysed in this way when the matrix $\mathbf{L}$ happens to be the identity. For more general $\mathbf{L}$, an extension of the singular value decomposition called the generalized singular value decomposition (GSVD) may be defined, but we shall not be considering it in this course. The solution to the problem is given by

$$\left(\lambda^2\mathbf{I}+\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)\hat{\mathbf{f}}=\lambda^2\mathbf{f}^\infty+\mathbf{A}^{\mathrm{t}}\mathbf{d} \tag{3.14}$$

Let us suppose that we have computed the singular value decomposition of $\mathbf{A}$ in the usual form

$$\mathbf{A}=\sum_{l=1}^{r}\sigma_l\mathbf{u}_l\mathbf{v}_l^{\mathrm{t}} \tag{3.15}$$

then

$$\left(\lambda^2\mathbf{I}+\mathbf{A}^{\mathrm{t}}\mathbf{A}\right)\hat{\mathbf{f}}=\lambda^2\sum_{l=1}^{n}\hat{f}_l\mathbf{v}_l+\sum_{l=1}^{r}\sigma_l^2\hat{f}_l\mathbf{v}_l \tag{3.16}$$

$$=\sum_{l=1}^{r}\left(\lambda^2+\sigma_l^2\right)\hat{f}_l\mathbf{v}_l+\lambda^2\sum_{l=r+1}^{n}\hat{f}_l\mathbf{v}_l \tag{3.17}$$

where $\hat{f}_l=\mathbf{v}_l^{\mathrm{t}}\hat{\mathbf{f}}$ and

$$\lambda^2\hat{\mathbf{f}}^\infty+\mathbf{A}^{\mathrm{t}}\mathbf{d}=\lambda^2\sum_{l=1}^{n}f_l^\infty\mathbf{v}_l+\sum_{l=1}^{r}\sigma_l d_l\mathbf{v}_l \tag{3.18}$$

$$=\sum_{l=1}^{r}\left[\lambda^2 f_l^\infty+\sigma_l^2\left(\frac{d_l}{\sigma_l}\right)\right]\mathbf{v}_l+\lambda^2\sum_{l=r+1}^{n}f_l^\infty\mathbf{v}_l \tag{3.19}$$

where $f_l^\infty=\mathbf{v}_l^{\mathrm{t}}\mathbf{f}^\infty$, $d_l=\mathbf{u}_l^{\mathrm{t}}\mathbf{d}$ and we have made use of the fact that

$$\mathbf{I}=\sum_{l=1}^{n}\mathbf{v}_l\mathbf{v}_l^{\mathrm{t}}$$

since the $\{\mathbf{v}_l\}_{l=1}^n$ form an orthonormal basis of $\mathbb{R}^n$. Equating (3.17) and (3.19) and using the linear independence of the vectors $\mathbf{v}_l$ we see that

$$\hat{f}_l=\begin{cases}\frac{\lambda^2}{\lambda^2+\sigma_l^2}f_l^\infty+\frac{\sigma_l^2}{\lambda^2+\sigma_l^2}\left(\frac{d_l}{\sigma_l}\right) & \text{for } l=1,2,...,r,\\ f_l^\infty & \text{for } l=r+1,...,n.\end{cases} \tag{3.20}$$

This displays the solutions to the regularization problem for all values of $\lambda$ in a convenient form. In the directions of the singular vectors $\mathbf{v}_{r+1}, ..., \mathbf{v}_n$ which span the null space of $\mathbf{A}$, the projections $\hat{f}_l$ of the regularized solution are equal to the projections of the default solution $f_l^\infty$. This is not unreasonable as the data do not give us any information about those aspects of the image. On the other hand, along each of the directions $\mathbf{v}_1, ..., \mathbf{v}_r$ for which the data do give us some information, the regularized solution is a weighted linear combination of $f_l^\infty$, which is what the default solution would have us take, and of $d_l/\sigma_l$ which is what the data alone would have suggested. Notice that since the weights add up to one, the regularized solution lies along the line in $n$ space joining these points. The value of the weights is also of interest. As $\lambda$ becomes large, the solution is pulled towards the default solution, as expected, but it should be noticed that where along the line the solution ends up at depends on the relative values of $\lambda$ and of $\sigma_l$. The larger is the singular value $\sigma_l$, the smaller is the relative effect of the regularization parameter $\lambda$. In fact we need to have $\lambda = \sigma_l$ in order to pull the component of the solution to the midpoint of the line joining $f_l^\infty \mathbf{v}_l$ and $(d_l/\sigma_l)\,\mathbf{v}_l$. This is desirable since it is precisely in the directions corresponding to the large singular values that the data give us the greatest information.

The quantities $\sigma_l^2/\left(\lambda^2 + \sigma_l^2\right)$ which multiply the data coefficient $(d_l/\sigma_l)$ are called *filter factors*. They show how the algorithm reduces the weighting for the data which are associated with the small singular values. Depending on the level of the noise on the data, we need different amounts of protection against the noise-amplifying effects of reconstruction using the small singular values.

By contrast to the Tikhonov regularization algorithm in which the filter factors smoothly decrease to zero as the singular values gets smaller, the truncated singular value algorithms have filter factors which are equal to unity for those singular values which are deemed to be non-negligible $(l \leq k)$ and to zero for those singular values which are negligible $(l > k)$. The value of the components of the regularized solution in the directions of the significant singular values are completely determined by the data, as

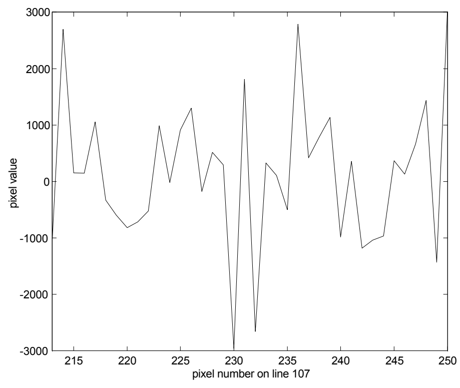
$$\hat{f}_l = \frac{d_l}{\sigma_l} \text{ for } l = 1, 2, ..., k$$

The other components $\hat{f}_{k+1}, ..., \hat{f}_n$ are adjusted so as to minimize the solution semi-norm.

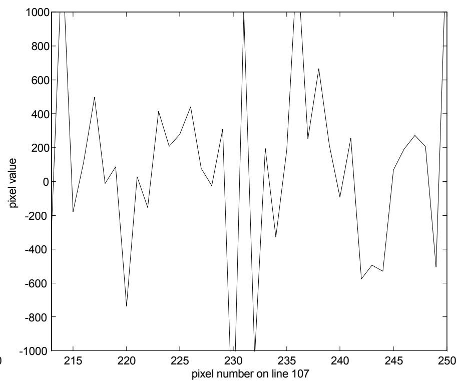
## 3.5 Smoothness versus data-fitting in the deblurring example

The regularizing parameter, $\lambda$, can be thought of as controlling the balance between minimizing the regularizing term – which is often a criterion of smoothness of the reconstructed image – and minimizing the term which corresponds to fitting the data. When $\lambda$ is small, there is little weight put on the regularizing term, the data is fitted well and the reconstructed image is not smooth. Conversely when $\lambda$ is large, the regularizer dominates the minimization and the reconstructed image is smooth – at the expense of not fitting the data so well.

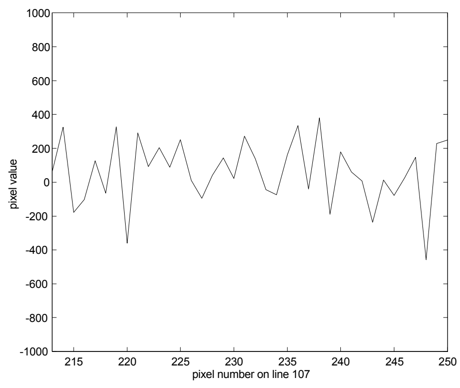
As an example, I have plotted some pixel values from the deblurring example in Chapter 1. The pixels shown are those with indices `(107,210:250)` – to use the Matlab notation – and are the pixels through the middle of the word "way".



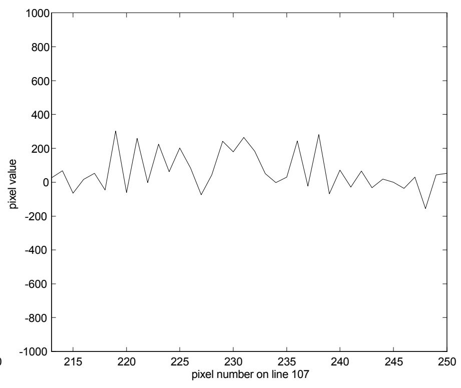
Pixel values in unregularized inverse.



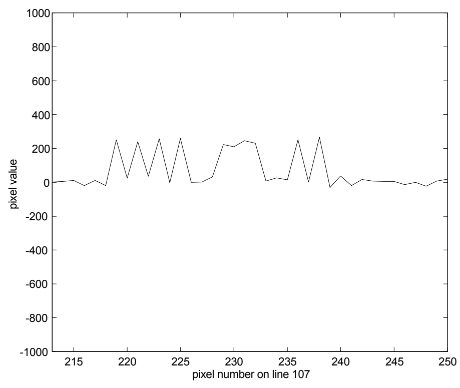
Regularized inverse: $\lambda = 0.1$



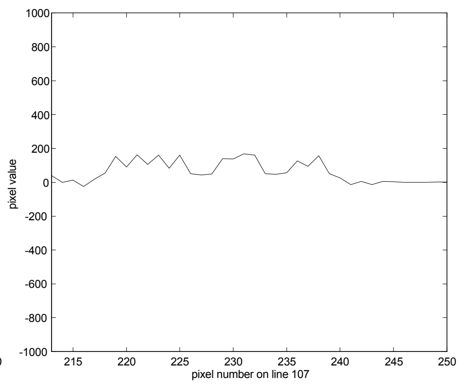
Regularized inverse: $\lambda = 1$



Regularized inverse: $\lambda = 10$



Regularized inverse: $\lambda = 100$



Regularized inverse: $\lambda = 1000$

Note how the graphs of pixel value become more smooth as $\lambda$ is increased.

## 3.6 Choosing the regularization parameter

We have seen that $\lambda$ sets the balance between minimizing the residual norm $\|\mathbf{d} - \mathbf{A}\mathbf{f}_\lambda\|_2$and minimizing the roughness $\|\mathbf{f}_\lambda - \mathbf{f}^\infty\|_2$ .The big question now is "how to choose $\lambda$"?

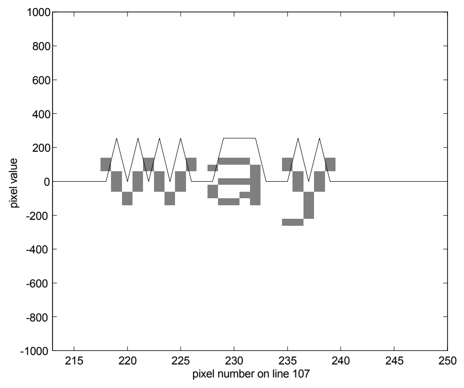


Figure 3.1 Pixel values in original (unblurred and un-noisy) image with the text, as reference.

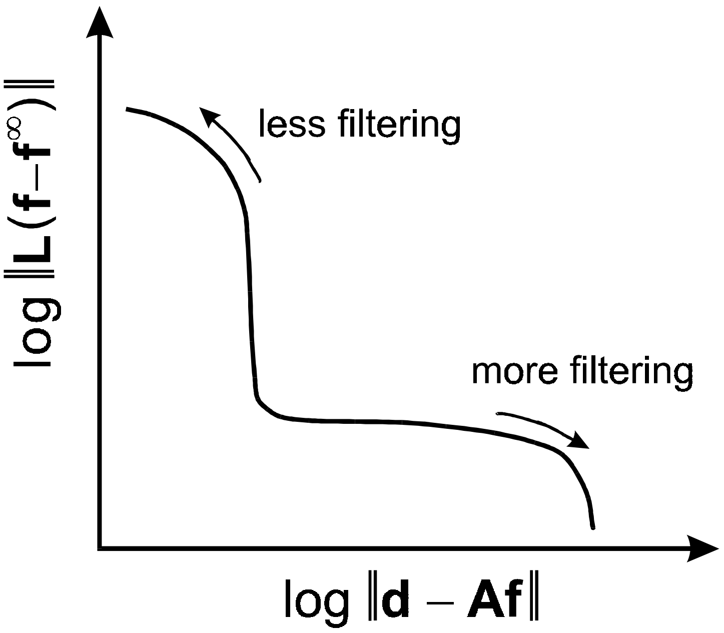

$\log \|\mathbf{L}(\mathbf{f}-\mathbf{f}^\infty)\|$
less filtering
more filtering
$\log \|\mathbf{d}-\mathbf{Af}\|$

Figure 3.2 The generic form of the L-curve.

### 3.6.1 The L-Curve

Perhaps the most convenient graphical tool for setting $\lambda$ is the "L-curve". When we plot $\log \|\mathbf{d}-\mathbf{A}\mathbf{f}_\lambda\|$ versus $\log \|\mathbf{f}_\lambda - \mathbf{f}^\infty\|$ (for a discrete problem) we get the characteristic L-shaped curve with a (often) distinct corner separating vertical and horizontal parts of the curve.

The rationale for using the L curve is that regularization is a trade-off between the data misfit and the solution seminorm. In the vertical part of the curve the solution seminorm $\|\mathbf{L}(\mathbf{f}-\mathbf{f}^\infty)\|$ is a very sensitive function of the regularization parameter because the solution is undergoing large changes with $\lambda$ in an attempt to fit the data better. At these low levels of filtering, there are still components in the solution which come from dividing by a small singular value, which corresponds to having inadequate regularization. On the horizontal part, the solution is not changing by very much (as measured by the solution seminorm) as $\lambda$ is changed. However, the data misfit is increasing sharply with more filtering. Along this portion, our preference for the more highly filtered solutions increases only slightly at the cost of a rapidly rising data misfit, and so it is desirable to choose a solution which lies not too far to the right of the corner.

The following figure shows the L-curve that is associated with the deblurring example in chapter 1.

The curve is labelled parametrically with the values of the regularization parameter. In this example, the bend in the curve is not very sharp, and the solution which appears optimal visually lies slightly to the right

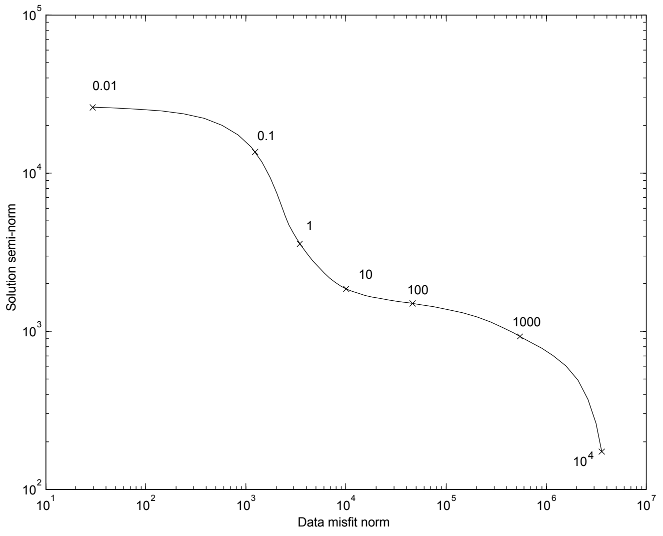


Figure 3.3 L-curve for the deblurring example

of the position of largest upwards-pointing curvature.

### 3.6.2 The Discrepency Principle

This is an alternative criterion advocated by those who believe that "the data come first". The value of $\lambda$ is chosen so as to place the solution on the edge of the feasible set as defined by $C(\mathbf{f}) \le C_0$. Since $C(\mathbf{f})$ is the sum of squares of noise terms, its expectation value can be calculated if one knows the amount of noise that is present. The value of $C_0$ is selected such that the probability of exceeding this value due to chance alone is smaller than some threshold, say one percent. A problem with the practical use of this method is that there are often systematic errors as well (such as an inaccurate characterization of the forward problem) in the observation process, so that attempting to fit the data to within the precision allowed by the random errors alone can sometimes lead to under-regularization. This corresponds to finding a solution on the vertical part of the L curve.

## 3.7 Solving Large Systems of Simultaneous Equations for Regularization Problems

This is a very large subject in its own right which we cannot treat in detail. Although the singular value decomposition (and the generalized SVD) is useful for the theoretical study of inverse problems, it is rather numerically intensive to calculate in practice. As a rough guide, finding the SVD is feasible when there are up to a few hundred variables, and so is of limited use for problems involving several thousand to several million components in the image and data vectors.

The mapping from the image to the data is specified by multiplication by the matrix $\mathbf{A}$. For large problems, it is often not feasible to store the matrix $\mathbf{A}$ in the computer, or to compute its action by direct multiplication. If for example, we take the problem of blurring an image consisting of a $256\times256$ pixel scene to produce a data photograph of the same size, the sizes of image and data space are each 65536 dimensional, and the matrix $\mathbf{A}$ has $256^4$ elements. In the case of blurring, we know that the two-dimensional convolution corresponding to the action of $\mathbf{A}$ can be computed efficiently via the trick of multiplying together Fourier transforms, and so the matrix $\mathbf{A}$ is never formed explicitly. When writing algorithms for solving such problems, we cannot use methods which involve manipulating the full matrices, and we should regard even the storage of vectors in image and data space being rather costly. For generality, we assume that the user will provide us with a function that will calculate $\mathbf{Av}$ and $\mathbf{Lv}$ when provided with a vector $\mathbf{v} \in \mathbb{R}^n$. As we shall see, we also need to be provided with routines to calculate $\mathbf{A}^t\mathbf{u}$ for $\mathbf{u} \in \mathbf{R}^m$ and $\mathbf{L}^t\mathbf{w}$ for $\mathbf{w} \in \mathbb{R}^p$. These functions will usually employ some trick such as using Fourier transforms or exploiting the sparsity of the matrices in order to do the calculation in a reasonable time.

In the above, we derived the simultaneous equations for Tikhonov regularization from the minimization of the function

$$\mathcal{L}(\mathbf{f}) = \lambda^2 \left\|\mathbf{L}(\mathbf{f} - \mathbf{f}^\infty)\right\|^2 + \left\|\mathbf{d} - \mathbf{Af}\right\|^2$$

For the linear inverse problem, this is a *quadratic* in the image $\mathbf{f}$. Before describing the algorithm, we need to consider some properties of such quadratic expressions.

### 3.7.1 Minimizing a quadratic in many dimensions

A quadratic expression in a vector $\mathbf{x} \in \mathbb{R}^n$ has the general form

$$Q(\mathbf{x}) = \mathbf{x}^t\mathbf{Hx} - \mathbf{x}^t\mathbf{G} - \mathbf{G}^t\mathbf{x} + Q_0 \tag{3.21}$$

where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a real symmetric matrix, $\mathbf{G} \in \mathbb{R}^n$ is a vector and $Q_0 \in \mathbb{R}$. This quadratic has a stationary point whenever

$$\frac{\partial Q}{\partial x_i} = 2\sum_j h_{ij}x_j - 2g_i = 0 \tag{3.22}$$

for all $i \in \{1, ..., n\}$. i.e., stationary points $\mathbf{x}_s$ of $Q$ satisfy the set of linear equations

$$\mathbf{Hx}_s = \mathbf{G} \tag{3.23}$$

Depending on the nature of the matrix $\mathbf{H}$, this can have none, one or infinitely many solutions. A special case of interest is when $\mathbf{H}$ is *positive definite,* which means that $\mathbf{x}^t\mathbf{Hx} \geq 0$ for all $\mathbf{x}$, and that $\mathbf{x}^t\mathbf{Hx} = 0$ only if $\mathbf{x} = \mathbf{0}$. If $\mathbf{H}$ is positive definite, it is invertible and there is a unique stationary point

$$\mathbf{x}_s = \mathbf{H}^{-1}\mathbf{G} \tag{3.24}$$

The original quadratic may be written as

$$Q(\mathbf{x}) = (\mathbf{x} - \mathbf{x}_s)^t \mathbf{H}(\mathbf{x} - \mathbf{x}_s) + \left(Q_0 - \mathbf{x}_s^t\mathbf{Hx}_s\right) \tag{3.25}$$

from which it is clear that $\mathbf{x}_s$ is the global minimum of $Q$. When $\mathbf{H}$ is positive definite, we may regard solving the system of equations (3.23) and minimizing the quadratic (3.21) as being equivalent problems.

Let us now consider the problem of minimizing $Q(\mathbf{x})$ when $n$ is so large that computing $\mathbf{H}^{-1}$ explicitly is not feasible. We can consider an iterative algorithm which proceeds from an initial guess $\mathbf{x}_0$ of the minimum to a point $\mathbf{x}_1$ which is hopefully a better estimate of $\mathbf{x}_s$. One way of doing this is to consider the direction of *steepest descent* from $\mathbf{x}_0$. This is along the direction opposite to the gradient of $Q$ at $\mathbf{x}_0$. We see that

$$\nabla Q = 2(\mathbf{Hx} - \mathbf{G})$$

and so the direction of steepest descent is along $-\nabla Q$ which is parallel to $\mathbf{s}_1 = \mathbf{G} - \mathbf{Hx}_0$. We now proceed along the line $\mathbf{x}_0 + c_1\mathbf{s}_1$ trying to find a better approximation to $\mathbf{x}_s$. It is easy to find out how $Q$ behaves on this line, since

$$\begin{aligned} Q(\mathbf{x}_0 + c_1\mathbf{s}_1) &= (\mathbf{x}_0 + c_1\mathbf{s}_1)^t \mathbf{H}(\mathbf{x}_0 + c_1\mathbf{s}_1) - (\mathbf{x}_0 + c_1\mathbf{s}_1)^t \mathbf{G} - \mathbf{G}^t(\mathbf{x}_0 + c_1\mathbf{s}_1) + Q_0 \\ &= \left(\mathbf{s}_1^t\mathbf{Hs}_1\right)c_1^2 - \left\{\mathbf{s}_1^t(\mathbf{G} - \mathbf{Hx}_0) + (\mathbf{G} - \mathbf{Hx}_0)^t\mathbf{s}_1\right\}c_1 + Q(\mathbf{x}_0) \end{aligned} \tag{3.26}$$

This is a quadratic in $c_1$ whose minimum is readily found to be at

$$c_1 = \frac{\mathbf{s}_1^t(\mathbf{G} - \mathbf{Hx}_0) + (\mathbf{G} - \mathbf{Hx}_0)^t\mathbf{s}_1}{2\left(\mathbf{s}_1^t\mathbf{Hs}_1\right)} = \frac{\mathbf{s}_1^t\mathbf{s}_1}{\mathbf{s}_1^t\mathbf{Hs}_1} \tag{3.27}$$

We now set $\mathbf{x}_1 = \mathbf{x}_0 + c_1\mathbf{s}_1$ as our next estimate of the position of $\mathbf{x}_s$. Notice that in order to compute $c_1$, all that we need to be able to do is to calculate $\mathbf{Hs}_1$ and to carry out arithmetic with *vector* quantities. So long that $\mathbf{Hs}_1$ can be computed efficiently, no large matrices need to be stored. We can proceed iteratively, finding $-\nabla Q(\mathbf{x}_1)$ and searching along this direction from $\mathbf{x}_1$ to find the point $\mathbf{x}_2$ which minimizes $Q$ along this line. This is known as the *steepest descent algorithm* for minimizing a function. At each iteration, a one dimensional search is carried out, and the hope is that a succession of these will ultimately lead to the minimization of the $n$ dimensional quadratic. Despite its intuitive appeal, it is a very inefficient way of minimizing a function of many variables. Unless the contours of $Q$ are spherical, it requires many more than $n$ iterations to find the minimum due to a phenomenon called "hem-stitching" in which the succession of iterates slowly crawls down the walls of a long elliptical valley. In effect, each successive search largely undoes the work of the previous step, and the result is only a very gradual reduction in $Q$. For large $n$, this algorithm is essentially useless.

Instead of starting at an initial guess and searching along a single line for the minimum of $Q$, we can consider searching in a larger *subspace* starting from an initial guess. For the moment, suppose that someone gives us a list of linearly independent *search directions,* $\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_k$ and asks us to minimize $Q$ within the subspace

$$\mathbf{x}_0 + c_1\mathbf{s}_1 + ... + c_n\mathbf{s}_n = \mathbf{x}_0 + \mathbf{Sc} \tag{3.28}$$

where $\mathbf{S}$ is the matrix whose columns are the search directions and $\mathbf{c}$ is the column vector of the coefficients. Within this $k$ dimensional affine space, we see that

$$\begin{aligned} Q(\mathbf{x}_0 + \mathbf{Sc}) &= (\mathbf{x}_0 + \mathbf{Sc})^{\mathrm{t}} \mathbf{H} (\mathbf{x}_0 + \mathbf{Sc}) - (\mathbf{x}_0 + \mathbf{Sc})^{\mathrm{t}} \mathbf{G} - \mathbf{G}^{\mathrm{t}} (\mathbf{x}_0 + \mathbf{Sc}) + Q_0 && (3.29)\\ &= \mathbf{c}^{\mathrm{t}} \left(\mathbf{S}^{\mathrm{t}}\mathbf{HS}\right) \mathbf{c} - \mathbf{c}^{\mathrm{t}}\mathbf{S}^{\mathrm{t}} (\mathbf{G} - \mathbf{Hx}_0) - (\mathbf{G} - \mathbf{Hx}_0)^{\mathrm{t}} \mathbf{Sc} + \left(Q_0 + \mathbf{x}_0^{\mathrm{t}}\mathbf{Hx}_0 - \mathbf{x}_0^{\mathrm{t}}\mathbf{G} - \mathbf{G}^{\mathrm{t}}\mathbf{x}_0\right) && (3.30)\\ &= \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{H}}\mathbf{c} - \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{G}} - \tilde{\mathbf{G}}^{\mathrm{t}}\mathbf{c} + \tilde{Q}_0 && (3.31) \end{aligned}$$

which is also a quadratic in $\mathbf{c}$, with the matrices

$$\begin{aligned} \tilde{\mathbf{H}} &= \mathbf{S}^{\mathrm{t}}\mathbf{HS} && (3.32)\\ \tilde{\mathbf{G}} &= \mathbf{S}^{\mathrm{t}} (\mathbf{G} - \mathbf{Hx}_0) && (3.33)\\ \tilde{Q}_0 &= Q(\mathbf{x}_0) = Q_0 + \mathbf{x}_0^{\mathrm{t}}\mathbf{Hx}_0 - \mathbf{x}_0^{\mathrm{t}}\mathbf{G} - \mathbf{G}^{\mathrm{t}}\mathbf{x}_0 && (3.34) \end{aligned}$$

Since the columns of $\mathbf{S}$ are linearly independent and the matrix $\mathbf{H}$ is positive definite, so is the matrix $\tilde{\mathbf{H}}$. By the above discussion, the minimum of $Q(\mathbf{x}_0 + \mathbf{Sc})$ in this affine subspace is located at

$$\hat{\mathbf{c}} = \tilde{\mathbf{H}}^{-1}\tilde{\mathbf{G}} \tag{3.35}$$

This can be readily computed since the matrix $\tilde{\mathbf{H}}$ is only of size $k \times k$ where $k$ is the number of search directions in the subspace. When using a subspace search, the next guess at the minimizing point $\mathbf{x}_s$ is

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{S}\hat{\mathbf{c}} = \mathbf{x}_0 + \mathbf{S}\tilde{\mathbf{H}}^{-1}\tilde{\mathbf{G}} \tag{3.36}$$

We can then proceed iteratively to get a succession of estimates to $\mathbf{x}_s$.

The efficiency of the resulting algorithm depends critically on making an appropriate choice for the search directions. One might imagine starting at $\mathbf{x}_0$ and searching in the direction of $\mathbf{s}_1 = -\nabla Q(\mathbf{x}_0)$, thus reaching $\mathbf{x}_1$. From there, we find $\mathbf{s}_2 = -\nabla Q(\mathbf{x}_1)$, but instead of simply searching along this line, we can search in the affine subspace spanned by $\mathbf{s}_1$ and $\mathbf{s}_2$ starting at $\mathbf{x}_1$. Having found the minimum of $Q$ in this subspace at $\mathbf{x}_2$, we then set $\mathbf{s}_3 = -\nabla Q(\mathbf{x}_2)$ and search the subspace spanned by $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$. In this way, we search for the minimum over larger and larger spaces, which ensures that each search does not undo the work of the previous searches. The spaces generated in this manner are known as the *Krylov subspaces.* The algorithm, as described would rapidly become unworkable since the search space increases in size on each iteration. However, due to a truly amazing result which we do not have time to prove, it turns out that for quadratics, it is possible to achieve the same result as searching over all the space spanned by all the previous search directions by searching firstly along $-\nabla Q(\mathbf{x}_0)$ and subsequently over only a two-dimensional space on each iteration.

Suppose we have reached the point $\mathbf{x}_l$ after the $l$'th iteration. We calculate $-\nabla Q(\mathbf{x}_l)$ as before and search within the affine space spanned by this vector and the vector $\mathbf{x}_l - \mathbf{x}_{l-1}$. It can be shown that in the absence of round-off errors, the effect is the same as if we had searched in the space spanned by $-\nabla Q(\mathbf{x}_0)$, $-\nabla Q(\mathbf{x}_1), ..., -\nabla Q(\mathbf{x}_l)$. With this algorithm and perfect arithmetic, one can reach the minimum of an $n$ dimensional quadratic in at most $n$ steps.

The following Matlab code illustrates how a search can be carried out over an arbitrary affine subspace around the current point. The user specifies the search directions as the columns of the matrix $\mathbf{S}$.

```
function [xnew,Qpred] = search1(x0,res,Hfunc,Qnow,S)
% Searches in an affine subspace for the minimum of the
%  quadratic
%   Q(x) = x'*H*x-x'*G-G'*x+Q0
% x0    = Initial guess of location of the minimum
% res   = G-H*x0
% Hfunc = Name of function which calculates H*x for given x
% Qnow  = Value of Q(x0)
% S     = matrix with search directions as its columns

% Sze Tan, University of Auckland, July 1998

nsrch = size(S,2);
HS = zeros(length(x0),nsrch);
fprintf('Value of quadratic (current)   = %f\n',Qnow);
for k = 1 : nsrch
   HS(:,k) = feval(Hfunc,S(:,k));
end
Hq = S'*HS;
Gq = S'*res;
c  = Hq\Gq;
Qpred = Qnow + c'*Hq*c - c'*Gq - Gq'*c;
fprintf('Value of quadratic (predicted) = %f\n',Qpred);
xnew  = x0 + S * c;
```

In this code, the function whose name is in `Hfunc` is used to apply the matrix $\mathbf{H}$ to an arbitrary vector. In general, this will compute the product in some indirect way for efficiency. However, as an illustrative example, the following shows how we can solve the problem $\mathbf{Hx} = \mathbf{G}$ for a small matrix $\mathbf{H}$ by making the function `Hfunc` do an explicit matrix multiplication. Note that $\mathbf{H}$ has to be positive definite for the algorithm to work. In this algorithm, we predict the value that $Q(\mathbf{x})$ is going to have on the next iteration by examining its behaviour in the subspace. On the next iteration the value of $Q$ is recalculated at this point and it is a good check to see that the actual value agrees with the predicted value.

```
global H
% Set up a random positive definite matrix H and a right-hand side
%  of the equation
neq = 20;
H = rand(neq,neq);
H = H'*H + 1e-6*eye(neq,neq);
G = randn(neq,1);
% Hmult is the name of a simple function that multiplies by H
Hfunc = 'Hmult';
S = [];
% Random starting guess
x0 = randn(neq,1);
while 1,
   Hx = feval(Hfunc,x0);
   res = G - Hx;
   fprintf('Norm of residual = %f\n',norm(res));
   Qnow = x0'*Hx - x0'*G - G'*x0;
```

```
    S(:,1) = 2*res; % Search direction along negative gradient
    [xnew,Qpred] = search1(x0,res,Hfunc,Qnow,S);
    S(:,2) = xnew - x0; % Second search direction
    x0 = xnew;
    keyboard % Type ''return'' to continue to next iteration
  end
```

Notice how the matrix of search directions $\mathbf{S}$ is set up. On the first iteration, it consists of a single column containing $2(\mathbf{G}-\mathbf{H}\mathbf{x}_0)$. This is the negative gradient at the starting guess. After the first iteration, the second column of $\mathbf{S}$ is set to $\mathbf{x}_1-\mathbf{x}_0$. On the second iteration, the first column is set to $2(\mathbf{G}-\mathbf{H}\mathbf{x}_1) \equiv -\nabla Q(\mathbf{x}_1)$ so that the function `search1` finds the minimum in the two dimensional space spanned by $-\nabla Q(\mathbf{x}_1)$ and $\mathbf{x}_1-\mathbf{x}_0$, as required. This process continues on subsequent iterations.

The function `Hmult` is simply

```
  function y = Hmult(x)
  global H
  y = H*x;
```

### 3.7.2 Application to Tikhonov Regularization

For Tikhonov regularization, we need to find the minimum of the quadratic expression

$$\mathcal{L}(\mathbf{f}) = \lambda^2 \left\|\mathbf{L}(\mathbf{f}-\mathbf{f}^\infty)\right\|^2 + \left\|\mathbf{d}-\mathbf{A}\mathbf{f}\right\|^2 \tag{3.37}$$

$$= \lambda^2 \Omega(\mathbf{f}) + C(\mathbf{f}) \tag{3.38}$$

where we shall assume that $\lambda$ has been given. One can apply the algorithm discussed above directly to this problem by multiplying out the norms in order to find the matrices $\mathbf{H}$ and $\mathbf{G}$, but it is convenient to use the special form of the expression and to assume that the user can provide functions which will apply the forward operator $\mathbf{A}$ and the operator $\mathbf{L}$ to any given vector.

Starting from $\mathbf{f}_0$ and searching within a subspace spanned by the columns of $\mathbf{S}$ as before, we wish to consider

$$\mathcal{L}(\mathbf{f}_0+\mathbf{S}\mathbf{c}) = \lambda^2\Omega(\mathbf{f}_0+\mathbf{S}\mathbf{c}) + C(\mathbf{f}_0+\mathbf{S}\mathbf{c}) \tag{3.39}$$

Substituting into the expression for $\Omega$, we find

$$\Omega(\mathbf{f}_0+\mathbf{S}\mathbf{c}) = \left\|\mathbf{L}(\mathbf{f}_0+\mathbf{S}\mathbf{c}-\mathbf{f}^\infty)\right\|^2 \tag{3.40}$$

$$= (\mathbf{f}_0+\mathbf{S}\mathbf{c}-\mathbf{f}^\infty)^{\mathrm{t}}\,\mathbf{L}^{\mathrm{t}}\mathbf{L}(\mathbf{f}_0+\mathbf{S}\mathbf{c}-\mathbf{f}^\infty) \tag{3.41}$$

$$= \mathbf{c}^{\mathrm{t}}\mathbf{S}^{\mathrm{t}}\mathbf{L}^{\mathrm{t}}\mathbf{L}\mathbf{S}\mathbf{c} + \mathbf{c}^{\mathrm{t}}\mathbf{S}^{\mathrm{t}}\mathbf{L}^{\mathrm{t}}\mathbf{L}(\mathbf{f}_0-\mathbf{f}^\infty) + (\mathbf{f}_0-\mathbf{f}^\infty)^{\mathrm{t}}\,\mathbf{L}^{\mathrm{t}}\mathbf{L}\mathbf{S}\mathbf{c}+\Omega(\mathbf{f}_0) \tag{3.42}$$

$$= \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{H}}_\Omega\mathbf{c} - \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{G}}_\Omega - \tilde{\mathbf{G}}_\Omega^{\mathrm{t}}\mathbf{c}+\Omega(\mathbf{f}_0) \tag{3.43}$$

where

$$\tilde{\mathbf{H}}_\Omega = \mathbf{S}^{\mathrm{t}}\mathbf{L}^{\mathrm{t}}\mathbf{L}\mathbf{S} = (\mathbf{L}\mathbf{S})^{\mathrm{t}}\,(\mathbf{L}\mathbf{S}) \tag{3.44}$$

$$\tilde{\mathbf{G}}_\Omega = -\mathbf{S}^{\mathrm{t}}\mathbf{L}^{\mathrm{t}}\mathbf{L}(\mathbf{f}_0-\mathbf{f}^\infty) = -(\mathbf{L}\mathbf{S})^{\mathrm{t}}\,\mathbf{L}(\mathbf{f}_0-\mathbf{f}^\infty) \tag{3.45}$$

Similarly using the expression for $C$, we find

$$C(\mathbf{f}_0+\mathbf{S}\mathbf{c}) = \left\|\mathbf{d}-\mathbf{A}(\mathbf{f}_0+\mathbf{S}\mathbf{c})\right\|^2 \tag{3.46}$$

$$= \{\mathbf{d}-\mathbf{A}(\mathbf{f}_0+\mathbf{S}\mathbf{c})\}^{\mathrm{t}}\,\{\mathbf{d}-\mathbf{A}(\mathbf{f}_0+\mathbf{S}\mathbf{c})\} \tag{3.47}$$

$$= \mathbf{c}^{\mathrm{t}}\mathbf{S}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}\mathbf{S}\mathbf{c} - \mathbf{c}^{\mathrm{t}}\mathbf{S}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}(\mathbf{d}-\mathbf{A}\mathbf{f}_0) - (\mathbf{d}-\mathbf{A}\mathbf{f}_0)^{\mathrm{t}}\,\mathbf{A}\mathbf{S}\mathbf{c}+C(\mathbf{f}_0) \tag{3.48}$$

$$= \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{H}}_C\mathbf{c} - \mathbf{c}^{\mathrm{t}}\tilde{\mathbf{G}}_C - \tilde{\mathbf{G}}_C^{\mathrm{t}}\mathbf{c}+C(\mathbf{f}_0) \tag{3.49}$$

where

$$\tilde{\mathbf{H}}_C = \mathbf{S}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}\mathbf{S} = (\mathbf{A}\mathbf{S})^{\mathrm{t}}(\mathbf{A}\mathbf{S}) \tag{3.50}$$

$$\tilde{\mathbf{G}}_C = \mathbf{S}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}(\mathbf{d}-\mathbf{A}\mathbf{f}_0) = (\mathbf{A}\mathbf{S})^{\mathrm{t}}\mathbf{r}_0 \text{ where } \mathbf{r}_0 = \mathbf{d}-\mathbf{A}\mathbf{f}_0 \tag{3.51}$$

Thus

$$\mathcal{L}(\mathbf{f}_0+\mathbf{S}\mathbf{c}) = \mathbf{c}^{\mathrm{t}}\left(\lambda^2\tilde{\mathbf{H}}_\Omega+\tilde{\mathbf{H}}_C\right)\mathbf{c} - \mathbf{c}^{\mathrm{t}}\left(\lambda^2\tilde{\mathbf{G}}_\Omega+\tilde{\mathbf{G}}_C\right) - \left(\lambda^2\tilde{\mathbf{G}}_\Omega+\tilde{\mathbf{G}}_C\right)^{\mathrm{t}}\mathbf{c} + \mathcal{L}(\mathbf{f}_0) \tag{3.52}$$

The minimum in the subspace occurs where

$$\hat{\mathbf{c}} = \left(\lambda^2\tilde{\mathbf{H}}_\Omega+\tilde{\mathbf{H}}_C\right)^{-1}\left(\lambda^2\tilde{\mathbf{G}}_\Omega+\tilde{\mathbf{G}}_C\right) \tag{3.53}$$

and so we set

$$\mathbf{f}_1 = \mathbf{f}_0 + \mathbf{S}\left(\lambda^2\tilde{\mathbf{H}}_\Omega+\tilde{\mathbf{H}}_C\right)^{-1}\left(\lambda^2\tilde{\mathbf{G}}_\Omega+\tilde{\mathbf{G}}_C\right) \tag{3.54}$$

These considerations are illustrated in the Matlab program included below

```
function [fnew,cpred,wpred] = subsearch(f0,res,fdef,Afunc,Lfunc,lambda,S)
% Searches in subspace spanned by columns of S for the optimal solution
%  of the regularization problem
%   (lambda)^2*||L*(f-fdef)||^2 + ||d-A*f||^2
% f0     = Initial guess at location of the minimum
% res    = d - A*f0, the current residual
% fdef   = The default image
% Afunc  = Name of function which applies A to a vector
% Lfunc  = Name of function which applies L to a vector
% lambda = Weighting between solution seminorm and data misfit
% S      = matrix with search directions as its columns
%
% fnew   = Position of minimum within subspace
% cpred  = Predicted value of ||d-A*fnew||^2
% wpred  = Predicted value of ||L*(fnew-fdef)||^2

% Sze Tan, University of Auckland, July 1998

nsrch = size(S,2);
pref  = feval(Lfunc,f0-fdef);
w0 = pref' * pref;
c0 = res' * res;
fprintf('Square of regularization semi-norm (current)   = %f\n',w0);
fprintf('Square of data misfit norm (current)           = %f\n',c0);
AS = zeros(length(res),nsrch);
LS = zeros(length(pref),nsrch);
for k = 1 : nsrch
   AS(:,k) = feval(Afunc,S(:,k));
   LS(:,k) = feval(Lfunc,S(:,k));
end
Hc = AS' * AS;  Hw = LS' * LS;
Gc = AS' * res; Gw = -LS'*pref;
c = (Hc + lambda^2 * Hw) \ (Gc + lambda^2 *Gw);
cpred = c0 + c'*Hc*c - c'*Gc - Gc'*c;
wpred = w0 + c'*Hw*c - c'*Gw - Gw'*c;
fprintf('Square of regularization semi-norm (predicted) = %f\n',wpred);
fprintf('Square of data misfit norm (predicted)         = %f\n\n',cpred);
fnew = f0 + S * c;
```

Notice that for the minimization in the subspace, it is only necessary to be able to apply $\mathbf{A}$ and $\mathbf{L}$ to vectors. It only remains for us to calculate the search directions for the minimization. The negative gradient of $\mathcal{L}$ is

$$-\nabla\mathcal{L}(\mathbf{f}) = -\lambda^2\mathbf{L}^{\mathrm{t}}\mathbf{L}\left(\mathbf{f}-\mathbf{f}^{\infty}\right)+\mathbf{A}^{\mathrm{t}}\left(\mathbf{d}-\mathbf{A}\mathbf{f}\right)$$

In order to calculate this, we also need functions which will apply $\mathbf{L}^{\mathrm{t}}$ and $\mathbf{A}^{\mathrm{t}}$ to a vector. A method which works reasonably well in practice is to calculate $\mathbf{L}^{\mathrm{t}}\mathbf{L}\left(\mathbf{f}_l-\mathbf{f}^{\infty}\right)$, $\mathbf{A}^{\mathrm{t}}\left(\mathbf{d}-\mathbf{A}\mathbf{f}_l\right)$ as search directions on the first iteration ($l=0$), and to append the direction $\mathbf{f}_l-\mathbf{f}_{l-1}$ on subsequent iterations. This is illustrated in the code fragment below:

```
S = [];
while 1,
   pref = feval(Lfunc,f - fdef);
   res  = data - feval(Afunc,f);
   S(:,1) = feval(Ahfunc,res); S(:,2) = -feval(Lhfunc,pref);
   test = 1 - abs(S(:,1)'*S(:,2)./(norm(S(:,1))*norm(S(:,2))));
   fprintf('Test statistic = %f\n',test);
   [fnew,cpred,spred] = subsearch(f,res,fdef,Afunc,Lfunc,lambda,S);
   S(:,3) = fnew - f;
   f = fnew;
   keyboard   % Pause to allow user to view result
end
```

In this example, the functions whose names are in `Afunc` and `Lfunc` apply the matrices $\mathbf{A}$ and $\mathbf{L}$ to a vector, while the functions whose names are in `Ahfunc` and `Lhfunc` apply the (conjugate) transposes ($\mathbf{A}^{\mathrm{H}}$ and $\mathbf{L}^{\mathrm{H}}$) to a vector. The search directions are placed in the columns of $\mathbf{S}$. The quantity `test` indicates whether the vectors $\nabla C$ and $\nabla\Omega$ are parallel to each other, as they should be at the optimal point. When the value of `test` is sufficiently small, the algorithm may be deemed to have converged. The figures below show a reconstruction based on the blurred image discussed in the first chapter, with the added complication that half the points of the blurred image are now assumed to be unmeasured or corrupted. In Figure 3.4, all unmeasured points are shown as black. The forward problem is as before, except that after the convolution is complete, the unmeasured points are discarded before comparison with the data. In this case, the Fourier division method fails completely, but a regularized reconstruction based on the above ideas still allows the message to be read as shown in Figure 3.5.

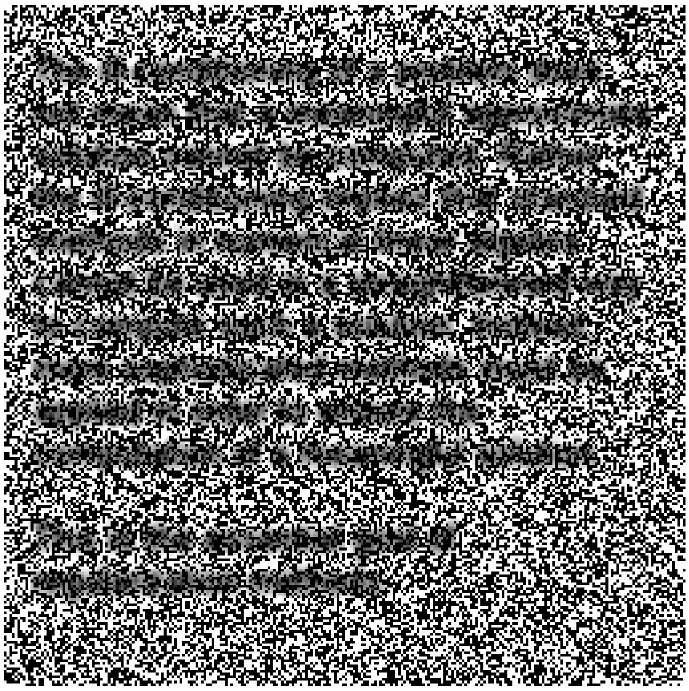
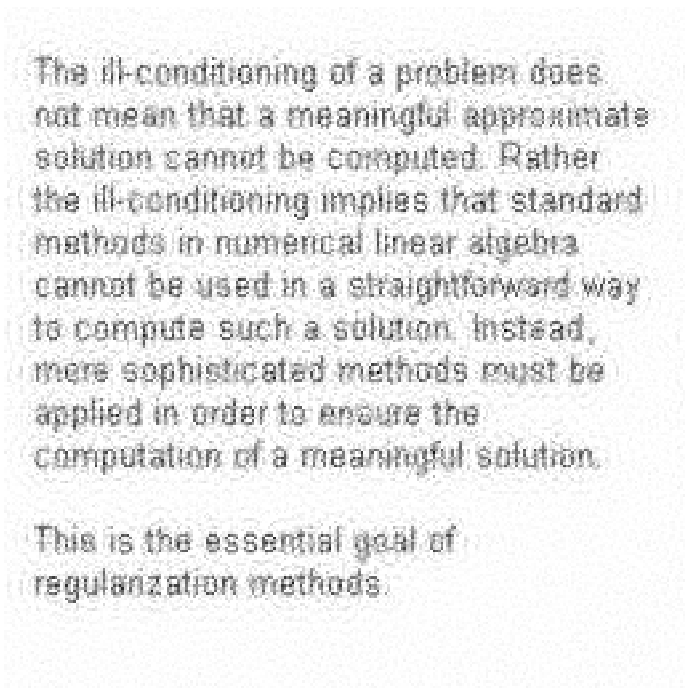


Figure 3.4 Blurred image with missing data. Every black pixel indicates a data point that was unmeasured. In this image, approximately half the pixels are unmeasured.



The ill-conditioning of a problem does
not mean that a meaningful approximate
solution cannot be computed. Rather
the ill-conditioning implies that standard
methods in numerical linear algebra
cannot be used in a straightforward way
to compute such a solution. Instead,
more sophisticated methods must be
applied in order to ensure the
computation of a meaningful solution.

This is the essential goal of
regularization methods.

Figure 3.5 Reconstruction from the above data set