# Optimisation and Inverse Problems in Imaging

Simon R. Arridge

March 2, 2016

† *Department of Computer Science, University College London, Gower Street London, WC1E 6BT*

**GV08**

*2007-2016*

# 1 Introduction

The topic of Optimisation covers many areas. In this course we are going to consider the role of Optimisation in imaging. In particular (but not exclusively) we are going to consider the role of optimisation in the topic of *Inverse Problems*, and within this subject we are going to consider in particular (but not exclusively) the topic of *Inverse Problems in Imaging*.

Broadly speaking Optimisation requires us to consider four things

- A *metric*. This addresses the question "Optimisation *of* what?". A metric is a single real positive number that acts as a "figure of merit" of our success at optimisation.
- A *model*. This addresses the question "what is the metric applied to?". A model is a mathematical or computational process that predicts a result which maybe a function, vector, list or some other quantity.
- *Parameters*. This addresses the question "Optimisation *over* what ?". These are the variable controls of the model. The aim of optimisation is to adjust the parameters until the figure of metric is in some way "as good as we can get"
- An *algorithm*. This addresses the question "How do we optimise?"

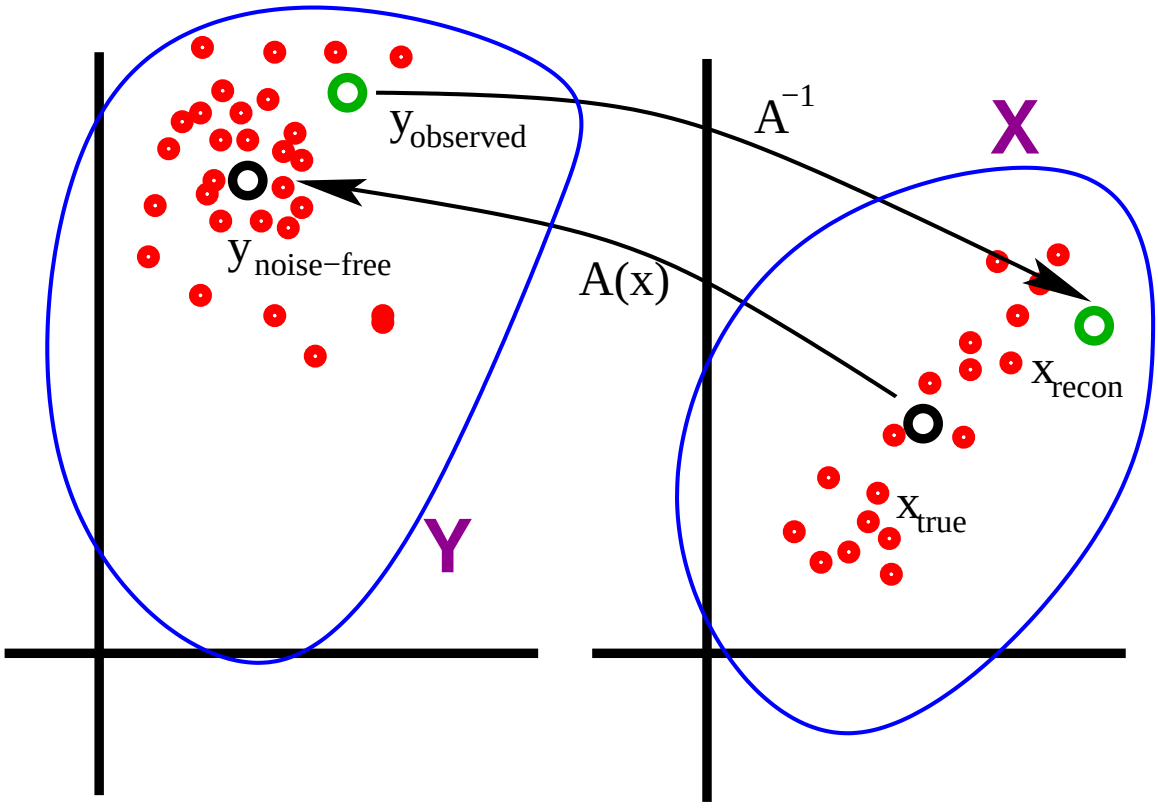



Figure 1.1:

There are many generalisations. We may sometimes consider not one but multiple metrics. An important type of optimisation is *model-fitting*. In this case we need a 5th concept: *Data*. In this case the metric is not just applied to the model, but to the model and the data together to evaluate how well the model predicts the data.

# 2 Basic Concepts

## 2.1 Some examples

### 2.1.1 Minimum norm solutions to Underdetermined Problems

Consider the following problem : Solve

$$x_1 + 2x_2 = 5 \tag{2.1}$$

There is no unique solution. The set of possible solutions forms a line on the $x_1, x_2$ plane. However the following can be solved :

$$\begin{aligned} \text{minimise} \quad & \Phi = x_1^2 + x_2^2 \\ \text{subject to} \quad & x_1 + 2x_2 = 5 \end{aligned} \tag{2.2}$$

which has solution $(x_1, x_2) = (1, 2)$. It is illustrated graphically in figure 2.1. This is an example

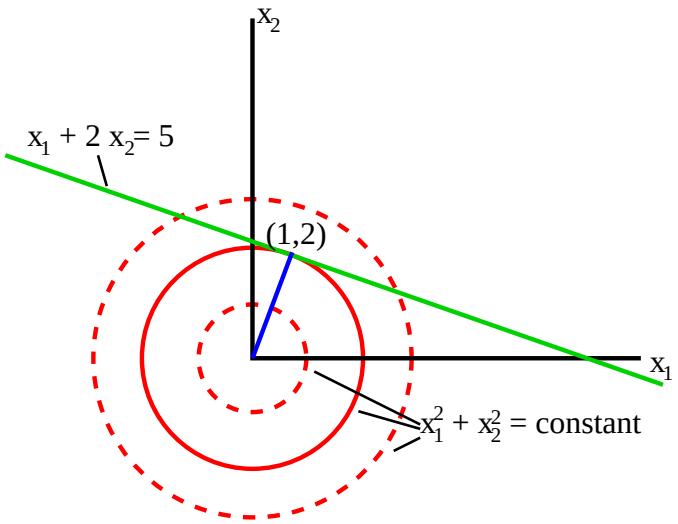



Figure 2.1:

of *constrained optimisation*. The unique solution is the one where the surface $\Phi$ = constant is tangent to the constraint $x_1 + 2x_2 = 5$. The expression $\Phi = x_1^2 + x_2^2$ is a *norm* (i.e. a metric, or distance measure) on the Euclidean space $\mathbb{R}^2$ with basis vectors $\hat{\boldsymbol{x}}_1 = (1, 0)$, $\hat{\boldsymbol{x}}_2 = (0, 1)$. In this

case, because the norm is quadratic, the solution is found by taking the *Moore-Penrose generalised inverse* of the underdetermined linear equation Eq. 2.1 :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \left( (12) \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right)^{-1} (5) = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{2.3}$$

and the solution is the *minimum norm* solution, where the norm is the normal Euclidean $L_2$ norm.

Now consider that we take a different norm and ask for the solution to

$$\begin{aligned} \text{minimise} \quad & \Phi = x_1^p + x_2^p \\ \text{subject to} \quad & x_1 + 2x_2 = 5\,. \end{aligned} \tag{2.4}$$

When $1 < p < \infty$ the surfaces $\Phi$ = constant are *convex* and solutions can be found relatively easily. We see that we can obtain any solution between $(0, 2.5)$ at $p = 1$ and $(1\frac{2}{3}, 1\frac{2}{3})$ for $p = \infty$.

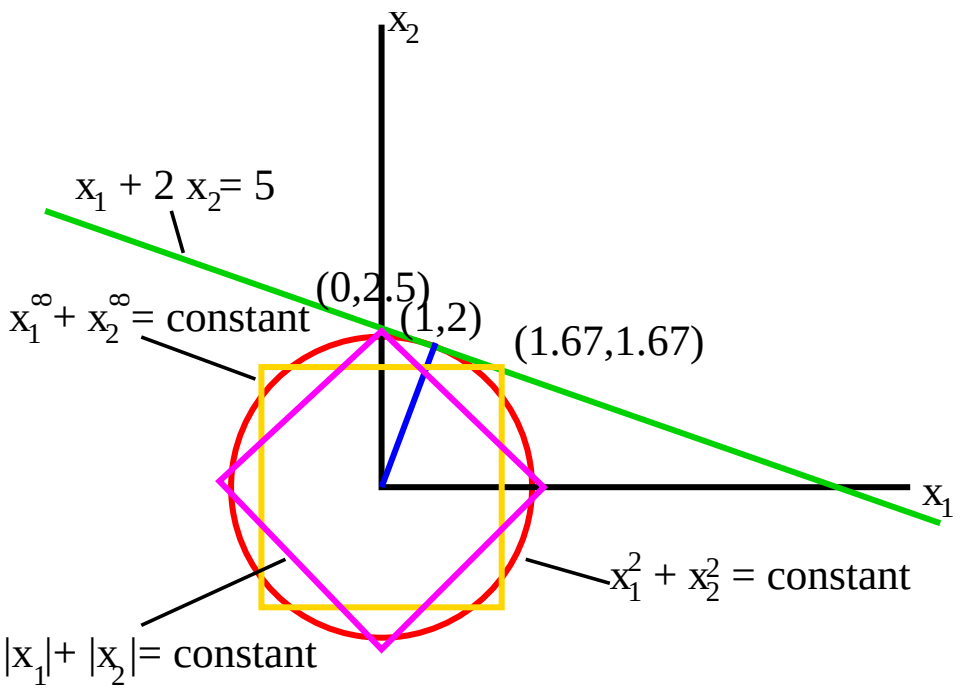



Figure 2.2:

However, actually calculating the solution is not as simple as solving a linear problem like Eq. 2.3. The case $p = 1$ is known as the *sparse* solution.

The case $0 < p < 1$ leads to *concave* surfaces and to the more complex problem of *non-convex* optimisation.

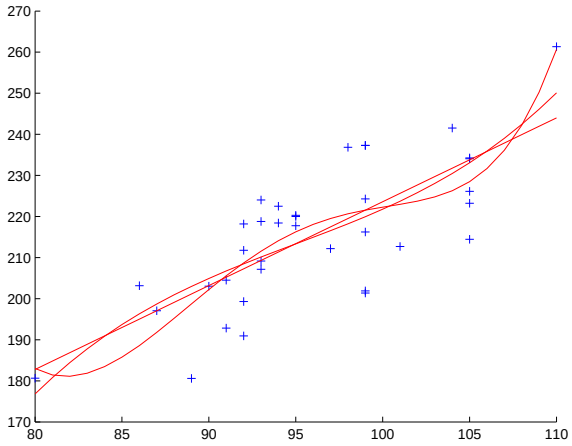



Figure 2.3: Fitting polynomials of increasing order to a set of points

### 2.1.2 Model Fitting

Suppose we have a set of data $\{x_i, y_i; i = 1 \dots N\}$ and a hypothetical model

$$y = A(x; \boldsymbol{\xi})$$

parameterised by values $\{\xi_0, \xi_1, ...\xi_M\}$. We can provide a fit of the model by optimising a data fitting function such as

$$\Phi = \frac{1}{2} \sum_{i=1}^{N} |y_i - A(x_i, \boldsymbol{\xi})|^2 \tag{2.5}$$

see figure 2.3. More generally, we may define a *data-fitting* functional

$$\Phi = \mathcal{D}(\boldsymbol{y}, A(\boldsymbol{x}, \boldsymbol{\xi}))\,. \tag{2.6}$$

This framework admits, for example, the $p$-norms we introduced in Eq.(2.4), or they may be more general norms such as Kullback-Liebeck divergence, or Cross-entropy. In general the choice of norm in the space of the *data* is dicated by the expected statistics of the errors in the data. A choice of the $L_2$ norm corresponds to assuming *identically distributed, independent Gaussian random variable* for the errors (iid Normal distributions).

If the model is linear, and the data-fitting functional is sum of squares the problem is *linear least-squares* and can be solved using linear algebra. For example, fitting a cubic polynomial $y = \xi_0 + \xi_1 x + \xi_2 x^2 + \xi_3 x^3$ to $N > 4$ data is an *over-determined* problem defined by

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 \end{pmatrix} \begin{pmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} \tag{2.7}$$

If $A$ is non-linear with respect to the parameters $\boldsymbol{\xi}$ then the process is one of *non-linear, unconstrained optimisation*. This could be non-linear least-squares, or a more general problem.

### 2.1.3 Inequality constraints

Consider the same problem as in section 2.1.1, but introduce bound on the solution

$$\begin{aligned} \text{minimise} \quad & \Phi = x_1^2 + x_2^2 \\ \text{subject to} \quad & x_1 + 2x_2 = 5 \\ & 2 \leq x_1 \leq 4 \\ & 1 \leq x_2 \leq 5 \end{aligned} \tag{2.8}$$

(2.9)

which has solution $(x_1, x_2) = (2, 1.5)$, see figure 2.4. Unlike Eq. 2.2 this cannot be solved by a

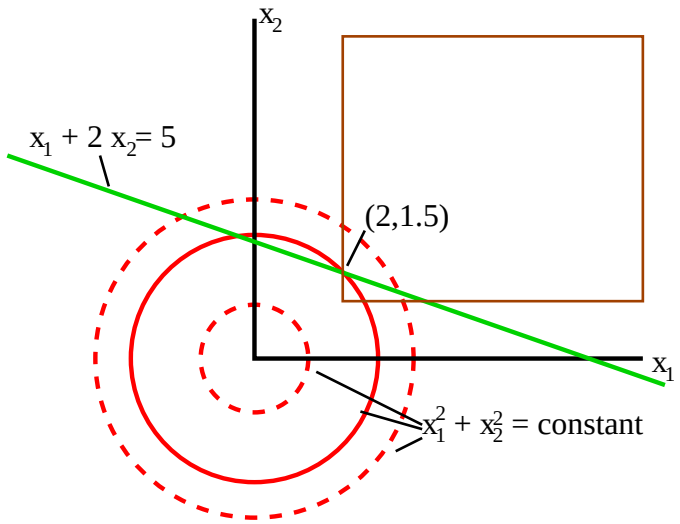



Figure 2.4:

simple linear method like Eq. 2.3.

**References for this section**
[1, 2, 3, 4]

## 2.2 Inverse Problems

### 2.2.1 Problem 1 : Non Uniqueness

Solve

$$x_1 + x_2 = 2 \tag{2.10}$$

This problem has an infinite number of solutions. The set of equations are *underdetermined*. To find a single solution we need to add constraints and use a constrained optimisation method.

### 2.2.2 Problem 2 : Non Existence

Solve

$$\begin{aligned} x_1 &= 1 \\ x_1 &= 3 \end{aligned} \tag{2.11}$$

This problem has no solution. The set of equations are *overdetermined*. We can find a possible solution using unconstrained optimisation. Consider the following problem :

$$\Phi = (x_1 - 1)^2 + (x_1 - 3)^2 \rightarrow \min \tag{2.12}$$

It is easy to see we cannot get a solution with $\Phi = 0$ because we can solve a quadratic equation exactly

$$\begin{aligned} (x_1 - 1)^2 + (x_2 - 3)^2 &= 0 \\ \Rightarrow 2x_1^2 - 8x_1 + 10 &= 0 \\ \Rightarrow x_1 &= \frac{4 \pm \sqrt{16 - 20}}{2} \\ &= 2 \pm 2i \end{aligned}$$

i.e. the solution has no real roots. Instead we can find the minimum :$x_1 = 2, \Phi = 2$. This is the *least squares solution*.

### 2.2.3 Problem 3 : Discontinuous dependence on data

Solve

$$\mathsf{A}\boldsymbol{x} = \begin{pmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{2.13}$$

This problem has a solution that tends towards instability. There are a number of ways to see this. The matrix $\mathsf{A}$ has determinant $\epsilon^2$ so its inverse is

$$\mathsf{A}^{-1} = \frac{1}{\epsilon^2} \begin{pmatrix} 1 & -(1+\epsilon) \\ -(1-\epsilon) & 1 \end{pmatrix} \tag{2.14}$$

Thus the inverse grows as an inverse square in the parameter $\epsilon$.

To see the effect of the inverse with respect to noise, consider the following example :

Results are shown in figure 2.2.3. We see that the data has an isotropic distribution around the specified noise free point whereas the solution has a highly *non-isotropic* distribution. This

**Algorithm 1** Statistical trial of ill-posed matrix inversion

**for all** trials $i = 1 \dots N$ **do**

draw a random vector of length 2 with Gaussian probability $\eta_i \sim N\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\right)$

generate data $b_i = (10, 20) + \eta_i$

solve $\mathsf{A}\boldsymbol{x}_i = b_i$

**end for**

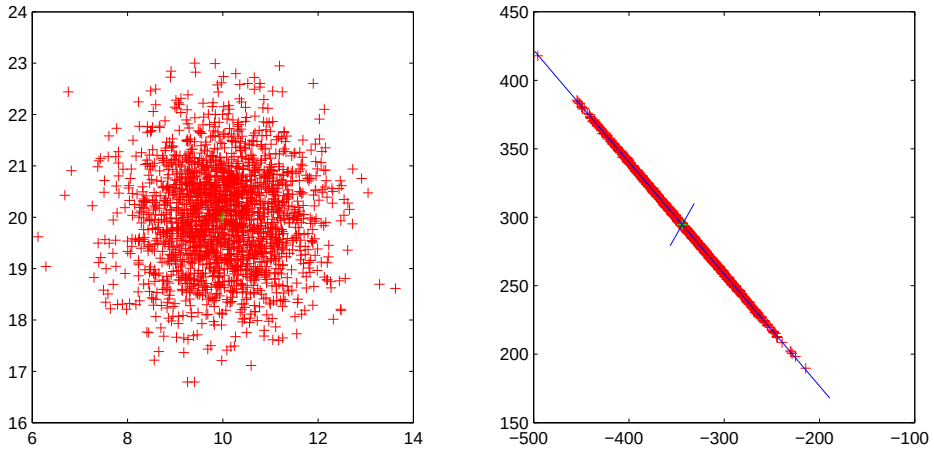



Figure 2.5: Results of ill-posed matrix inversion trial for $\epsilon \simeq 0.2$.Left : The distribution of data points. Right : the distribution of solutions. The singular values of the matrix are $2.02, 0.0202$, and the standard deviation of the covariance of the solution points are $1.9984, 0.020502$

highly anisotropic amplification of noise preferentially in some directions is an example of the third kind of ill-posedness of inverse problems. To understand the nature of this ill-posedness we can investigate the structure of the linear forward problem using the Singular Value Decomposition (SVD).

**References for this section**

[5]

## 2.3 Singular Value Decomposition (SVD)

The eigenvalues of $\mathsf{A}$ can be found by solving

$$\lambda^2 - 2\lambda + \epsilon^2 = 0 \quad \Rightarrow \quad \lambda = 1 \pm \sqrt{1 - \epsilon^2}.$$

with corresponding eigenvectors (unnormalised)

$$\boldsymbol{e}_1 = \begin{pmatrix}\sqrt{1-\epsilon^2}\\1-\epsilon\end{pmatrix}, \quad \boldsymbol{e}_2 = \begin{pmatrix}-\sqrt{1-\epsilon^2}\\1-\epsilon\end{pmatrix}$$

[NOTE : Since the matrix is not symmetric, the eigenvectors are not orthogonal]

Taking the limit of these solutions as $\epsilon \to 0$ we find the limiting values for the eigenvalues :

$$\lambda_1 \to 2 - \frac{1}{2}\epsilon\,, \quad \lambda_2 \to \frac{1}{2}\epsilon$$

with corresponding eigenvectors

$$\boldsymbol{e}_1 \to \begin{pmatrix} 1 \\ 1 \end{pmatrix}\,, \quad \boldsymbol{e}_2 \to \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

[NOTE : In the limit the matrix is symmetric, so the limiting eigenvectors are orthogonal]

We may also take the Singular Value Decomposition (SVD); In Matlab :

```
eps = 0.01;
A = [1 1+eps; 1 - eps 1];
[U,W,V] = svd(A);
```

[Note that Matlab requires numerics. The symbolic part of Matlab is not very comprehensive]

We can write the SVD explicitly

$$\begin{aligned}
\boldsymbol{u}_1 &= \frac{1}{L_+}\begin{pmatrix} \epsilon + \sqrt{1+\epsilon^2} \\ 1 \end{pmatrix} \quad , \quad \boldsymbol{u}_2 = \frac{1}{L_-}\begin{pmatrix} \epsilon - \sqrt{1+\epsilon^2} \\ 1 \end{pmatrix} \\
w_1 &= \left(2 + \epsilon^2 + 2\sqrt{1+\epsilon^2}\right)^{1/2} \quad , \quad w_2 = \left(2 + \epsilon^2 - 2\sqrt{1+\epsilon^2}\right)^{1/2} \\
\boldsymbol{v}_1 &= \frac{1}{L_-}\begin{pmatrix} \sqrt{1+\epsilon^2} - \epsilon \\ 1 \end{pmatrix} \quad , \quad \boldsymbol{v}_2 = \frac{1}{L_+}\begin{pmatrix} -\sqrt{1+\epsilon^2} - \epsilon \\ 1 \end{pmatrix}
\end{aligned}$$

where $L_+ = \left(1 + (\epsilon + \sqrt{1+\epsilon^2})^2\right)^{1/2}$, and $L_- = \left(1 + (\epsilon - \sqrt{1+\epsilon^2})^2\right)^{1/2}$ are normalisations ; [Note that normalisation must be carried out for SVD to provide the correct mappings between left and right spaces].

***Exercise* 1** *Verify the above SVD. I.e. show*

$$\mathsf{A}\boldsymbol{v}_i = w_i \boldsymbol{u}_i\,, \quad \mathsf{A}^{\mathrm{T}}\boldsymbol{u}_i = w_i \boldsymbol{v}_i \qquad i = 1, 2$$

The SVD provides an alternative way to express the matrix inverse, and works also for non square matrices.

### 2.3.1 Properties of SVD for matrices

The SVD can be defined for any linear operator. To begin with we will consider matrices. The SVD of any $N \times M$ matrix $\mathsf{A}$ can be written as a product

$$\underbrace{\mathsf{A}}_{N\times M} = \underbrace{\mathsf{U}}_{N\times N}\underbrace{\mathsf{W}}_{N\times M}\underbrace{\mathsf{V}^{\mathrm{T}}}_{M\times M} \tag{2.15}$$

The columns of $\mathsf{U}$ are orthonormal, as are the columns of $\mathsf{V}$. I.e. if we consider

$$\mathsf{U} = [\boldsymbol{u}_1\, \boldsymbol{u}_1\, \ldots\, \boldsymbol{u}_N] = \begin{bmatrix} \begin{pmatrix} u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{1,N} \end{pmatrix} & \begin{pmatrix} u_{2,1} \\ u_{2,2} \\ \vdots \\ u_{2,N} \end{pmatrix} & \ldots & \begin{pmatrix} u_{N,1} \\ u_{N,2} \\ \vdots \\ u_{N,N} \end{pmatrix} \end{bmatrix}$$

and

$$\mathsf{V} = [\boldsymbol{v}_1\, \boldsymbol{v}_1\, \ldots\, \boldsymbol{v}_M] = \begin{bmatrix} \begin{pmatrix} v_{1,1} \\ v_{1,2} \\ \vdots \\ v_{1,M} \end{pmatrix} & \begin{pmatrix} v_{2,1} \\ v_{2,2} \\ \vdots \\ v_{2,M} \end{pmatrix} & \ldots & \begin{pmatrix} v_{M,1} \\ v_{M,2} \\ \vdots \\ v_{M,M} \end{pmatrix} \end{bmatrix}$$

Then we have

$$\boldsymbol{u}_i \cdot \boldsymbol{u}_j = \delta_{ij} \quad i,j \in [1 \ldots N]\,, \qquad \boldsymbol{v}_i \cdot \boldsymbol{v}_j = \delta_{ij} \quad i,j \in [1 \ldots M]\,.$$

The matrix $\mathsf{W}$ has a structure depending on whether $N$ is greater than, equal to, or less than $M$.

If $N = M$, $\mathsf{W}$ is square, with diagonal elements $W_{i,i} = w_i, i = 1 \ldots N$
If $N > M$, $\mathsf{W}$ has a $M \times M$ diagonal matrix above $N - M$ rows of zero :

$$\mathsf{W} = \begin{pmatrix} \mathrm{diag}(\boldsymbol{w}) \\ \mathsf{0} \end{pmatrix} = \begin{pmatrix} w_1 & 0 & 0 & \ldots & 0 \\ 0 & w_2 & 0 & \ldots & 0 \\ 0 & 0 & w_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & w_M \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

If $N < M$, $\mathsf{W}$ has a $N \times N$ diagonal matrix to the left of $M - N$ columns of zero :

$$\mathsf{W} = \begin{pmatrix} \mathrm{diag}(\boldsymbol{w}) & \mathsf{0} \end{pmatrix} = \begin{pmatrix} w_1 & 0 & 0 & \dots & 0 & 0 \dots & 0 \\ 0 & w_2 & 0 & \dots & 0 & 0 \dots & 0 \\ 0 & 0 & w_3 & \dots & 0 & 0 \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \\ 0 & 0 & 0 & \dots & w_N & 0 \dots & 0 \end{pmatrix}$$

The number of non-zeros in the diagonal part of $\mathsf{W}$ is equal to the *rank* of the matrix $\mathsf{A}$ i.e. the number of linearly independent columns (or rows) of the matrix

$$R = \mathrm{rank}(\mathsf{A}) \leq min(N, M)$$

For convenience, assume that the linearly independent columns of $\mathsf{U}$, $\mathsf{V}$ are arranged as the first $R$ components, and further that the indexing of columns is such that $|w_i| > |w_{i+1}|$. The elements of the diagonal part of $\mathsf{W}$ are termed the *spectrum* of $\mathsf{A}$.

The $R$ column-vectors of $\mathsf{V}$ corresponding to the non-zero elements of the diagonal of $\mathsf{W}$ form an orthonormal basis spanning a $R$-dimensional vector space called the *row-space*, and the $R$ column-vectors of $\mathsf{U}$ corresponding to these elements form an orthonormal basis spanning a $R$-dimensional vector space called the *range* range($\mathsf{A}$) or *column-space*. These basis functions are related through

$$\mathsf{A}\boldsymbol{v}_i = w_i \boldsymbol{u}_i \,, \quad \mathsf{A}^{\mathrm{T}}\boldsymbol{u}_i = w_i \boldsymbol{v}_i \qquad i = 1 \dots R \tag{2.16}$$

The remaining $M - R$ column vectors of $\mathsf{V}$ (if greater than zero) form an orthonormal basis spanning a $M - R$-dimensional vector space called the *null-space* Null($\mathsf{A}$), or *kernel* kern($\mathsf{A}$). For any vector $\boldsymbol{x}_\perp \in \mathrm{Null}(\mathsf{A})$ we have

$$\mathsf{A}\boldsymbol{x}_\perp = \mathbf{0} \tag{2.17}$$

Similarly, the $N - R$ column vectors of $\mathsf{U}$ (if greater than zero) form an orthonormal basis spanning a $N - R$-dimensional vector space called the *range complement* range$_\perp$($\mathsf{A}$) or *co-kernel*. For any vector $\boldsymbol{b}_\perp \in \mathrm{range}_\perp(\mathsf{A})$ we have that the linear equation

$$\mathsf{A}\boldsymbol{x} = \boldsymbol{b}_\perp \tag{2.18}$$

has no solutions.

For the simple case $N = M = R$ the matrix $\mathsf{A}$ is square and full-rank and its inverse can be expressed straightforwardly

$$\mathsf{A}^{-1} = \mathsf{V}\mathsf{W}^{-1}\mathsf{U}^{\mathrm{T}} \tag{2.19}$$

where $\mathsf{W}^{-1}$ is diagonal with components $1/w_i, i = 1 \dots R$. This representation allows to express the solution of a particular problem

$$\mathsf{A}\,\boldsymbol{x} = \boldsymbol{b} \;\rightarrow \boldsymbol{x} = \sum_{i=1}^{R} \frac{\boldsymbol{u}_i \cdot \boldsymbol{b}}{w_i} \boldsymbol{v}_i \tag{2.20}$$

Now we can see a problem that is related to the third definition of an ill-posed problem given above : even if $\mathsf{A}$ is full rank, if the relative magnitudes of the $w_i$ decrease rapidly, then the correponding components of the inverse increase at the same rate. In particular, if the decay of the spectrum is bounded by a geometric ratio then the inverse problem is termed *exponentially ill-posed*.

### 2.3.2 Pseudo-Inverse

The inverse matrix built from the row-space of $\mathsf{A}$ is called the *Moore-Penrose pseudo-inverse* of $\mathsf{A}$.

$$\mathsf{A}^\dagger = \mathsf{V}\mathsf{W}^\dagger\mathsf{U}^{\mathrm{T}} = \sum_{i=1}^{R} \frac{\boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}}}{w_i} \tag{2.21}$$

Here $\mathsf{W}^\dagger$ is a $M \times N$ matrix with the complementary structure to $\mathsf{W}$. To be specific :

If $N = M = R$, $\mathsf{W}^\dagger$ is square, with diagonal elements $W^\dagger_{i,i} = 1/w_i, i = 1 \ldots N$

If $N > M$, $\mathsf{W}^\dagger$ has a $M \times M$ diagonal matrix to the left of $N - M$ columns of zero :

$$\mathsf{W}^\dagger = \begin{pmatrix} \mathrm{diag}(\mathbf{1}/\boldsymbol{w}) & \mathsf{0} \end{pmatrix} = \begin{pmatrix} 1/w_1 & 0 & 0 & \ldots & 0 & 0 \ldots & 0 \\ 0 & 1/w_2 & 0 & \ldots & 0 & 0 \ldots & 0 \\ 0 & 0 & 1/w_3 & \ldots & 0 & 0 \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & & \\ 0 & 0 & 0 & \ldots & 1/w_M & 0 \ldots & 0 \end{pmatrix}$$

If $N < M$, $\mathsf{W}^\dagger$ has a $N \times N$ diagonal matrix above $M - N$ rows of zero :

$$\mathsf{W}^\dagger = \begin{pmatrix} \mathrm{diag}(\mathbf{1}/\boldsymbol{w}) \\ \mathsf{0} \end{pmatrix} = \begin{pmatrix} 1/w_1 & 0 & 0 & \ldots & 0 \\ 0 & 1/w_2 & 0 & \ldots & 0 \\ 0 & 0 & 1/w_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1/w_N \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

***Exercise* 2** *Write down the stuctures of the matrices* $\mathsf{W}\mathsf{W}^\dagger$ *and* $\mathsf{W}^\dagger\mathsf{W}$ *for the cases* $N > M$ *and* $M < N$ *and* $M = N$*. Assume for the sake of illustration that the rank* $R < min(M, N)$*, i.e. that there is a non-trivial null space and range complement space even in the case where* $M = N$*.*

The pseudo-inverse can always be constructed regardless if $\mathsf{A}$ is non-square and/or rank-deficient. Thus any linear matrix problem has a pseudo-solution

$$\mathsf{A}\,\boldsymbol{x} = \boldsymbol{b} \;\rightarrow \boldsymbol{x}^\dagger = \mathsf{A}^\dagger \boldsymbol{b} \tag{2.22}$$

This solution has the property that it is the *least-squares, minimum norm* solution in the sense that

$$|\boldsymbol{b} - \mathsf{A}\,\boldsymbol{x}^{\dagger}|^2 \rightarrow min\,, |\boldsymbol{x}^{\dagger}|^2 \rightarrow min$$

To see this, first we note :

$$\begin{aligned} \boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{\dagger} &= \left[\mathsf{I} - \mathsf{A}\mathsf{A}^{\dagger}\right]\boldsymbol{b} \\ &= \mathsf{U}\left[\mathsf{I} - \mathsf{W}\mathsf{W}^{\dagger}\right]\mathsf{U}^{\mathrm{T}}\boldsymbol{b} \\ &= \mathsf{P}_{\perp}\boldsymbol{b} \end{aligned}$$

where

$$\mathsf{P}_{\perp} = \sum_{i=R+1}^{N} \boldsymbol{u}_i \boldsymbol{u}_i^{\mathrm{T}} \tag{2.23}$$

is a matrix that projects data space vectors into the orthogonal complement of the range. Now consider another solution space vector $\tilde{\boldsymbol{x}} = \boldsymbol{x}^{\dagger} + \boldsymbol{x}'$, with $\boldsymbol{b}' = \mathsf{A}\boldsymbol{x}'$ the corresponding data space vector, which by definition lies in the range of $\mathsf{A}$. We have

$$\begin{aligned} |\boldsymbol{b} - \mathsf{A}\tilde{\boldsymbol{x}}|^2 &= \left|\boldsymbol{b} - \mathsf{A}\boldsymbol{x}^{\dagger} - \mathsf{A}\boldsymbol{x}'\right|^2 \\ &= |\mathsf{P}_{\perp}\boldsymbol{b} - \mathsf{U}\mathsf{U}^{\mathrm{T}}\boldsymbol{b}'|^2 \\ &= \sum_{i=R+1}^{N} (\boldsymbol{u}_i \cdot \boldsymbol{b})^2 + \sum_{i=1}^{R} (\boldsymbol{u}_i \cdot \boldsymbol{b}')^2 \end{aligned}$$

where the second sum on the right holds due to the fact that $\boldsymbol{b}' \in$ Range($\mathsf{A}$). Thus any vector $\boldsymbol{x}'$ with a non-zero component in the Null-space of $\mathsf{A}$ will increase the data space norm. We now need to see if any vector $\boldsymbol{x}' \in$ Null($\mathsf{A}$) could give a smaller norm $|\tilde{x}|$. We write

$$\begin{aligned} |\tilde{\boldsymbol{x}}|^2 &= \left|\boldsymbol{x}^{\dagger} + \boldsymbol{x}'\right|^2 \\ &= \left|\mathsf{V}\mathsf{W}^{\dagger}\mathsf{U}^{\mathrm{T}}\boldsymbol{b} + \mathsf{V}\mathsf{V}^{\mathrm{T}}\boldsymbol{x}'\right|^2 \\ &= \left|\mathsf{V}\left[\mathsf{W}^{\dagger}\mathsf{U}^{\mathrm{T}}\boldsymbol{b} + \mathsf{V}^{\mathrm{T}}\boldsymbol{x}'\right]\right|^2 \\ &= \left|\left[\mathsf{W}^{\dagger}\mathsf{U}^{\mathrm{T}}\boldsymbol{b} + \mathsf{V}^{\mathrm{T}}\boldsymbol{x}'\right]\right|^2 \\ &= \sum_{i=1}^{R} \left(\frac{\boldsymbol{u}_i \cdot \boldsymbol{b}}{w_i}\right)^2 + \sum_{i=R+1}^{M} (\boldsymbol{v}_i \cdot \boldsymbol{x}')^2 \end{aligned}$$

where the second sum on the right holds due to the fact that $\boldsymbol{x}' \in$ Null($\mathsf{A}$). Thus any vector $\boldsymbol{x}'$ with a non-zero component in the null-space of $\mathsf{A}$ will increase the solution space norm.

To complete our exposition, let us solve the three problems in section 2.2 using the pseudo-inverse.

### 2.3.3 Problem 1

$$\begin{aligned} \mathsf{A} = \begin{pmatrix} 1 & 1 \end{pmatrix} \quad &\Rightarrow \quad \mathsf{A}^\dagger = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ \text{SVD:} \quad & (1)\begin{pmatrix} \sqrt{2} & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\ \Rightarrow \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathsf{A}^\dagger(2) \quad &= \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{aligned}$$

The row-space is spanned by vector $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and the null space is spanned by vector $\frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

### 2.3.4 Problem 2

$$\begin{aligned} \mathsf{A} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad &\Rightarrow \quad \mathsf{A}^\dagger = \frac{1}{2}\begin{pmatrix} 1 & 1 \end{pmatrix} \\ \text{SVD:} \quad & \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} \sqrt{2} \\ 0 \end{pmatrix}(1) \\ \Rightarrow \quad x_1 = \mathsf{A}^\dagger \begin{pmatrix} 1 \\ 3 \end{pmatrix} \quad &= \quad 2 \end{aligned}$$

The range is spanned by vector $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and the complement of the range is spanned by vector $\frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

### 2.3.5 Problem 3

The matrix inverse was given in Eq. 2.14 so we do not need to write it again. However, by considering it as

$$\mathsf{A}^\dagger(\boldsymbol{b}_0 + \boldsymbol{\eta}) \quad = \quad \mathsf{A}^\dagger \boldsymbol{b}_0 + \frac{\boldsymbol{\eta}\cdot\boldsymbol{u}_1}{w_1}\boldsymbol{v}_1 + \frac{\boldsymbol{\eta}\cdot\boldsymbol{u}_2}{w_2}\boldsymbol{v}_2$$

We see that the solution can be considered as a random vector with mean the noise free vector $\mathsf{A}^\dagger\boldsymbol{b}_0$ and covariance given by

$$\mathsf{C} = \frac{\boldsymbol{v}_1\boldsymbol{v}_1^{\mathrm{T}}}{w_1^2} + \frac{\boldsymbol{v}_2\boldsymbol{v}_2^{\mathrm{T}}}{w_2^2} = \mathsf{V}\mathsf{W}^{-2}\mathsf{V}^{\mathrm{T}}$$

## 2.4 SVD for Operators

Most of what we have discussed for matrices applies also to Linear Operators.

Let us restrict the discussion to linear integral operators $A : X \mapsto Y$

$$b = A\,f \quad \equiv \quad b(y) = \int_{-\infty}^{\infty} K(y,x) f(x) \mathrm{d}x \tag{2.24}$$

We define the *adjoint* operator $A^* : Y \mapsto X$ (the equivalent of matrix transpose) by integration of the (complex conjugate of the) kernel function $K(x,y)$ with the other variable :

$$h = A^*\,g \quad \equiv \quad h(x) = \int_{-\infty}^{\infty} \overline{K}(y,x) g(y) \mathrm{d}y \tag{2.25}$$

and we also need to attach a meaning to the inner product :

$$\langle h, f \rangle_X := \int_{-\infty}^{\infty} \overline{h}(x) f(x) \mathrm{d}x \;, \quad \langle g, b \rangle_Y := \int_{-\infty}^{\infty} \overline{g}(y) b(y) \mathrm{d}y \tag{2.26}$$

Note that the inner product is technically different for the functions in the data space and the solution space, because they might in general have different variables (in this case they are the same : $x$ and $y$).

The SVD of operator $A$ is then defined as the set of *singular functions* $\{v_k(x), u_k(y)\}$ and singular values $\{w_k\}$ such that

$$A v_k = w_k u_k \,; \quad A^* u_k = w_k v_k$$

and with

$$\langle u_j, u_k \rangle_Y = \delta_{jk} \,, \quad \langle v_j, v_k \rangle_X = \delta_{jk} \,.$$

One major difference is that the set of such functions is (usually) *infinite*. There is often no equivalence to the kernel (Null space) of the operator, although sometimes there is. Note also the mappings

$$\begin{aligned} A^*A : X \mapsto X \qquad & \text{given by} \\ h = A^*Af \qquad & \equiv \qquad h(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \overline{K}(y,x) K(y,x') f(x') \mathrm{d}x' \mathrm{d}y \\ & \equiv \qquad h(x) = \int_{-\infty}^{\infty} B(x,x') \mathrm{d}x' \\ & \text{where} \quad B(x,x') = \int_{-\infty}^{\infty} \overline{K}(y,x) K(y,x') \mathrm{d}y \end{aligned}$$

and

$$
\begin{aligned}
AA^* : Y \mapsto Y \qquad & \text{given by} \\
b = A^*Ag \qquad \equiv \qquad & b(y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \overline{K}(y',x)K(y,x)g(y')\mathrm{d}x\mathrm{d}y \\
\equiv \qquad & b(y) = \int_{-\infty}^{\infty} C(y,y')\mathrm{d}y' \\
\text{where} \quad & C(y,y') = \int_{-\infty}^{\infty} \overline{K}(y',x)K(y,x)g(y')\mathrm{d}x
\end{aligned}
$$

**Example: convolution as a linear operator with stationary kernel.** By *stationary* we mean that the kernel depends only on the difference $(x-y)$, in other words $K(x,y)$ is a function of only one variable $K(y-x) \equiv K(t)$

$$b = A\,f := K * f \quad \equiv \quad b(y) = \int_{-\infty}^{\infty} K(y-x)f(x)\mathrm{d}x \tag{2.27}$$

An important property of convolution is the *Convolution Theorem* which relates the functions under convolution, and the result, to their Fourier Transforms. Assuming that

$$\hat{b}(k) = \mathcal{F}_{x\to k}b(x), \quad \hat{K}(k) = \mathcal{F}_{x\to k}K(x), \quad \hat{f}(k) = \mathcal{F}_{x\to k}f(x)$$

this convolution theorm states that

$$b = K * f \quad \Rightarrow \quad \hat{b} = \hat{K}\hat{f} \tag{2.28}$$

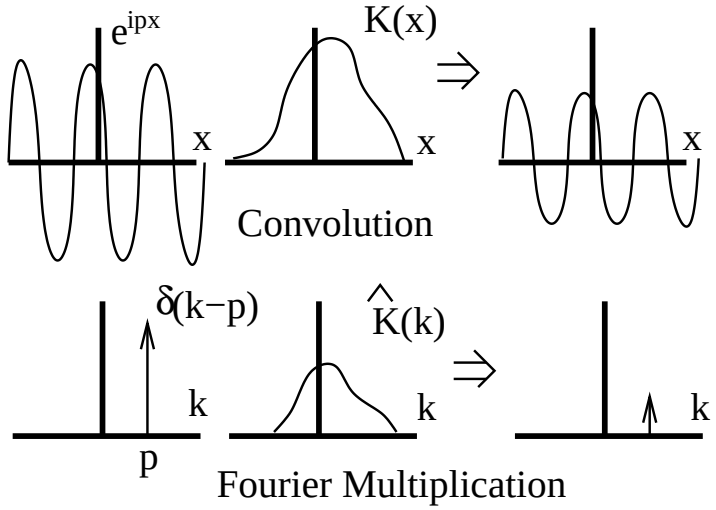



Figure 2.6: Convolution of a trigonometric function of frequency $p$ with any function $K$ results in the same frequency trigonometric function, scaled by amplitude $\hat{K}(p)$

It follows from the convolution theorm in Fourier theory that the singular vectors of $A$ are the

functions $\{\cos px, \sin px\}$ with singular values $\hat{K}(p)$ where $\hat{K}$ is the Fourier Transform of $K$

$$A \begin{pmatrix} \cos px \\ \sin px \end{pmatrix} = \hat{K}(p) \begin{pmatrix} \cos py \\ \sin py \end{pmatrix} \tag{2.29}$$

$$A^* \begin{pmatrix} \cos py \\ \sin py \end{pmatrix} = \hat{K}(p) \begin{pmatrix} \cos px \\ \sin px \end{pmatrix} \tag{2.30}$$

***Exercise* 3** *Verify that* $\sin px$ *and* $\cos px$ *are eigenvectors (and therefore also singular vectors) of the linear convolution operator with a Gaussian* $K(y-x) = \exp\left[-\frac{(y-x)^2}{2\sigma^2}\right]$ *(for any non zero value of* $\sigma$*) and that singular values are* $\lambda_p = \exp\left[-\frac{1}{2}\sigma^2 p^2\right]$.

We also see exactly how to do the inversion - applying the Moore-Penrose inverse in the continuous case is given by

$$f^{\text{recon}}(x) = \sum_{i=1}^{\infty} v_i(x) \frac{\langle b(x), u_i(x) \rangle}{w_i} \quad \equiv \quad \mathcal{F}^{-1}_{k \to x} \left[ \frac{\mathcal{F}_{x \to k} b(x)}{\mathcal{F}_{x \to k} K(x)} \right]$$

This also makes clear how the noise is propagated - the covariance of noise in the reconstruction is diagonalised by its Fourier components, and the higher spatial frequencies are amplified by the reciprocal of the square of the corresponding frequency of the convolution kernel. An example is show in figure 2.4; the convolution kernel is a Gaussian so the noise is propogated with the exponential of the square of the frequency number.

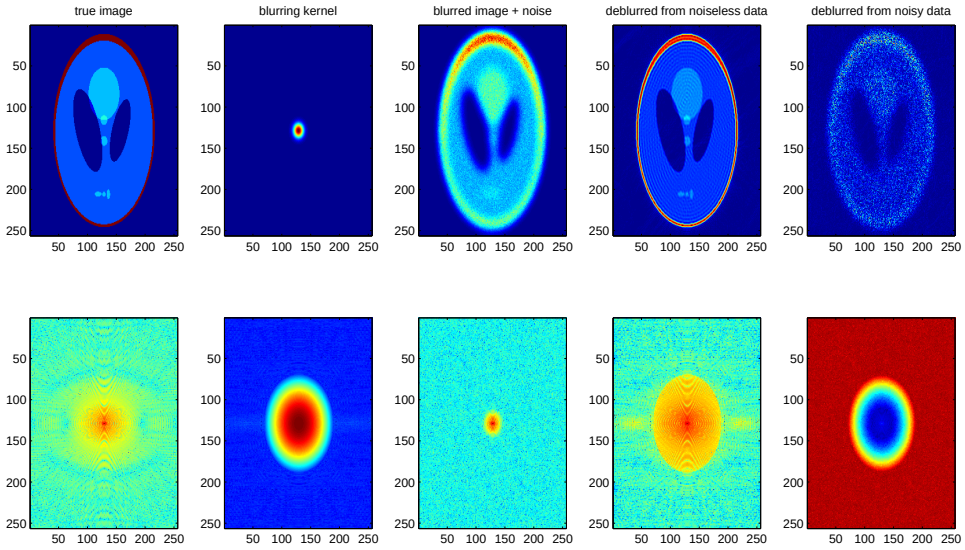



Figure 2.7: Convolution and Deconvolution in the Fourier Domain. Top row is spatial domain and bottom row is the Fourier Transform of the top row, on a logarithmic scale.

## 2.5 The Continuous-Discrete Case

Lastly, we may consider the case where the data are a finite set of samples, but the reconstruction space consists of functions. The forward mapping is

$$\boldsymbol{b} = Af \quad \equiv \quad \boldsymbol{b} = \int_{-\infty}^{\infty} \boldsymbol{K}(x) f(x) \mathrm{d}x$$

where $\boldsymbol{K}(x)$ is a vector valued function. The Adjoint becomes a sum of functions

$$h = A^* \boldsymbol{g} \quad \equiv \quad h(x) = \boldsymbol{g}^{\mathrm{T}} \boldsymbol{K}(x) = \sum_{i=1}^{N} g_i \overline{K}_i(x)$$

We also see that $A^*A : X \mapsto X$ is an operator

$$h = A^*Af \quad \equiv \quad h(x) = \int_{-\infty}^{\infty} \underbrace{\boldsymbol{K}^{\mathrm{T}}(x)\boldsymbol{K}(x')}_{B(x,x'} f(x') \mathrm{d}x' \quad = \quad \int_{-\infty}^{\infty} \underbrace{\left( \sum_{i=1}^{N} \overline{K}_i(x) K_i(x') \right)}_{B(x,x')} f(x') \mathrm{d}x'$$

and that $AA^* : Y \mapsto Y$ is a $N \times N$ matrix

$$\boldsymbol{b} = AA^* \boldsymbol{g} \quad \equiv \quad \boldsymbol{b} = \underbrace{\left( \int_{-\infty}^{\infty} \boldsymbol{K}(x) \boldsymbol{K}^{\mathrm{T}}(x) \mathrm{d}x \right)}_{\mathsf{C}} \boldsymbol{g} \quad = \quad \sum_{j=1}^{N} \underbrace{\left( \int_{-\infty}^{\infty} K_i(x) K_j^{\mathrm{T}}(x) \mathrm{d}x \right)}_{C_{ij}} g_j$$

In this case the number of left singular vectors $\{\boldsymbol{u}_i; i = 1 \ldots N\}$ is finite, and the number of right singular functions $\{v_i(x)\}$ is (usually) infinite, but those greater than $i = N$ will be in the null-space of the forward operator. This is the *usual* case for real measurements : there exists an infinite dimensional space Null-Space, "invisible" to the measurements!

**References for this section**
Chapter 2 of [6], Chapter 4 of [7], Chapter 9 of [8].

# 3 Introduction to Regularisation

## 3.1 Illustrative Example

Consider the Fredholm integral equation of the first kind representing stationary (space-invariant) convolution in one space dimension

$$g(x) = \mathcal{A}f := \int_0^1 k(x - x')f(x')\mathrm{d}x' \quad x \in [0, 1] \tag{3.1}$$

The kernel $k(x)$ is a blurring function, which is a function of only one parameter because we defined the problem to be space-invariant. In imaging systems it is called a *Point Spread Function* (PSF) [note the connection with the *Impulse Response Function* mentioned in section section B.1.3, Eq. B.3].

For an illustrative example take the kernel to be a Gaussian

$$k(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2}\right] \tag{3.2}$$

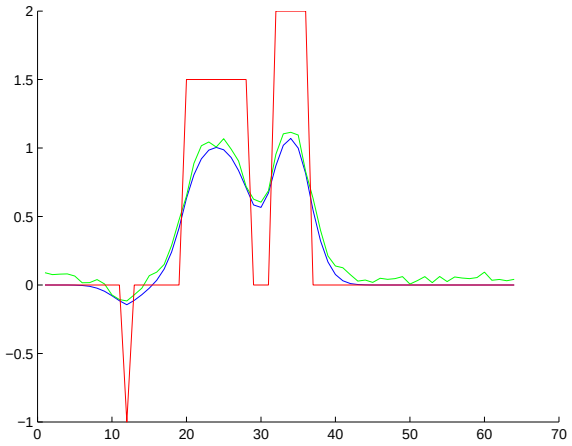



Figure 3.1: Example of blurring by Eq. 3.1 with kernel Eq. 3.2. Red curve : the original function; Blue curve the output $\mathcal{A}f$; green curve the data with added random noise.

The effect of the bluring is shown in figure 3.1. The data is corrupted with some random "measurement" noise to give data

$$\tilde{g}(x) = \mathcal{A}f + \eta \tag{3.3}$$

We want to solve the inverse problem to find $f$ given $\tilde{g}$. To introduce the ideas, we will use a

discrete setting. We construct a discrete matrix approximation $\mathsf{A}$ to $\mathcal{A}$ as

$$A_{ij} = \frac{h}{\sqrt{2\pi}\sigma} \exp\left[-\frac{((i-j)h)^2}{2\sigma^2}\right] \quad i,j \in [1 \dots n] \tag{3.4}$$

where $h = 1/n$ is the discretisation size.

The matrix $\mathsf{A}$ is rank-complete so is invertible.

$$f^\dagger = \mathsf{A}^\dagger \tilde{g} = \mathsf{A}^{-1}\tilde{g} \tag{3.5}$$

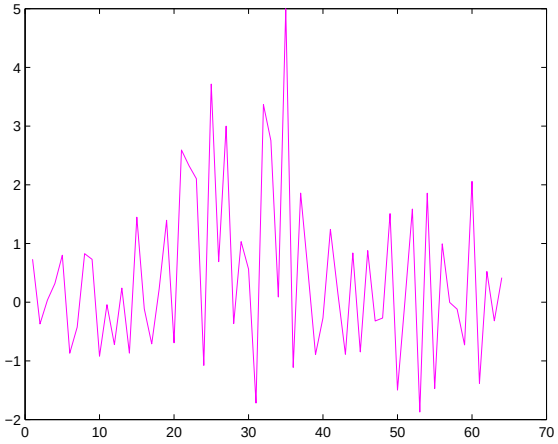



Figure 3.2: Naive reconstruction of $f$ from $\tilde{g}$ without regularisation.

Figure 3.2 shows what happens if we apply the Moore-Penrose inverse directly to the noisy data. The reason is seen from the SVD of $\mathsf{A}$. The singular spectrum decays exponentially (in fact as $\exp(-i^2)$, so the inverse adds components of the noise whose projections on the singular vectors with higher index increase exponentially.

## 3.2 Regularisation by Filtering

Our first idea for regularisation comes from the idea of *filtering* the contribution of the singular vectors in the inverse. Rather than the pseudo inverse Eq. 2.21 we construct a *regularised inverse*

$$\mathsf{A}^\dagger_\alpha = \mathsf{V}\mathsf{W}^\dagger_\alpha \mathsf{U}^{\mathrm{T}} = \sum_{i=1}^{R} \boldsymbol{v}_i \frac{q_\alpha(w_i^2)}{w_i} \boldsymbol{u}_i^{\mathrm{T}} \tag{3.6}$$

where the function $q_\alpha(w_i^2)$ is a filter on the SVD spectum. There are two commonly used filters :

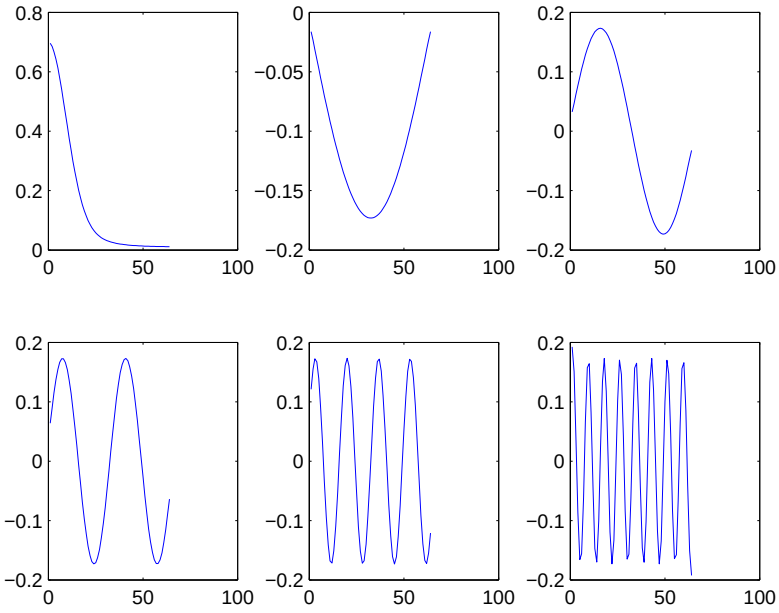


Figure 3.3: SVD of the matrix A. Shown are the singular spectrum $\{w_i\}$ and the singular vectors 1,2,4,8,16. The matrix is symmetric so the left and right singular vectors are identical.

**Truncated SVD :** here the filter is

$$q_\alpha(w_i^2) = \begin{cases} 1, & \text{if } i < \alpha \\ 0, & \text{if } i \geq \alpha \end{cases} \tag{3.7}$$

The parameter $\alpha$ acts as a threshold, cutting off the contribution of higher order singular vectors. This is exactly analogous to the idea of a *low-pass filter* in signal processing, and the analysis of SVD filtering sometimes goes by the name of *Generalised Fourier Analysis*. The regularised inverse in this case is

$$\mathsf{A}_\alpha^\dagger = \sum_{i=1}^{\alpha} \frac{\boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}}}{w_i} \tag{3.8}$$

Rather than supply $\alpha$ as an index on the singular values (which requires them to be ordered in monotonically decreasing sequence), we could specify the threshold on the value of the singular component

$$q_\alpha(w_i^2) = \begin{cases} 1, & \text{if } w_i^2 > \alpha \\ 0, & \text{if } w_i^2 \leq \alpha \end{cases} \tag{3.9}$$

**Tikhonov filtering :** here the filter is

$$q_\alpha(w_i^2) = \frac{w_i^2}{w_i^2 + \alpha} \tag{3.10}$$

The regularised inverse in this case is

$$\mathsf{A}_\alpha^\dagger = \left(\mathsf{A}^{\mathrm{T}}\mathsf{A} + \alpha\mathsf{I}\right)^{-1}\mathsf{A}^{\mathrm{T}} \tag{3.11}$$

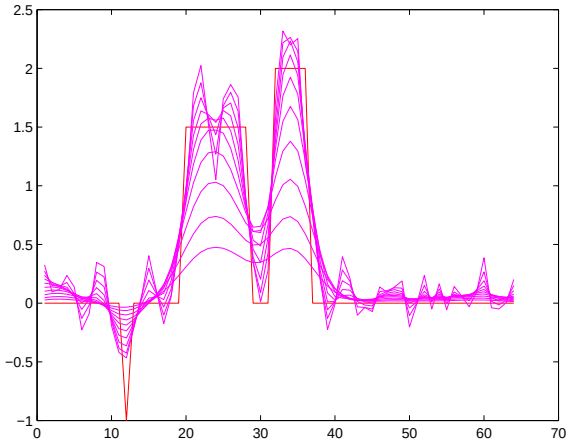



Figure 3.4: Reconstruction of $f$ from $\tilde{g}$ with Tikhonov regularistion. Curves show the effect of decreasing $\alpha$ from 1 in steps of $\frac{1}{2}$

Figure 3.4 shows the effect of Tikhonov regularisation with decreasing parameter $\alpha$. The higher values give a strong smoothing effect; the smaller values recover more of the function, at the expense of increasing noise.

***Exercise* 4** *The deblurring problem of section 3.1 is very similar but not identical to smoothing with a low-pass filter. Why is it different ? Repeat the results of this section using a low-pass filter implemented in the Fourier domain, and comment on the differences.*

## 3.3 Variational Regularisation Methods

When the system under consideration is large it is usually impractical to construct the SVD explicitly. However we can specify the solution by minimisation of a variational form. The solution

$$\boldsymbol{f}_\alpha^\dagger = \mathsf{A}_\alpha^\dagger\tilde{\boldsymbol{g}} = \left(\mathsf{A}^{\mathrm{T}}\mathsf{A} + \alpha\mathsf{I}\right)^{-1}\mathsf{A}^{\mathrm{T}}\tilde{\boldsymbol{g}} \tag{3.12}$$

is also given by the following *unconstrained* optimisation problem

$$\boldsymbol{f}_\alpha^\dagger = \arg\min_{\boldsymbol{f}\in\mathbb{R}^n}\left[\Phi = ||\tilde{g} - \mathsf{A}\boldsymbol{f}||^2 + \alpha||\boldsymbol{f}||^2\right] \tag{3.13}$$

More generally we could consider an optimisation problem

$$\boldsymbol{f}_\alpha^\dagger = \arg\min_{\boldsymbol{f}\in\mathbb{R}^n}\left[\Phi = \mathcal{D}(\tilde{g}, \mathsf{A}\boldsymbol{f}) + \alpha\Psi(f)\right] \tag{3.14}$$

where $\mathcal{D}$ is a data fitting functions (c.f.Eq. 2.6) and $\Psi$ is a regularisation functional.

### 3.3.1 Generalised Tikhonov Regularisation

The particular case

$$\Psi(\boldsymbol{f}) = \frac{1}{2}||\boldsymbol{f}||^2 \tag{3.15}$$

is known as *zero-order Tikhonov regularisation* (the factor $\frac{1}{2}$.is for convenience and can be absorbed into $\alpha$. The "zero-order" refers to the highest order of derivative in the functional, i.e. zero. Tikhonov refers in general to any *quadratic* functional

$$\Psi(\boldsymbol{f}) = \frac{1}{2}||\boldsymbol{f}||^2_{\mathsf{\Gamma}} = \frac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{\Gamma}\boldsymbol{f} \tag{3.16}$$

where $\mathsf{\Gamma}$ is a matrix. There are two ways that this is commonly used. The first is to induce *correlation* in the elements of the solution. This is clear from the Bayesian interpretation of the regularisation functional as the negative log of a prior probability density on the solution:

$$\Psi(\boldsymbol{f}) = -\log \pi(\boldsymbol{f}) \quad \Leftrightarrow \quad \pi(\boldsymbol{f}) = \exp - \left[\frac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{\Gamma}\boldsymbol{f}\right] \tag{3.17}$$

where the expression on the right is a multivariate Gaussian distribution with covariance $\mathsf{C} = \mathsf{\Gamma}^{-1}$.

Secondly, consider that the functional is specified on the *derivative* of $\boldsymbol{f}$.

$$\Psi(\boldsymbol{f}) = \frac{1}{2}||\boldsymbol{f}'||^2 \tag{3.18}$$

In the discrete setting, the derivative can be implemented as a finite difference matrix giving

$$\Psi(\boldsymbol{f}) = \frac{1}{2}||\mathsf{D}\boldsymbol{f}||^2 = \frac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{D}^{\mathrm{T}}\mathsf{D}\boldsymbol{f} \tag{3.19}$$

From the regularisation point of view we can argue that Eq. 3.18, is preferable to Eq. 3.15 because it imposes a penalty on the oscillations in the solution directly, rather than just a penalty on the magnitude of the solution. It is also clear that Eq. 3.19 is of the same form as Eq. 3.16 with $\mathsf{\Gamma} = \mathsf{D}^{\mathrm{T}}\mathsf{D}$. However, it is not conceptually straightforward to associate a covariance matrix $\mathsf{C} = \left(\mathsf{D}^{\mathrm{T}}\mathsf{D}\right)^{-1}$. This is because operator $\mathsf{D}$ is rank-deficient : it has as a null space any constant function. From the Bayesian point of view this means that the prior is such that any solutions differing by a constant are equally probable. In fact this is desirable feature. It means that we are not biasing the solution to have a proscribed mean. Remember that we are not optimising $\Psi(\boldsymbol{f})$ on its own, but in conjunction with data. It is the data that will dictate the mean of the solution.

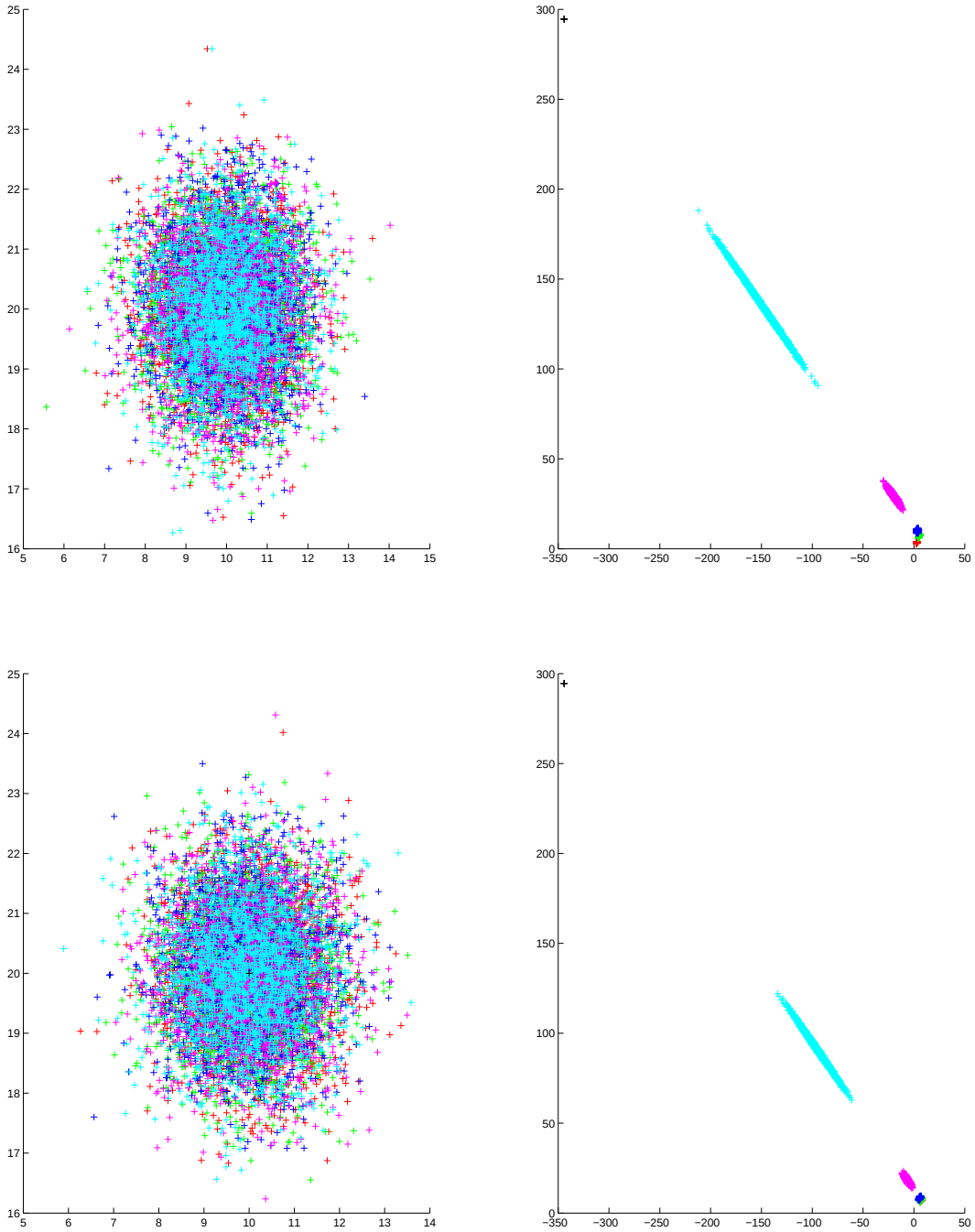



Figure 3.5: Regularised inversion of the case in figure 2.2.3; top row with prior covariance $\mathsf{C} = \mathsf{I}$, bottom row with $\mathsf{C} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$. Regularisation parameter $\alpha = [5, 0.5, 0.05, 0.005, 0.0005]$

### 3.3.2 Total Variation

One of the disadvantages of using a Tikhonov regularisation functional is that the quadratic growth of the penalty term can be too fast, and lead to over damping of solutions that are actually desirable. For example if the true function $\boldsymbol{f}$ has a step edge

$$\boldsymbol{f} = \begin{cases} 1 & 0.4 \leq \boldsymbol{x} \leq 0.6 \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

its derivative consists of a $\delta$-function at $x = 0.4$ and a negative $\delta$-functions at $x = 0.6$. Taking the square norm of a $\delta$-function leads to infinity in the penalty term, making such a solution impossible to acheive. In fact one can see that the form Eq. 3.18 will force the edges to be less and less steep; a desirable property to eliminate noise, but undesirable if real edges are present. If instead we use a 1-norm

$$\Psi(\boldsymbol{f}) = |\boldsymbol{f}'| \tag{3.21}$$

Then the value will be $\Psi = 2$ for the case considered in Eq. 3.20.

The so-called *Total Variation* (TV) functional is just

$$\Psi(\boldsymbol{f}) = |\mathsf{D}\boldsymbol{f}| = \sum_{i=1}^{n-1} |f_{i+1} - f_i| \tag{3.22}$$

and will be considered in more detail for images in section 5. The use of TV as a regulariser can lead to reconstructions with sharp edges that still supress noise. However, there is no longer a direct matrix inversion like Eq. 3.12.

## 3.4 Choice of Regularisation parameter

Choice of regularisation parameter is usually considered a tradeoff between the accuracy of recovery of the correct solution, and the accuracy of fitting to the data.

A measure of reconstruction accuracy can only be determined if a true solution is actually known. Then we may define the *estimation error*

$$\boldsymbol{e}_\alpha := \boldsymbol{f}_{\text{true}} - \boldsymbol{f}_\alpha^\dagger \,. \tag{3.23}$$

Usually we only know the measured data, and the approximation to it given by the model acting on our estimated solution. We define the data residual as

$$\boldsymbol{r}_\alpha := \tilde{g} - A\boldsymbol{f}_\alpha^\dagger \,. \tag{3.24}$$

Note that the true solution $\boldsymbol{f}_{\text{true}}$ should not give a zero residual, because the data contains noise. If the true solution were available we could define the *predictive error*

$$\boldsymbol{p}_\alpha := A\boldsymbol{e}_\alpha = A\boldsymbol{f}_{\text{true}} - A\boldsymbol{f}_\alpha^\dagger \,. \tag{3.25}$$

Regularisation parameters selection methods can be divided between those where the noise statistics are known, and those where it is not. In the following we assume that the noise is identical and independent on each component of the data, i.e that it is drawn from a zero-mean Gaussian probability distribution with a multiple of identity as covariance

$$\eta \sim N(0, \sigma^2 \mathsf{I})$$

This is known as *white noise*. The case where the noise is correlated can be handled equally well by rotation of the data-space coordinates. The case where the the noise is non-Gaussian is more complex and will be considered later.

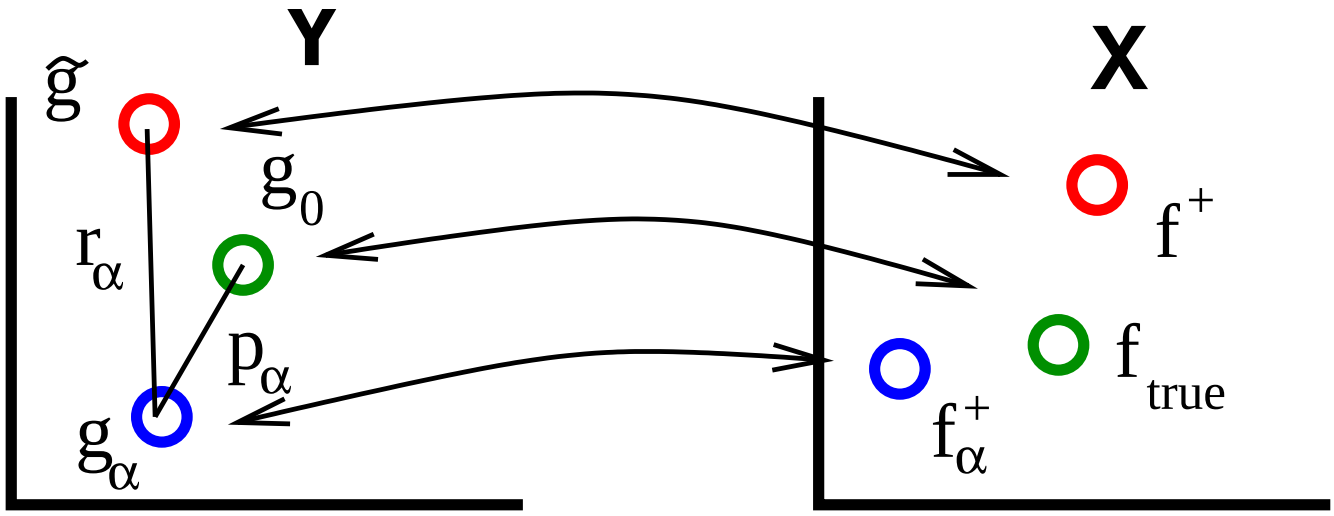



Figure 3.6: Under forward mapping the true solution $f_{\mathsf{true}}$ gives rise to noise free data $g_0$. In the presence of noise the measured data is $\tilde{g} = g_0 + \eta$. Reconstruction of $\tilde{g}$ with regularisation gives a solution $f_\alpha$ whose forward map gives modelled data $g_\alpha$. The *residual errror* is the data-fit discrepency $r_\alpha$, whereas the *predictive error* $p_\alpha$ is the difference of the modelled data to the true, noise-free data.

### 3.4.1 Discrepancy Principle (DP)

In this method we assume that the norm of the residual should be equal to the expectation of the norm of the noise

$$||\boldsymbol{r}_\alpha||^2 = \mathbb{E}(||\eta||^2) = n\sigma^2 \tag{3.26}$$

where $n$ is the length of the data vector. We therefore need to solve the nonlinear optimisation problem

$$\mathrm{DP}(\alpha) := \frac{1}{n}||\boldsymbol{r}_\alpha||^2 - \sigma^2 = 0\,. \tag{3.27}$$

In some cases we can get an expression for $DP(\alpha)$ in terms of the SVD. For example using Eq. 3.6 we can put

$$\begin{aligned} \boldsymbol{r}_\alpha &= \left(\mathsf{UWW}_\alpha^\dagger \mathsf{U}^{\mathrm{T}} - \mathsf{I}\right)\tilde{\boldsymbol{g}} \\ \Rightarrow \quad DP(\alpha) &= \frac{1}{n}\sum_{i=1}^{n}\left(q_\alpha(w_i^2)-1\right)^2(\boldsymbol{u}_i\cdot\tilde{\boldsymbol{g}})^2 - \sigma^2 \end{aligned} \tag{3.28}$$

If $0 \leq q_\alpha(w_i^2) < 1$ is montononically decreasing (for example Tikhonov regularisation Eq. 3.10) then $DP(\alpha)$ is monotonically increasing. This means that the solution to Eq. 3.27 can be found straightforwardly, e.g. by binary search.

Suppose that the noise is correlated

$$\eta \sim N(0, \mathsf{C})$$

we may write a different expression for the discrepancy principle by taking the norm of the residual with the metric $\mathsf{\Gamma} = \mathsf{C}^{-1}$ and similarly for the noise metric, a technique known as *pre-whitening*

$$||\boldsymbol{r}_\alpha||_{\mathsf{\Gamma}}^2 = \mathbb{E}(||\eta||_{\mathsf{\Gamma}}^2) = 1 \tag{3.29}$$

### 3.4.2 Miller Criterion

The Discrepancy Principle only takes account of the statistical properties of the *data*. Is there a similar procedure for consideration of the solution error ? If we knew $\boldsymbol{f}_{\text{true}}$ we could in principle require

$$||\boldsymbol{e}_\alpha||^2 - m\sigma_f^2 = 0\,. \tag{3.30}$$

In practice we do not know $\boldsymbol{f}_{\text{true}}$ but when using a regularisation functional we are implicitly assuming that the solution is statistically distributed with probablity

$$\pi(\boldsymbol{f}) = \exp(-\Psi(\boldsymbol{f}))\,.$$

We may therefore choose a parameter $\alpha$ so that the contributions to the total functional from the data and prior terms is about equal

$$\text{Miller}(\alpha) = \frac{\frac{1}{2}||\boldsymbol{r}_\alpha||_{\mathsf{\Gamma}}^2}{\sigma^2} - \Psi(\boldsymbol{f}) \to 0 \tag{3.31}$$

As for the Discrepancy Principle, it is the zero of this function that is required.

### 3.4.3 Unbiased Predictive Risk Estimation (UPRE)

The UPRE is based on a statistical estimator of the mean squared norm of the predictive error

$$\frac{1}{n}||\boldsymbol{p}_\alpha||^2 = \frac{1}{n}||A\boldsymbol{f}_\alpha^\dagger - A\boldsymbol{f}_{\text{true}}||^2\,, \tag{3.32}$$

which is called the *predictive risk*. We define the $n \times n$ *influence matrix* as

$$\mathsf{P}_\alpha = \mathsf{A}\mathsf{A}^\dagger_\alpha \tag{3.33}$$

[ Note that in terms of the SVD $\mathsf{P}_\alpha = \mathsf{U}\mathsf{W}\mathsf{W}^\dagger_\alpha\mathsf{U}^{\mathrm{T}}$ ]. Using Eq. 3.3 we write

$$\boldsymbol{p}_\alpha = (\mathsf{P}_\alpha - \mathsf{I})\,\mathsf{A}\boldsymbol{f}_{\mathrm{true}} + \mathsf{P}_\alpha\boldsymbol{\eta} \tag{3.34}$$

Since the first term on the right is deterministic it has the same value under expectation and the second term has expectation

$$\mathbb{E}\left[||\mathsf{P}_\alpha\boldsymbol{\eta}||^2\right] = \mathbb{E}\left[\boldsymbol{\eta}^{\mathrm{T}}\mathsf{P}^{\mathrm{T}}_\alpha\mathsf{P}_\alpha\boldsymbol{\eta}\right] = \sum_{i,j}\mathsf{P}^{\mathrm{T}}_\alpha\mathsf{P}_\alpha\mathbb{E}\left[\eta_i\eta_j\right] = \mathrm{Tr}[\mathsf{P}^{\mathrm{T}}_\alpha\mathsf{P}_\alpha]\sigma^2 \tag{3.35}$$

We write

$$\mathbb{E}\left[\frac{1}{n}||\boldsymbol{p}_\alpha||^2\right] = \frac{1}{n}||\,(\mathsf{P}_\alpha - \mathsf{I})\,\mathsf{A}\boldsymbol{f}_{\mathrm{true}}||^2 + \frac{\sigma^2}{n}\mathrm{Tr}[\mathsf{P}^{\mathrm{T}}_\alpha\mathsf{P}_\alpha] \tag{3.36}$$

Comparing this with the Discrepancy Principle, we can write

$$\boldsymbol{r}_\alpha = \tilde{g} - A\boldsymbol{f}^\dagger_\alpha = (\mathsf{P}_\alpha - \mathsf{I})\,\mathsf{A}\boldsymbol{f}_{\mathrm{true}} + (\mathsf{P}_\alpha - \mathsf{I})\,\boldsymbol{\eta}\,, \tag{3.37}$$

so that

$$E\left[\frac{1}{n}||\boldsymbol{r}_\alpha||^2\right] = \frac{1}{n}||\,(\mathsf{P}_\alpha - \mathsf{I})\,\mathsf{A}\boldsymbol{f}_{\mathrm{true}}||^2 + \frac{\sigma^2}{n}\mathrm{Tr}[\mathsf{P}^{\mathrm{T}}_\alpha\mathsf{P}_\alpha] - \frac{2\sigma^2}{n}\mathrm{Tr}[\mathsf{P}_\alpha] + \sigma^2\,, \tag{3.38}$$

and we write

$$\mathbb{E}\left[\frac{1}{n}||\boldsymbol{p}_\alpha||^2\right] = \mathbb{E}\left[\frac{1}{n}||\boldsymbol{r}_\alpha||^2\right] + \frac{2\sigma^2}{n}\mathrm{Tr}[\mathsf{P}_\alpha] - \sigma^2\,. \tag{3.39}$$

The UPRE is the random function whos expectation is given by Eq. 3.39 :

$$\mathrm{UPRE}(\alpha) = \frac{1}{n}||\boldsymbol{r}_\alpha||^2 + \frac{2\sigma^2}{n}\mathrm{Tr}[\mathsf{P}_\alpha] - \sigma^2\,. \tag{3.40}$$

The quantity required is the value of $\alpha$ that minimises Eq. 3.40.

### 3.4.4 Generalised Cross Validation (GCV)

The method of Generalised Cross Validation (GCV) is an alternative to UPRE that does not require prior knowledge of the variance $\sigma^2$ of the noise. It is given by minimising

$$\mathrm{GCV}(\alpha) = \frac{\frac{1}{n}||\boldsymbol{r}_\alpha||^2}{\left(\frac{1}{n}\mathrm{Tr}[\mathsf{I} - \mathsf{P}_\alpha]\right)^2} \tag{3.41}$$

### 3.4.5 L-curve

The L-curve is a graphical interpretation of the Miller method. To implement it, we plot a graph of the points $\left\{||\boldsymbol{r}_\alpha||^2, \Psi(\boldsymbol{f}_\alpha^\dagger)\right\}$ for different values of $\alpha$ on a log-log plot. The graph usually has a sharply defined "corner" like that of the letter "L" (hence the name). It is the value of $\alpha$ corresponding to this corner that is chosen as the regularisation parameter value. Sometimes we fit a spline curve through a set of $\left\{||\boldsymbol{r}_\alpha||^2, \Psi(\boldsymbol{f}_\alpha^\dagger)\right\}$ points, and derive the required $\alpha$ as the point of maximum curvature of this fitted curve.

**References for this section :** [8, 5, 9].

# 4 Iterative Methods

When the dimension of the problem becomes large the SVD is an impractical tool, and direct inversion of $A$ is often impossible. Even storage of a matrix representation of $A$ becomes difficult if the dimensions is very large. This leads to the requirement for iterative methods. We first consider linear problems.

## 4.1 Gradient Descent and Landweber Method

Consider the minimisation of a quadratic form

$$\frac{1}{2}||\mathsf{A}\boldsymbol{f}-\boldsymbol{g}||^2 \to \min \quad \equiv \quad \Phi(\boldsymbol{f},\boldsymbol{g}) = \frac{1}{2}\left\langle \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{f},\boldsymbol{f}\right\rangle - \left\langle \mathsf{A}^{\mathrm{T}}\boldsymbol{g},\boldsymbol{f}\right\rangle \to \min \tag{4.1}$$

The gradient is

$$\nabla_{\boldsymbol{f}}\Phi(\boldsymbol{f},\boldsymbol{g}) = \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{f} - \mathsf{A}^{\mathrm{T}}\boldsymbol{g} \tag{4.2}$$

which coincides with the negative residual associated with the least-squares criterion

$$\boldsymbol{r} = \mathsf{A}^{\mathrm{T}}(\boldsymbol{g}-\mathsf{A}\boldsymbol{f}) \tag{4.3}$$

Given an approximation $\boldsymbol{f}_k$ for the solutions, in a neighbourhood of the point the functional $\Phi$ decreases most rapidly in the direction of the *negative* gradient, i.e. in the direction of the residual $\boldsymbol{r}$. Thus for small values $\tau$, we can guarantee that a new solution

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \tau\boldsymbol{r}_k \tag{4.4}$$

will give a smaller value of $\Phi$. How do we choose $\tau$ ? It makes sense to minimise the one dimensional function

$$\phi(\tau) = \Phi(\boldsymbol{f}_k+\tau\boldsymbol{r}_k,\boldsymbol{g}) = \Phi(\boldsymbol{f}_k,\boldsymbol{g}) + \frac{\tau^2}{2}||\mathsf{A}\boldsymbol{r}_k||^2 - \tau||\boldsymbol{r}_k||^2\,. \tag{4.5}$$

Since this is a quadratic function of $\tau$ we have immediately that

$$\tau_k = \frac{||\boldsymbol{r}_k||^2}{||\mathsf{A}\boldsymbol{r}_k||^2}\,, \tag{4.6}$$

which gives rise to the *steepest descent* update rule. [1] We can see immediately that the residuals have the properties

- They can be obtained iteratively

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \tau_k\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{r}_k\,, \tag{4.7}$$

[1] More generally the minimisation of the one-dimensional functional $\phi(\tau)$ would be non-quadratic and would require a *line-search* method.

- They satisfy an orthogonality condition

$$\langle \boldsymbol{r}_{k+1}, \boldsymbol{r}_k \rangle = 0\,. \tag{4.8}$$

Since $\mathsf{A}^{\mathrm{T}}\mathsf{A}$ is positive-definite the method has a simple geometric interpretation. Starting at $f_0$

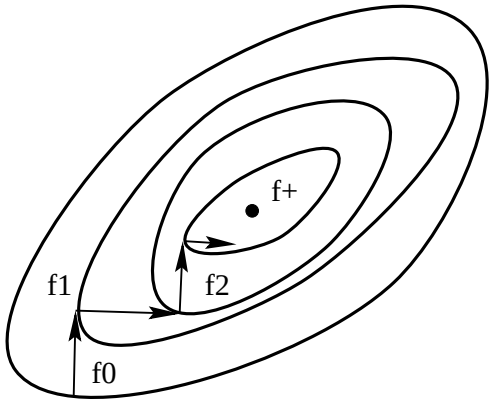



Figure 4.1: The steepest descent method.

one moves perpendicular to the level set of $\Phi(f_0, g)$ for a distance so that at the next point $f_1$ the line $f_0 - f_1$ is tangent to $\Phi(f_1, g)$. Then one moves perpendicular to the level set of $\Phi(f_1, g)$ for a distance so that at the next point $f_2$ the line $f_1 - f_2$ is tangent to $\Phi(f_2, g)$.

The steepest-descent method is a nonlinear method for solving least-squares problems. It works equally well for any optimisation problem sufficiently close to a local minimum. Stopping the iterations before minimisation has a regularising effect. However it can be unreasonably slow. This is clear from figure 4.1, where in the case of an anisotropic optimisation surface, the iterations can only descend very gradually to the minimum. Therefore it is almost always better to use a conjugate gradient method.

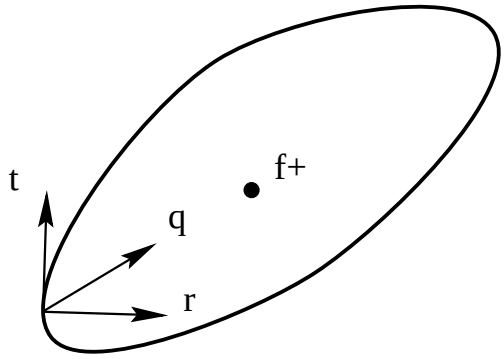



Figure 4.2: For a level surface $\Phi$ = constant, $\boldsymbol{t}$ is a tangent vector, $\boldsymbol{r}$ is the normal vector, and $\boldsymbol{q}$ is the conjugate direction.

## 4.2 Conjugate Gradients

Two vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are said to be conjugate with respect to a matrix $\mathsf{B}$ if

$$\langle \boldsymbol{p}, \mathsf{B}\boldsymbol{q} \rangle = 0$$

Then, given any level surface (ellipsoid) of the discrepency functional and a point on this surface, the direction of the conjugate gradient at this point is the direction orthogonal to the tangent plane w.r.t the matrix $\mathsf{A}^{\mathrm{T}}\mathsf{A}$.

We observe that a descent method such as described in section 4.1 builds the solution at the $j^{\text{th}}$ iteration, from a vector space spanned by the set of vectors

$$\mathcal{K}^{(j)}(\mathsf{A}, \boldsymbol{g}) = \left\{ \mathsf{A}^{\mathrm{T}}\boldsymbol{g}, \mathsf{A}^{\mathrm{T}}\mathsf{A}\mathsf{A}^{\mathrm{T}}\boldsymbol{g}, \ldots \left(\mathsf{A}^{\mathrm{T}}\mathsf{A}\right)^{(j)} \mathsf{A}^{\mathrm{T}}\boldsymbol{g} \right\} \tag{4.9}$$

where it is assumed that all of the vectors in the set are mutually independent. This space is called the *Krylov subspace* of $\mathsf{A}$. An optimal solution may be found by choosing at each iteration a direction $\boldsymbol{q}$ that is the least-squares solution in this Krylov subspace. At the first iteration the subspace is of dimension one, and our only choice is the same as steepest descents. At subsequent directions we choose a direction that points towards the centre of an ellipsoid given by the intersection of the functional level surface with the Krylov subspace.

If $\mathsf{P}_k$ is the projection operator onto $\mathcal{K}^{(j)}(\mathsf{A}, \boldsymbol{g})$ then we need to solve

$$\begin{aligned} ||\mathsf{A}\mathsf{P}_k\boldsymbol{f} - \boldsymbol{g}||^2 \quad &\rightarrow \quad \min \\ \text{subject to} \quad & \quad \mathsf{P}_k\boldsymbol{f} = \boldsymbol{f} \end{aligned} \tag{4.10}$$

which is given by

$$\mathsf{P}_k\mathsf{A}^{\mathrm{T}}\mathsf{A}\mathsf{P}_k\boldsymbol{f} = \mathsf{P}_k\mathsf{A}^{\mathrm{T}}\boldsymbol{g} \tag{4.11}$$

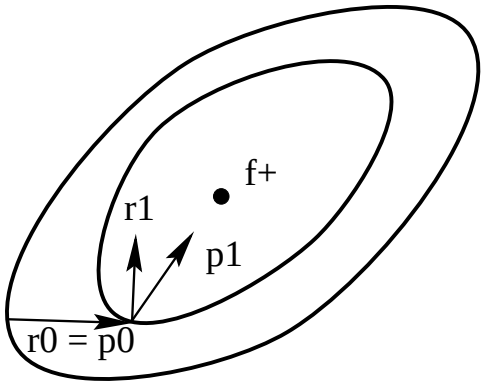



Figure 4.3: Two dimensional representation of the conjugate gradient method.

The CG method is based on the construction of two bases $\{\boldsymbol{r}_k\}$ and $\{\boldsymbol{p}_k\}$ as follows

- $$\boldsymbol{r}_0 = \boldsymbol{p}_0 = \mathsf{A}^{\mathrm{T}}\boldsymbol{g} \tag{4.12}$$

- compute
$$\alpha_k = \frac{||\boldsymbol{r}_k||^2}{\langle \boldsymbol{r}_k, \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k\rangle} \tag{4.13}$$

- compute
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k \tag{4.14}$$

- compute
$$\beta_k = \frac{\left\langle \boldsymbol{r}_{k+1}, \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k\right\rangle}{\langle \boldsymbol{p}_k, \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k\rangle} \tag{4.15}$$

- compute
$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k \tag{4.16}$$

Then the iterative scheme for the solutions is

$$\begin{aligned} \boldsymbol{f}_0 &= 0 \\ \boldsymbol{f}_{k+1} &= \boldsymbol{f}_k + \alpha_k \boldsymbol{p}_k \end{aligned} \tag{4.17}$$

The coefficients $\alpha_k$, $\beta_k$ are chosen so that the following orthogonality conditions hold

$$\langle \boldsymbol{r}_{k+1}, \boldsymbol{r}_k\rangle = 0\,, \quad \left\langle \boldsymbol{p}_{k+1}, \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k\right\rangle = 0\,,$$

i.e. the sequence of $\{\boldsymbol{r}_k\}$ are such that $\boldsymbol{r}_1 \perp \boldsymbol{r}_0$, $\boldsymbol{r}_2 \perp \boldsymbol{r}_1$, etc. and $\boldsymbol{p}_1$ is conjugate to $\boldsymbol{p}_0$, $\boldsymbol{p}_2$ is conjugate to $\boldsymbol{p}_1$ etc. Furthermore we can show that

- the set $\{\boldsymbol{r}_k\}$ form an orthogonal basis of $\mathcal{K}^{(j)}(\mathsf{A}, \boldsymbol{g})$,
- the set $\{\boldsymbol{p}_k\}$ form an $\mathsf{A}^{\mathrm{T}}\mathsf{A}$-orthogonal basis of $\mathcal{K}^{(j)}(\mathsf{A}, \boldsymbol{g})$,
- the set $\{\boldsymbol{r}_k\}$ are the residuals
$$\boldsymbol{r}_k = \mathsf{A}^{\mathrm{T}}\boldsymbol{g} - \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{f}_k$$
- the following alternative expression for $\alpha_k$, $\beta_k$ hold :
$$\alpha_k = \frac{||\boldsymbol{r}_k||^2}{\langle \boldsymbol{p}_k, \mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{p}_k\rangle} \quad \beta_k = \frac{||\boldsymbol{r}_{k+1}||^2}{||\boldsymbol{r}_k||^2}$$
which implies that $\alpha_k$, $\beta_k$ are positive.

Starting at $f_0$ one moves perpendicular to the level set of $\Phi(f_0, g)$ for a distance so that at the next point $f_1$ the line $f_0 - f_1$ is tangent to $\Phi(f_1, g)$. Then one moves to the centre of the ellipse given by the intersection of the level surface $\Phi(f_1, g) =$ constant with the plane spanned by $\boldsymbol{r}_0$ and $\boldsymbol{r}_1$. The direction of movement is $\boldsymbol{p}_1$. Then we move to centre of the ellipse given by the

intersection of the level surface $\Phi(f_2, g) =$ constant with the three-dimensional space spanned by $\boldsymbol{r}_0$, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$. The direction of movement is $\boldsymbol{p}_2$. And so on. If the problem is $n-$dimensional we reach the minimum in $n$ steps.

**References for this section :** [8, 5, 7].

# 5 Image Regularisation functionals

We here introduce a functional used frequently in image reconstruction and image processing.

$$\Psi(f) = \int_\Omega \psi\left(|\nabla f|\right) \mathrm{d}\boldsymbol{x} \tag{5.1}$$

We want to derive the Gateaux derivative for Eq. 5.1 in direction $h$, defined to be

$$\Psi'(f)h := \lim_{\epsilon \to 0} \left[ \frac{\Psi(f + \epsilon h) - \Psi(f)}{\epsilon} \right] \tag{5.2}$$

for arbitrary function $h$ satisfying the same boundary conditions as $f$. $\Psi'(f)$ is a linear operator mapping from the space of functions to a scalar value. If we can show that it does not depend on $h$ then $\Psi'(f)$ is the Fréchet derivative of $\Psi$ and constitutes a generalised notion of a gradient. See appendix section B.1 for more details.

Expanding Eq. 5.1 we have

$$\Psi(f + \epsilon h) = \int_\Omega \psi\left(\sqrt{\nabla f \cdot \nabla f + 2\epsilon \nabla f \cdot \nabla h + \epsilon^2 \nabla h \cdot \nabla h}\right) \mathrm{d}\boldsymbol{x}$$

Taking the Taylor series we have

$$\Psi(f + \epsilon h) = \Psi(f) + \int_\Omega 2\epsilon \psi'\left(|\nabla f|\right) \frac{1}{2} |\nabla f|^{-1} \nabla f \cdot \nabla h \mathrm{d}\boldsymbol{x} + O(\epsilon^2)$$

Making the definition

$$\kappa := \frac{\psi'\left(|\nabla f|\right)}{|\nabla f|}$$

we have

$$\Psi'(f)h = \int_\Omega \kappa \nabla f \cdot \nabla h \mathrm{d}\boldsymbol{x} = -\int_\Omega h \nabla \cdot \kappa \nabla f \mathrm{d}\boldsymbol{x}$$

where the last expression results from using the divergence theorem. We now define the differential operator

$$\mathcal{L}(f) = -\nabla \cdot \kappa \nabla$$

so that we can state

$$\Psi'(f)h = \int_\Omega h \mathcal{L}(f) f \mathrm{d}x = \langle h, \mathcal{L}(f) f \rangle = \langle h, \delta\Psi \rangle$$

Note that

- $\mathcal{L}(f)$ is a linear operator, but it is parameterised on $f$ because of the function $\kappa$.
- $\Psi'(f)$ does not depend on $h$. It is a linear integral operator with kernel function $\delta\Psi$.
- $\delta\Psi := \mathcal{L}(f) f$ is an image.

**Examples**

1. *First order Tikhonov.* If $\psi(s) = \frac{1}{2}s^2 \Leftrightarrow \psi'(s) = s$ then $\kappa = 1$

$$\mathcal{L}(f) = -\nabla^2$$

   which is the stationary Laplacian (does not depend on $f$).
2. *Total Variation.* If $\psi(s) = s \Leftrightarrow \psi'(s) = 1$, then $\kappa = \frac{1}{|\nabla f|}$

$$\mathcal{L}(f) = -\nabla \cdot \frac{1}{|\nabla f|}\nabla$$

3. Perona-Malik If $\psi(s) = T \log\left(1 + \frac{s}{T}\right) \Leftrightarrow \psi'(s) = \frac{sT}{T+s}$ then $\kappa = \frac{T}{T+|\nabla f|}$

$$\mathcal{L}(f) = -\nabla \cdot \frac{1}{1 + \frac{|\nabla f|}{T}}\nabla$$

   where $T$ is a threshold

Because of the division by zero of $\kappa = \frac{1}{|\nabla f|}$ total variation is implemented by some approximation such as $\psi(f) = \sqrt{|\nabla f|^2 + \beta^2}$.

Repeated application of the functional gradient generates the so-called *scale-space* of an image

$$f^{n+1} = f^n + \Psi'(f^n) \tag{5.3}$$

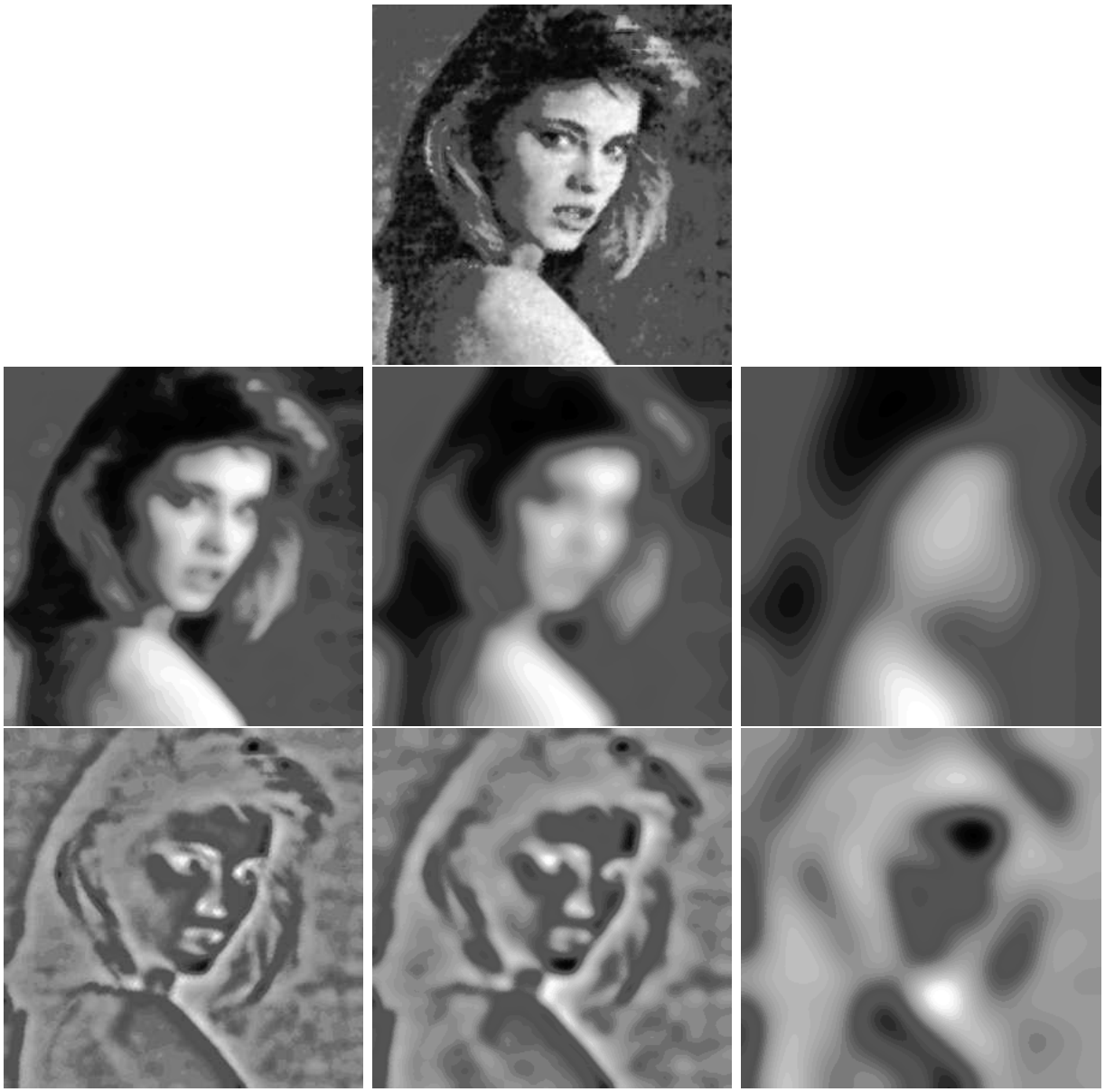


Figure 5.1: Isotropic scale resulting from the choice of a first order tikhonov prior. Iterations 1, 3,10 corresponding to $\Delta t = 12, 15, 40$. Top row : original image; second row : image scale space; third row : gradient of prior $\Psi'$.

Figure 5.2: Anisotropic scale resulting from the choice of a Perona-Malik prior with threshold parameter $T = 1.5\%$ of the image maximum. Iterations 10, 25,35 corresponding to $\Delta t = 40, 329, 1332$. Top row : original image; second row : image scale space; third row : diffusivity; fourth row : gradient of prior $\Psi'$.

# 6 Nonlinear Optimisation

We first consider a classical example of a function that is well-posed, but strongly non-linear. The Rosenbrock function

$$\Phi(\boldsymbol{f}) = 100(f_2 - f_1^2)^2 + (1 - f_1)^2 \tag{6.1}$$

for which the contours are shown:

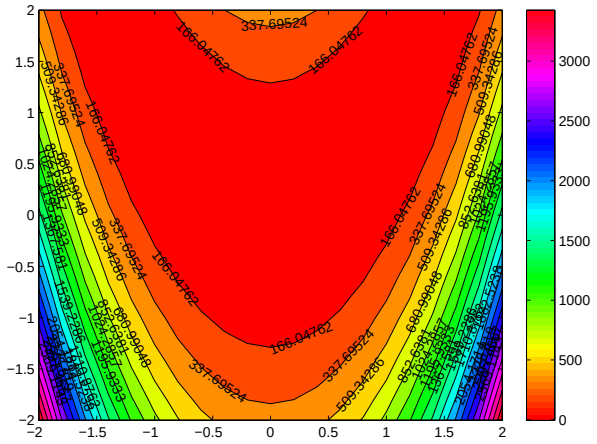



Figure 6.1: Rosenbrock function

## 6.1 Descent Methods

Both Steepest descents and Conjugate gradient have a non-linear equivalent. The key difference in each case is that i) the step length needs to be determined by a *line-search* step, and ii) a termination of *stopping* criterion needs to be determined.

We write the problem to be minimised simply as

$$\Phi(\boldsymbol{f}) \rightarrow \min \tag{6.2}$$

### 6.1.1 Non-linear Steepest Descents

As an example, consider *non-linear least squares*

$$\Phi(\boldsymbol{f}) := \frac{1}{2}||\boldsymbol{g} - A(\boldsymbol{f})||^2 \tag{6.3}$$

where $A(\boldsymbol{f})$ is a non-linear operator.

The algorithm is

**Algorithm 2** Non-linear Steepest Descent

$k = 0$, $\boldsymbol{f}_0 =$ initial guess
**while** stopping criterion unfulfilled **do**
  $\boldsymbol{p}_k = -\Phi'(\boldsymbol{f}_k)$
  $\tau_k = \arg\min_{\tau>0}\Phi(\boldsymbol{f}_k + \tau\boldsymbol{p}_k)$
  $\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \tau_k\boldsymbol{p}_k$
  $k = k + 1$
**end while**

For non-linear least squares as in Eq. 6.3 the descent direction is given by

$$-\Phi'(\boldsymbol{f}) = A'^*(\boldsymbol{f})[\boldsymbol{g} - A(\boldsymbol{f})] \tag{6.4}$$

where $A'^*$ is the *adjoint Fréchet derivative* of operator $A$.

Some example stopping criteria are

- The error functional reduces by a sufficient amount in total

$$\Phi(\boldsymbol{f}_k) \leq \epsilon\Phi(\boldsymbol{f}_0)$$

- The relative change in the functional at the last iteration is smaller than some threshold

$$\frac{\Phi(\boldsymbol{f}_{k-1}) - \Phi(\boldsymbol{f}_k)}{\Phi(\boldsymbol{f}_{k-1})} \leq \epsilon$$

- The gradient of the functional falls below some threshold

$$||\Phi'(\boldsymbol{f}_k)|| \leq \epsilon$$

The linesearch problem

$$\tau_k = \arg\min_{\tau>0}\Phi(\boldsymbol{f}_k + \tau\boldsymbol{p}_k) \tag{6.5}$$

is a one-dimensional optimistion problem; the direction $\boldsymbol{p}_k$ is called the *search direction* and the parameter $\tau$ is called the *step length*. The search direction is a *descent direction* if

$$\langle\Phi'(\boldsymbol{f}_k), \boldsymbol{p}_k\rangle < 0\,. \tag{6.6}$$

Obviously, the negative gradient direction is a descent direction because

$$\langle\Phi'(\boldsymbol{f}_k), -\Phi'(\boldsymbol{f}_k)\rangle \;=\; -||\Phi'(\boldsymbol{f}_k)||^2 < 0\,. \tag{6.7}$$

An important result of which we will make use later is that *any symmetric positive definite operator $B$ applied to the negative gradient direction is also a descent direction.* This follows directly from the definition of a symmetric positive definite operator :

$$\tilde{\boldsymbol{p}}_k = -B\Phi'(\boldsymbol{f}_k) \quad \Rightarrow \langle\Phi'(\boldsymbol{f}_k), \tilde{\boldsymbol{p}}_k\rangle = \langle\Phi'(\boldsymbol{f}_k), -B\Phi'(\boldsymbol{f}_k)\rangle = -||\Phi'(\boldsymbol{f}_k)||_B^2 < 0\,. \tag{6.8}$$

If the problem were quadratic $\Phi(\boldsymbol{f}) = ||\mathsf{A}\boldsymbol{f} - \boldsymbol{g}||^2$, then the linesearch would also be quadratic and would be minimised by Eq. 4.6. Instead we have to estimate the minimum. In practice it is not necessary to find the exact minimum, because the expense of doing so might not be as much as the recomputation of the gradient. If $\boldsymbol{p}_k$ is a descent direction it might seem OK to take any positive step length, but there are some conditions under which either too small or too large a step could converge to a limit that was not a local minimum. Therefore the step is required to satisfythe *Wolfe conditions* :

| | | |
|---|---|---|
| Wolfe condition 1 (sufficient decrease): | $\phi(\tau) \leq \phi(0) + c_1\tau\phi'(0)\,,$ | $\tau > 0$ |
| Wolfe condition 2 (curvature condition): | $\phi'(\tau) \geq c_2\tau\phi'(0)\,,$ | $\tau > 0$ |
| | | $0 < c_1 < c_2 < 1$ |

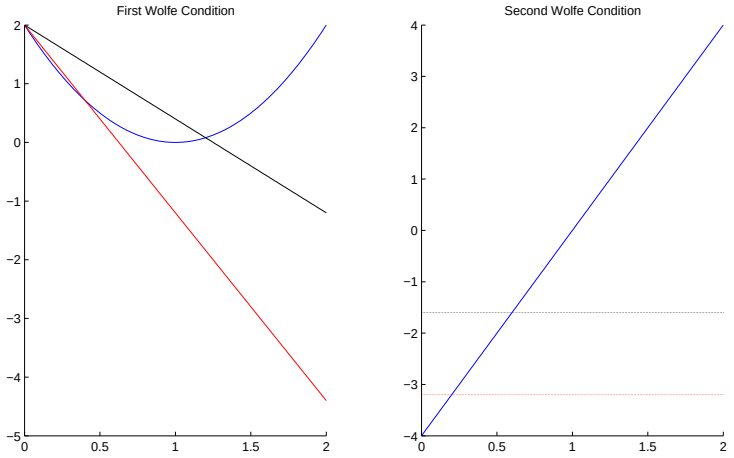



Figure 6.2: Illustration of Wolfe Conditions for simple quadratic problem.

A well-used and convenient technique for finding an approximate solution is a bracketing and quadratic interpolation technique. The aim of the bracketing part is to find three possible steps, with the middle one lower than the other two. i.e.

$$\text{find } \{\tau_a < \tau_b < \tau_c | \phi(\tau_a) > \phi(\tau_b) \& \phi(\tau_c) > \phi(\tau_b)\} \tag{6.9}$$

A plausible set can be found by a number of methods. For example if we begin with three search steps that are monotonically decreasing $\{\tau_1 < \tau_2 < \tau_3 | \phi(\tau_1) > \phi(\tau_2) > \phi(\tau_3)\}$, we can simply "reflect" the first point in the last : $\tau_n = 2\tau_{n-1} - \tau_{n-3}$, until Eq. 6.9 is satisfied.

The interpolation part simply consists of fitting a quadratic $\phi = \alpha\tau^2 + \beta\tau + \gamma$ through three points and finding it's minimum. I.e. we need to find coeffients $\alpha, \beta, \gamma$ so that

$$\begin{pmatrix} \phi(\tau_a) \\ \phi(\tau_b) \\ \phi(\tau_c) \end{pmatrix} = \begin{pmatrix} \tau_a^2 & \tau_a & 1 \\ \tau_b^2 & \tau_b & 1 \\ \tau_c^2 & \tau_c & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \tag{6.10}$$

whence the required line-search minimum is given by $\tau_{\min} = \frac{-\beta}{2\alpha}$

### 6.1.2 Non-Linear Conjugate Gradients

The non-linear CG method is derived from the CG method, again using lineseach. It can be written

**Algorithm 3** Non-linear Conjugate Gradients (Fletcher-Reeves)

$k = 0$, $\boldsymbol{f}_0 =$ initial guess
$\boldsymbol{p}_0 = \boldsymbol{r}_0 = -\Phi'(\boldsymbol{f}_0)$
**while** stopping criterion unfulfilled **do**
  $\tau_k = \arg\min_{\tau>0} \Phi(\boldsymbol{f}_k + \tau \boldsymbol{p}_k)$
  $\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \tau \boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = -\Phi'(\boldsymbol{f}_{k+1})$
  $\beta_k = \frac{||r_{k+1}||^2}{||r_k||^2}$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$
  $k = k + 1$
**end while**

An alternative (more commonly used) is the Polak-Ribiere algorithm which is identical except for the step

$$\beta_k = \frac{\langle \boldsymbol{r}_{k+1} - \boldsymbol{r}_k, \boldsymbol{r}_{k+1} \rangle}{\langle \boldsymbol{r}_{k+1} - \boldsymbol{r}_k, \boldsymbol{p}_k \rangle} \tag{6.11}$$

## 6.2 Newton Methods

From a quadratic approximation of the objective function

$$\Phi(\boldsymbol{f} + \boldsymbol{h}) \simeq \Phi(\boldsymbol{f}) + \langle \Phi'(\boldsymbol{f}), \boldsymbol{h} \rangle + \frac{1}{2} \langle \boldsymbol{h}, \Phi''(\boldsymbol{f}) \boldsymbol{h} \rangle \tag{6.12}$$

if $\Phi''(\boldsymbol{f})$ is positive definite then Eq. 6.12 has a unique minimiser found by solving

$$\Phi''(\boldsymbol{f}) \boldsymbol{h} = -\Phi'(\boldsymbol{f}) \,, \tag{6.13}$$

and the iterative Newton scheme is

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k - (\Phi''(\boldsymbol{f}_k))^{-1} \Phi'(\boldsymbol{f}_k) \tag{6.14}$$

Operator $\Phi''(\boldsymbol{f})$ is the *Hessian* of $\Phi$. Note that for a linear least squares problem $\Phi(\boldsymbol{f}) = ||\mathsf{A}\boldsymbol{f} - \boldsymbol{g}||^2$ we would have :

$$-\Phi'(\boldsymbol{f}) \rightarrow \mathsf{A}^{\mathrm{T}}(\boldsymbol{g} - \mathsf{A}\boldsymbol{f}) \,, \quad \Phi''(\boldsymbol{f}) = \mathsf{A}^{\mathrm{T}}\mathsf{A} \tag{6.15}$$

and Eq. 6.14 with $\boldsymbol{f}_0 = \boldsymbol{0}$ is just the same as solving the Moore-Penrose inverse

$$\mathsf{A}^{\mathrm{T}}\mathsf{A}\boldsymbol{f}_1 = \mathsf{A}^{\mathrm{T}}\boldsymbol{g} \tag{6.16}$$

For a non-linear least-squares problem we have

$$-\Phi'(\boldsymbol{f}) \rightarrow A^{'*}(\boldsymbol{f})[\boldsymbol{g} - A(\boldsymbol{f})]\,, \quad \Phi''(\boldsymbol{f}) \rightarrow A^{'*}(\boldsymbol{f})A^{'}(\boldsymbol{f}) - A^{''}(\boldsymbol{f})[\boldsymbol{g} - A(\boldsymbol{f})]\,. \tag{6.17}$$

The presence of the second term on the right in Eq. 6.17 involves the *second Fréchet derivative* of the forward mapping $A$ which can be difficult to compute. It further can lead to the Hessian becoming no-longer positive definite. Finally, as the model fits the data better, this term tends to zero since it operates on the discrepency $\boldsymbol{g} - A(\boldsymbol{f})$. For these reasons, the Newton step Eq. 6.14 is replaced by the so-called *Gauss-Newton* iteration

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k - \left(A^{'*}(\boldsymbol{f}_k)A^{'}(\boldsymbol{f}_k)\right)^{-1} A^{'*}(\boldsymbol{f}_k)[\boldsymbol{g} - A(\boldsymbol{f}_k)] \tag{6.18}$$

Note that in the Gauss-Newton, the operator $\left(A^{'*}(\boldsymbol{f}_k)A^{'}(\boldsymbol{f}_k)\right)^{-1}$ is symmetric positive definite, and thus the update direction is guaranteed to be a descent direction. However, the step length given in Eq. 6.18 is not necessarily a minimiser along the direction of step, so linesearch should still be used. Thus the Newton method will be

**Algorithm 4** Non-linear Newton

$k = 0$, $\boldsymbol{f}_0 =$ initial guess
**while** stopping criterion unfulfilled **do**
  $\boldsymbol{p}_k = -\left(\Phi''(\boldsymbol{f}_k)\right)^{-1}\Phi'(\boldsymbol{f}_k)$
  $\tau_k = \arg\min_{\tau>0} \Phi(\boldsymbol{f}_k + \tau\boldsymbol{p}_k)$
  $\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \tau\boldsymbol{p}_k$
  $k = k + 1$
**end while**

The Gauss-Newton version is got by replacing $\Phi''(\boldsymbol{f}_k)$ with its Gauss-Newton approximation $\Phi''_{\mathrm{GN}}(\boldsymbol{f}_k)$.

### 6.2.1 Levenberg-Marquardt

An alternative to the Gauss-Newton method (which is after all, only available for problems involving an explicit mapping $A : X \mapsto Y$), are methods which regularise the inversion step eq.(6.14). Specifically, if we consider

$$\left(\Phi''(\boldsymbol{f}) + \lambda \mathsf{I}\right)\boldsymbol{h} = \Phi'(\boldsymbol{f}) \tag{6.19}$$

then the operator on the left will be symmetric positive definite if $\lambda > -\min\{\sigma(\Phi''(\boldsymbol{f}))\}$, i.e. if we add a diagonal term with weighting greater than the minimum of the eigenvalues of the Hessian. Then this modified Hessian will be invertible and the product with the gradient will give a descent direction. Note that this is not regularisation in the sense we have used it before. We are still only optimising the function $\Phi(\boldsymbol{f})$ not an additional regularisation term, even though they have the same form.

In the *Levenberg-Marquardt* approach the weighting $\lambda$ is variable. The direction $\boldsymbol{p} = (\Phi''(\boldsymbol{f}) + \lambda \mathsf{I})^{-1} \Phi'(\boldsymbol{f})$ tends towards the descent direction, whereas as $\lambda \to 0$ the search direction becomes closer to the Newton direction.

Starting with an initially large value, the value of $\lambda$ is *(*decreased) if it proves to be a descent direction, and *increase* if not. Here is an outline of the method

**Algorithm 5** Levenberg-Marquardt

$k = 0$, $\boldsymbol{f}_0 =$ initial guess, $\lambda =$ initial value (high)
**while** stopping criterion unfulfilled **do**
  Find Step: $\boldsymbol{p}_k = -(\Phi''(\boldsymbol{f}_k) + \lambda \mathsf{I})^{-1} \Phi'(\boldsymbol{f}_k)$
  **if** $\Phi(\boldsymbol{f}_k + \boldsymbol{p}_k) < \Phi(\boldsymbol{f}_k)$ % ($\boldsymbol{p}_k$ is a descent direction)
    $\lambda = \lambda / C$ %(scale down $\lambda$)
  **else**
    $\lambda = \lambda * C$ %(scale up $\lambda$)
    **go to** Find Step
  **endif**
  $\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \boldsymbol{p}_k$ %(add the successful step. Do not use additional line search).
  $k = k + 1$
**end while**

## 6.3 Generic methods in MatLab

Matlab offers three generic functions for optimisation :

- fminsearch - a method for functions when no derivatives are present
- fminunc - a method for unconstrained optimisation
- fmincon - a method for constrained optimisation

Here are some Examples

```
%-------------------------Nedler-Mead-------------------------

NMopts = optimset('LargeScale','off','Display','iter','TolX',1e-4, 'TolFun',1e-7,
         'GradObj','off','PlotFcns',{@optimplotx,@optimplotfval});
[xOptNM,funcValNM,exitFlagNM,outputStructuresNM] = fminsearch(@(x) rosenbrock(x),
          xini,NMopts);

%-------------------------Trust-Region-------------------------
TRopts = optimset('LargeScale','off','Display','iter','TolX',1e-4, TolFun',1e-7,
         'GradObj','off','PlotFcns',{@optimplotstepsize,@optimplotfval});
[xOptTR,funcValTR,exitFlagTR,outputStructuresTR] = fminunc(@(x) rosenbrock(x),
          xini,TRopts);

%------------------------- Gradient Information ------------------------
TRGopts = optimset('LargeScale','on','Display','iter','TolX',1e-4, 'TolFun',1e-7,
         'GradObj','on','Hessian','off','PlotFcns',{@optimplotstepsize,@optimplotfval});
[xOptTRG,funcValTRG,exitFlagTRG,outputStructuresTRG] = fminunc(@(x) rosenbrock(x),
          xini,TRGopts);

%-------------- Gradient Information steepest descent --------------------
SDopts = optimset('LargeScale','off','Display','iter','TolX',1e-4, 'TolFun',1e-7,
         'GradObj','on','HessUpdate','steepdesc','PlotFcns',
         {@optimplotstepsize,@optimplotfval});
[xOptSD,funcValSD,exitFlagSD,outputStructuresSD] = fminunc(@(x) rosenbrock(x),
          xini,SDopts);

%------------------------- Hessian Information ------------------------
FNopts = optimset('LargeScale','on','Display','iter','TolX',1e-4, 'TolFun',1e-7,
           'GradObj','on','Hessian','on','PlotFcns',
           {@optimplotstepsize,@optimplotfval});
[xOptFN,funcValFN,exitFlagFN,outputStructuresFN] = fminunc(@(x) rosenbrockFN(x),
          xini,FNopts);
```

# 7 Statistical Methods

So far we considered linear and non-linear least-squares problems. The key point was that the *data-fitting* metric was a squared error. From the Bayesian point of view this corresponds to saying that the measurement error is Gaussian distributed with zero mean; the case of a non-diagonal data covariance matrix is also included in this model. However, there are other models for the measurement noise.

## 7.1 Maximum Likelihood

The Maximum likelihood method is the probabilistic interpretation of data fitting. Suppose a random vector $\boldsymbol{X}$ has a joint probability density function $P_{\boldsymbol{X}}(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an unknown parameter vector, and a data vector $\boldsymbol{g}$ is a given realisation of $\boldsymbol{X}$, i.e. a draw from the probabilty space with distribution $P_{\boldsymbol{X}}$.

A *maximum likelihood estimator* (MLE) for $\boldsymbol{\theta}$ given $\boldsymbol{g}$ is the vector $\hat{\boldsymbol{\theta}}$ which maximises the *likelihood* function

$$L(\boldsymbol{\theta}) := P_{\boldsymbol{X}}(\boldsymbol{g};\boldsymbol{\theta}) \tag{7.1}$$

More conveniently it is the minimiser of the *negative log likelihood*

$$\ell(\boldsymbol{\theta}) := -\log P_{\boldsymbol{X}}(\boldsymbol{g};\boldsymbol{\theta}) \tag{7.2}$$

Now we suppose that the forward model gives rise to observables

$$\boldsymbol{y} = A(\boldsymbol{f}) + \boldsymbol{\eta} \tag{7.3}$$

that are random variables by virtue of the $\boldsymbol{\eta}$ being a random variable. We assume that we can find the parameters $\boldsymbol{\theta}$ of the PDF of the data by taking moments of the forward model :

$$\boldsymbol{\theta} = \mathbb{E}[q(\boldsymbol{y})]$$

for some function $q$. E.g.

$$\begin{aligned} \overline{\boldsymbol{\theta}} &= \mathbb{E}[\boldsymbol{y}] \\ \operatorname{cov}(\boldsymbol{\theta}) &= \mathbb{E}[(\boldsymbol{y}-\overline{\boldsymbol{\theta}})(\boldsymbol{y}-\overline{\boldsymbol{\theta}})^{\mathrm{T}}] \end{aligned}$$

### 7.1.1 Gaussian Probability Model

A continuous random vector $\boldsymbol{X}$ has a Gaussian, or *Normal* distribution if its multivariate PDF is

$$P(\boldsymbol{x};\boldsymbol{\mu},\mathsf{C}) = \frac{1}{\sqrt{(2\pi)^n \det C}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathsf{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \tag{7.4}$$

The Normal distribution has $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ and $\mathrm{cov}(\boldsymbol{X}) = \mathsf{C}$ and is denoted

$$\boldsymbol{X} \sim N(\boldsymbol{\mu}, \mathsf{C})$$

If the noise is modelled by a Normal distribution, using Eq. 7.4 we can find the *maximum likelihood estimate* for $\boldsymbol{\mu}$ given data $\boldsymbol{g}$ by maximising $P(\boldsymbol{g}; \boldsymbol{\mu})$, equivalent to *minimising* the negative loglikelihood

$$\ell(\boldsymbol{\mu}) = -\log P(\boldsymbol{g}; \boldsymbol{\mu}, \mathsf{C}) = \frac{1}{2}(\boldsymbol{g} - \boldsymbol{\mu})^{\mathrm{T}} \mathsf{C}^{-1} (\boldsymbol{g} - \boldsymbol{\mu}) + k = \frac{1}{2} ||\boldsymbol{g} - \boldsymbol{\mu}||^2_{\mathsf{C}} + k \tag{7.5}$$

where $k$ is a constant independent of $\boldsymbol{\mu}$. We say that

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\mathsf{A}(\boldsymbol{f})] = \mathsf{A}(\boldsymbol{f})\,, \tag{7.6}$$

since $E[\boldsymbol{\eta}] = \mathbf{0}$. We thus find the minimiser of $\ell(\boldsymbol{\mu})$ is given by minimising

$$\Phi(\boldsymbol{f}, \boldsymbol{g}) = \frac{1}{2}(\boldsymbol{g} - \mathsf{A}(\boldsymbol{f}))^{\mathrm{T}} \mathsf{C}^{-1} (\boldsymbol{g} - \mathsf{A}(\boldsymbol{f})) \tag{7.7}$$

A Newton iteration would give

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k + \tau_k \left( A'^{*}(\boldsymbol{f}_k) \mathsf{C}^{-1} A'(\boldsymbol{f}_k) \right)^{-1} A'^{*}(\boldsymbol{f}_k) \mathsf{C}^{-1} [\boldsymbol{g} - A(\boldsymbol{f}_k)] \tag{7.8}$$

### 7.1.2 Poisson Probability Model

A discrete random vector $\boldsymbol{X}$ has a Poisson distribution with independent components if its multivariate PDF is

$$P(\boldsymbol{x}; \boldsymbol{\lambda}) = \begin{cases} \Pi_{i=1}^{n} \frac{\mathrm{e}^{-\lambda_i} \lambda_i^{x_i}}{x_i!}, & x_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

The Poisson distribution has $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\lambda}$ and $\mathrm{cov}(\boldsymbol{X}) = \mathrm{diag}(\boldsymbol{\lambda})$ and is denoted

$$\boldsymbol{X} \sim \mathrm{Poisson}(\boldsymbol{\lambda})$$

If the noise is modelled by a Poisson distribution, using Eq. 7.9 we can find the maximum likelihood estimate for $\boldsymbol{\lambda}$ given data $\boldsymbol{g}$ by maximising $P(\boldsymbol{g}; \boldsymbol{\lambda})$, equivalent to minimising the negative loglikelihood

$$\ell(\boldsymbol{\lambda}) = \sum_{i=1}^{N} (\lambda_i - g_i \log \lambda_i) + k \tag{7.10}$$

where $k$ is a constant independent of $\boldsymbol{\lambda}$. Now we suppose that the data is obtained through a forward operator, so that the actual Poisson variables are represented as

$$\boldsymbol{\lambda} = \mathsf{A}(\boldsymbol{f})$$

First take $\mathsf{A}$ to be linear. Eq. 7.10 becomes

$$\ell(\boldsymbol{g}, \mathsf{A}\boldsymbol{f}) = \sum_{i=1}^{N} \left( \sum_{j=1}^{M} A_{ij} f_j - g_i \log \left[ \sum_{j=1}^{M} A_{ij} f_j \right] \right) + k \tag{7.11}$$

from which we have

$$\frac{\partial \ell(\boldsymbol{g}, \mathsf{A}\boldsymbol{f})}{\partial \boldsymbol{f}} = \mathsf{A}^{\mathrm{T}}\mathbf{1} - \mathsf{A}^{\mathrm{T}} \left[ \frac{\boldsymbol{g}}{\mathsf{A}\boldsymbol{f}} \right] \tag{7.12}$$

where $\mathbf{1}$ is a vector of all ones, the same length as the data. To ensure positivity, make a change of variables $\boldsymbol{f} = \mathrm{e}^{\boldsymbol{\zeta}}$, whence

$$\frac{\partial \ell}{\partial \boldsymbol{\zeta}} = \frac{\partial \ell}{\partial \boldsymbol{f}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\zeta}} = \frac{\partial \ell}{\partial \boldsymbol{f}} \boldsymbol{f}$$

which leads to

$$\boldsymbol{f} \odot \mathsf{A}^{\mathrm{T}}\mathbf{1} = \boldsymbol{f} \odot \mathsf{A}^{\mathrm{T}} \left[ \frac{\boldsymbol{g}}{\mathsf{A}\boldsymbol{f}} \right] \tag{7.13}$$

where $\odot$ implies point wise multiplication. Now consider the operator

$$M(\boldsymbol{f}) = \frac{1}{\mathsf{A}^{\mathrm{T}}\mathbf{1}} \boldsymbol{f} \odot \mathsf{A}^{\mathrm{T}} \left[ \frac{\boldsymbol{g}}{\mathsf{A}\boldsymbol{f}} \right] \tag{7.14}$$

Then the maximum likelihood solution is a *fixed point* of this operator, i.e.

$$\boldsymbol{f}_{\mathrm{ML}} = M(\boldsymbol{f}_{\mathrm{ML}}) \tag{7.15}$$

The iterative *Expectation Maximisation* algorithm (called *Richardson-Lucy* in image processing) finds this fixed-point by iteration:

$$\boldsymbol{f}_{k+1} = \frac{\boldsymbol{f}_k}{\mathsf{A}^{\mathrm{T}}\mathbf{1}} \odot \mathsf{A}^{\mathrm{T}} \left[ \frac{\boldsymbol{g}}{\mathsf{A}\boldsymbol{f}_k} \right] \tag{7.16}$$

The above derivation is a little unrigorous. We can derive the same algorithm using rigorous EM theory, and using constrained optimisation theory, as we will see later.

In the non-linear case, the minimum of Eq. 7.10 requires

$$A'^{*}(\boldsymbol{f})\mathbf{1} = A'^{*}(\boldsymbol{f}) \frac{\boldsymbol{g}}{A(\boldsymbol{f})} \tag{7.17}$$

and we obtain the iteration

$$\boldsymbol{f}_{k+1} = \frac{\boldsymbol{f}_k}{A'^{*}(\boldsymbol{f}_k)\mathbf{1}} \odot A'^{*}(\boldsymbol{f}_k) \frac{\boldsymbol{g}}{A(\boldsymbol{f}_k)} \tag{7.18}$$

### 7.1.3 Gaussian Approximation to a Poisson Noise Process

The Central Limit Theorem tells us that in the limit of "large enough" number of observations all probability distributions tend to the Normal distribution.

In the case of a Poisson process with rate $\lambda$ the limiting Normal distribution is one with mean $\lambda$ and variance $\lambda$. Thus a possible alternative to minimising the negative Poisson log likelihood Eq. 7.10 is to minimise

$$\Phi(\boldsymbol{f},\boldsymbol{g}) = \frac{1}{2}(\boldsymbol{g}-\mathsf{A}\boldsymbol{f})^{\mathrm{T}}\left(\mathrm{diag}[\mathsf{A}\boldsymbol{f}]\right)^{-1}(\boldsymbol{g}-\mathsf{A}\boldsymbol{f}) \equiv \frac{1}{2}\left|\left|\frac{(\boldsymbol{g}-\mathsf{A}\boldsymbol{f})}{\sqrt{\mathsf{A}\boldsymbol{f}}}\right|\right|^2 \tag{7.19}$$

This is highly non-linear. We get another approximation if we use the data as an estimator of the variance and thus use

$$\Phi(\boldsymbol{f},\boldsymbol{g}) = \frac{1}{2}(\boldsymbol{g}-\mathsf{A}\boldsymbol{f})^{\mathrm{T}}\left(\mathrm{diag}[\boldsymbol{g}]\right)^{-1}(\boldsymbol{g}-\mathsf{A}\boldsymbol{f}) \equiv \frac{1}{2}\left|\left|\frac{(\boldsymbol{g}-\mathsf{A}\boldsymbol{f})}{\sqrt{\boldsymbol{g}}}\right|\right|^2 \tag{7.20}$$

which is solved (for a linear operator ) by

$$\boldsymbol{f}_* = \left(\mathsf{A}^*\mathrm{diag}\left[\frac{1}{\boldsymbol{g}}\right]\mathsf{A}\right)^{-1}\mathsf{A}^*\mathbf{1} \tag{7.21}$$

We can also use a gradient method,

$$\boldsymbol{f}^{(n+1)} = \boldsymbol{f}^{(n)} + \tau\left(\mathsf{A}^*\mathrm{diag}\left[\frac{1}{\boldsymbol{g}}\right]\mathsf{A}\right)^{-1}\mathsf{A}^*\left[\mathbf{1}-\frac{A\boldsymbol{f}^{(n)}}{\boldsymbol{g}}\right] \tag{7.22}$$

## 7.2 Kullback-Liebler and other divergences

In information theory, metrics are sometimes defined without a specific interpretation in terms of measurement noise

The Kullback-Liebler information divergence is an information-theoretic measure of the relative entropy between two functions $u$, $v$

$$KL(u,v) = \int\left(u(x)\log\left[\frac{u(x)}{v(x)}\right] - u(x) + v(x)\right)\mathrm{d}x\,. \tag{7.23}$$

If the functions are probability distributions the expression simplifies to the *relative entropy*

$$KL(P_1,P_2) = \int P_1(x)\log\left[\frac{P_1(x)}{P_2(x)}\right]\mathrm{d}x \tag{7.24}$$

since $\int P_1 = \int P_2 = 1$. The KL-divergence is always positive unless $u = v$ when $KL(u,u) = 0$. Note that the measure is not symmetric i.e. $KL(u,v) \neq KL(v,u)$.

The discrete version of KL-divergence measures the difference between vectors

$$KL(\boldsymbol{u}, \boldsymbol{v}) = \sum_i \left( u_i \log \left[ \frac{u_i}{v_i} \right] - u_i + v_i \right) . \tag{7.25}$$

Now consider

$$\Phi(\boldsymbol{f}, \boldsymbol{g}) = KL(\boldsymbol{g}, A(\boldsymbol{f})) \quad \Rightarrow \quad \Phi'(\boldsymbol{f}) = A'^{*}(\boldsymbol{f}) \left[ \mathbf{1} - \frac{\boldsymbol{g}}{A(\boldsymbol{f})} \right] \tag{7.26}$$

This is the same as the gradient of the negative Poisson log likelihood (NPLL) given inEq. 7.17. Note that both the KL-divergence and its derivative are zero for $\boldsymbol{g} = A(\boldsymbol{f})$, whereas the NPLL is not necessarily zero (but its derivative is); i.e. the NPLL and KL divergence differ by a constant.

The Hessian of the KL-divergence is

$$\Phi''(\boldsymbol{f}) = A'^{*}(\boldsymbol{f}) \mathrm{diag} \left( \frac{\boldsymbol{g}}{(A(\boldsymbol{f}))^2} \right) A'(\boldsymbol{f}) \tag{7.27}$$

which is symmetric positive definite.

## 7.3 The Bayesian MAP estimate

ML methods suffer from ill-posedness in just the same way that least-squares problems do. Rather than maximising the likelihood we maximise the Bayesian posterior

$$P(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{P(\boldsymbol{X})} \tag{7.28}$$

here $P(\boldsymbol{X}|\boldsymbol{\theta}) \equiv L(\boldsymbol{X}; \boldsymbol{\theta})$. The maximiser of Eq. 7.28 is the minimiser of

$$\Phi(\boldsymbol{f}, \boldsymbol{g}) = \ell(\boldsymbol{g}; A(\boldsymbol{f})) + \Psi(\boldsymbol{f}) \tag{7.29}$$

where $\Psi(\boldsymbol{f}) = -\log \pi(\boldsymbol{f})$. We find the gradient

$$\Phi'(\boldsymbol{f}, \boldsymbol{g}) = A'^{*}(\boldsymbol{f}) \left[ \mathbf{1} - \frac{\boldsymbol{g}}{A(\boldsymbol{f})} \right] + \Psi'(\boldsymbol{f}) \tag{7.30}$$

and we can apply the same derivation as in section 7.1.2 to obtain the *One Step Late* EM algorithm as

$$\boldsymbol{f}_{k+1} = \frac{\boldsymbol{f}_k}{A'^{*}(\boldsymbol{f}_k)\mathbf{1} + \Psi'(\boldsymbol{f})} \odot A'^{*}(\boldsymbol{f}_k) \frac{\boldsymbol{g}}{A(\boldsymbol{f}_k)} \tag{7.31}$$

This algorithm adds the regularisation term in the normalisation term, and now, obviously, it is impossible for the solution to exactly solve the original problem $\boldsymbol{g} = A(\boldsymbol{f})$ because this does not cancel the ratio

$$\frac{A'^{*}(\boldsymbol{f}_k)\mathbf{1}}{A'^{*}(\boldsymbol{f}_k)\mathbf{1} + \Psi'(\boldsymbol{f})} ;$$

however, this is not a contradiction : in the presence of regularisation we do not *want* the model to fit the data exactly, because doing so would propagate noise into the solution. This is the whole rationale for regularisation.

There is a danger with Eq. 7.31 : The gradient of the regulariser is often a derivative operator and may cause the denominator to become negative. Thus positivity might be lost.

## 7.4 Entropy as a prior

We can use Information measures as a prior. For example, the entropic prior is

$$\Psi(\boldsymbol{f}) = Ent(\boldsymbol{f}) = \sum_{i=1}^{n} f_i \log f_i - f_i \tag{7.32}$$

with derivative

$$\Psi'(\boldsymbol{f}) = Ent'(\boldsymbol{f}) = \log \boldsymbol{f} \tag{7.33}$$

Alternative, the KL-divergence can be used to provide the cross-entropy of *Mutual Information* to a reference vector $\boldsymbol{v}$

$$\Psi(\boldsymbol{f}, \boldsymbol{v}) = KL(\boldsymbol{f}, \boldsymbol{v}) = \sum_{i=1}^{n} (f_i \log f_i - f_i) - (f_i \log v_i - v_i) \tag{7.34}$$

with derivative

$$\Psi'(\boldsymbol{f}) = KL'(\boldsymbol{f}) = \log \left[ \frac{\boldsymbol{f}}{\boldsymbol{v}} \right] \tag{7.35}$$

# 8 Constrained Optimisation

A general constrained optimisation problem is stated

$$\begin{aligned} \text{minimise} \quad & \Phi(\boldsymbol{f}) && (8.1)\\ \text{subject to} \quad & c_i(\boldsymbol{f}) \leq 0\,, \quad i = 1 \ldots L && (8.2)\\ \text{and} \quad & c_e(\boldsymbol{f}) = 0\,, \quad e = 1 \ldots M && (8.3) \end{aligned}$$

where $c_i$ represent $L$ inequality constraints and $c_e$ represent $M$ equality constraints.

## 8.1 Parameter Transformation

Before looking at a general solution, we can consider a commonly occuring constraint and a simple work round. If $\boldsymbol{f} \succeq 0$ (i.e. $f_i \geq 0 \; \forall \, i$) we say that $\boldsymbol{f}$ requires a *positivity constraint*. We can make a simple change of variables

$$\boldsymbol{\zeta} = \log \boldsymbol{f} \quad \Leftrightarrow \quad \boldsymbol{f} = \mathrm{e}^{\boldsymbol{\zeta}} \tag{8.4}$$

and then solve an unconstrained problem for $\boldsymbol{\zeta}$ followed by exponentiation. For a non-linear problem we would have

$$\begin{aligned} & A(\boldsymbol{f}) \rightarrow A(\boldsymbol{\zeta}) = A(\mathrm{e}^{\boldsymbol{\zeta}}) \\ \Rightarrow \quad & A'(\boldsymbol{\zeta}) = A'(\boldsymbol{f}) \circ \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\zeta}} = A'(\boldsymbol{f}) \circ \boldsymbol{f} \end{aligned}$$

Finding an update $\boldsymbol{h}_{\boldsymbol{\zeta}}$ we could then write the update in $\boldsymbol{f}$ as a multiplicative update

$$\boldsymbol{\zeta}_{n+1} = \boldsymbol{\zeta}_n + \boldsymbol{h}_{\boldsymbol{\zeta}} \quad \Leftrightarrow \quad \boldsymbol{f}_{n+1} = \boldsymbol{f}_n \odot \mathrm{e}^{\boldsymbol{h}_{\boldsymbol{\zeta}}}$$

We could have used other transformations in place of Eq. 8.4. Another useful choice is the *logit Transform*

$$\boldsymbol{\zeta} = \log \frac{\boldsymbol{f} - a}{\boldsymbol{f} - b} \quad \Leftrightarrow \quad \boldsymbol{f} = \frac{b\mathrm{e}^{\boldsymbol{\zeta}} - a}{\mathrm{e}^{\boldsymbol{\zeta}} - 1} \tag{8.5}$$

which contrains $\boldsymbol{f}$ to lie between bounds $a \preceq \boldsymbol{f} \preceq b$.

## 8.2 Lagrange Multipliers and Dual Variables

Lagrangian is

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \Phi(\boldsymbol{f}) + \sum_{i=1}^{L} \lambda_i c_i(\boldsymbol{f}) + \sum_{e=1}^{M} \mu_e c_e(\boldsymbol{f}) \tag{8.6}$$

The *primary variables* are $\boldsymbol{f}$ and the *dual variables* are $\boldsymbol{\lambda}$,$\boldsymbol{\mu}$

## 8.3 Karush-Kuhn-Tucker (KKT) conditions

KKT conditions are a generalisation of Lagrange variables to inequalities.

The KKT conditions have to be satisfied at the optimal solution $\{\boldsymbol{f}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$

$$
\begin{aligned}
c_i(\boldsymbol{f}^*) &\leq 0\,, \quad i = 1 \ldots L && (8.7)\\
c_e(\boldsymbol{f}^*) &= 0\,, \quad e = 1 \ldots M && (8.8)\\
\lambda_i^* &\geq 0\,, \quad i = 1 \ldots L && (8.9)\\
\lambda_i^* c_i(\boldsymbol{f}^*) &= 0\,, \quad i = 1 \ldots L && (8.10)\\
\frac{\partial \Phi(\boldsymbol{f})}{\partial \boldsymbol{f}}|_{\boldsymbol{f}^*} + \sum_{i=1}^{L} \lambda_i \frac{\partial c_i(\boldsymbol{f})}{\partial \boldsymbol{f}}|_{\boldsymbol{f}^*} + \sum_{i=1}^{M} \mu_i \frac{\partial c_e(\boldsymbol{f})}{\partial \boldsymbol{f}}|_{\boldsymbol{f}^*} &= 0 && (8.11)
\end{aligned}
$$

## 8.4 Quadratic Programming

### 8.4.1 Equality Constraints

Consider the constrained optimisation problem

$$
\begin{aligned}
\text{minimise} \quad & \tfrac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{P}\boldsymbol{f} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{f} + r && (8.12)\\
\text{subject to} \quad & \mathsf{A}\,\boldsymbol{f} = \boldsymbol{b} && (8.13)
\end{aligned}
$$

where $\mathsf{P}$ is a symmetric positive definite $N \times N$ matrix, $\boldsymbol{q}$, $\boldsymbol{f}$ are vectors of length $N$, $\mathsf{A}$ is a $M \times N$ constraint matrix (with $M < N$) and $\boldsymbol{b}$ is a vector of length $M$.

It is equality only problem. The Lagrangian is written

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{P}\boldsymbol{f} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{f} + r + \boldsymbol{\mu}^{\mathrm{T}}\left[\mathsf{A}\boldsymbol{f} - \boldsymbol{b}\right] \tag{8.14}$$

We see that

$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{f}, \boldsymbol{\mu})}{\partial \boldsymbol{f}} &= \mathsf{P}\boldsymbol{f}^* + \boldsymbol{q} + \mathsf{A}^{\mathrm{T}}\boldsymbol{\mu}^* && (8.15)\\
\frac{\partial \mathcal{L}(\boldsymbol{f}, \boldsymbol{\mu})}{\partial \boldsymbol{\mu}} &= \mathsf{A}\boldsymbol{f}^* - \boldsymbol{b} && (8.16)
\end{aligned}
$$

Only the 2nd and 5th KKT conditions are relevent. These are exactly what we see in eqs.(8.15) and eqs.(8.16)

$$\mathsf{A}\boldsymbol{f}^* = \boldsymbol{b}\,, \quad \mathsf{P}\boldsymbol{f}^* + \boldsymbol{q} + \mathsf{A}^{\mathrm{T}}\boldsymbol{\mu}^* = 0$$

which can be written as

$$\begin{pmatrix} \mathsf{P} & \mathsf{A}^{\mathrm{T}} \\ \mathsf{A} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{f}^* \\ \boldsymbol{\mu}^* \end{pmatrix} = \begin{pmatrix} -\boldsymbol{q} \\ \boldsymbol{b} \end{pmatrix}$$

Since $P$ was spd, so is the augmented matrix

### 8.4.2 Inequality Constraints and the Active Set Method

Now consider the more general quadratic programming problem, with linear constraints

$$\begin{aligned} \text{minimise} \quad & \tfrac{1}{2}\boldsymbol{f}^{\mathrm{T}}\mathsf{P}\boldsymbol{f} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{f} + r && (8.17)\\ \text{subject to} \quad & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{f} \geq b_i\,, \quad i = 1\ldots L && (8.18)\\ \text{and} \quad & \boldsymbol{a}_e^{\mathrm{T}}\boldsymbol{f} = b_e\,, \quad e = 1\ldots M && (8.19) \end{aligned}$$

In the *primal active set method* certain constraints, indexed by the *active set* $\mathcal{A}$, are regarded as equalities whilst the rest are temporarily disregarded. On each iteration the constraints that are active are reevaluated. If $\boldsymbol{f}$ is feasible and $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{f} = b_i$ we say that the $i^{\text{th}}$ inequality constraint is *active* at $\boldsymbol{f}$. If $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{f} \geq b_i$ then this constraint is *inactive*. The equality constraints are always active. Note the connection to the KKT condition eq(8.10) : when the $i^{\text{th}}$ inequality constraint is satisfied, the corresponding Lagrangian variable $\lambda_i$ must be zero and it is removed from the active set. When the inequality is not satisfied the $\lambda_i$ has a positive value and the system seeks to find a point on the feasible region boundary for this constraint.

Now consider that $\boldsymbol{f}$ is a feasible point (i.e. satisfying the constraints) and notice that

$$\Phi(\boldsymbol{f} + \boldsymbol{h}) = \Phi(\boldsymbol{f}) + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{q} + \frac{1}{2}\boldsymbol{h}^{\mathrm{T}}\mathsf{P}\boldsymbol{h}$$

Also since for each active constraint $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{f} = b_i$ then

$$\boldsymbol{a}_i^{\mathrm{T}}(\boldsymbol{f} + \boldsymbol{h}) = b_i \Rightarrow \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{h} = 0$$

This means that an update step $\boldsymbol{h}$ must be tangent to the hyperplane of each constraint $i$. Therefore, given a feasible point $\boldsymbol{f}$ and a set of active constraints, we can solve the equality constrained quadratic equation

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\boldsymbol{h}^{\mathrm{T}}\mathsf{P}\boldsymbol{h} + \boldsymbol{q}^{\mathrm{T}}\boldsymbol{h} && (8.20)\\ \text{subject to} \quad & \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{h} = 0\,, \quad i \in \mathcal{A} && (8.21)\\ \text{and} \quad & \boldsymbol{a}_e^{\mathrm{T}}\boldsymbol{h} = 0\,, \quad e = 1\ldots M && (8.22) \end{aligned}$$

which results to solve a linear problem

$$\begin{pmatrix} \mathsf{P} & \tilde{\mathsf{A}}_{\mathrm{I}}^{\mathrm{T}} & \mathsf{A}_{\mathrm{E}}^{\mathrm{T}} \\ \tilde{\mathsf{A}}_{\mathrm{I}} & 0 & 0 \\ \mathsf{A}_{\mathrm{E}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{q} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \tag{8.23}$$

where $\tilde{\mathsf{A}}_{\mathrm{I}}$ is the matrix of the inequality constraints that are active. The vector $\boldsymbol{h}$ is an update for $\boldsymbol{f}$ that satisfies the currently active constraints. It needs to be tested also against the other, currently inactive constraints. If it satisfies these then

$$\boldsymbol{f} \rightarrow \boldsymbol{f} + \boldsymbol{h}$$

otherwise, a linesearch is taken in direction $\boldsymbol{h}$

$$\tau = \min \left[ 1, \min_{j \in \overline{\mathcal{A}} \& (\boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{h} > 0)} \frac{b_j - \boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{f}}{\boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{h}} \right] \tag{8.24}$$

and

$$\boldsymbol{f} \rightarrow \boldsymbol{f} + \tau \boldsymbol{h}$$

Note that eq(8.24) gives the projection onto the hyperplanes of all the inactive constraints. The minimum of the projection lengths is the closet hyperplane. Since only the inactive constraints are projected, the projection distances are always positive. If $\tau < 1$ then a new index is added to the active set list corresponding to the index $p$ which gave the minimum in eq(8.24). If any of the $\lambda_i$ are negative then the corresponding indices are removed from the active set list.

The algorithm is

---

1. given $\boldsymbol{f}^{(1)}$ feasible and $\mathcal{A}$ set $k = 1$
2. **if** $\boldsymbol{h} = 0$ does not solve eq(8.20) go to 8.
3. Compute Lagrange multipliers $\boldsymbol{\lambda}$.
4. **if** all $\lambda_i \geq 0$ terminate
5. **else** find index $l$ for which $\lambda_l$ is minimum,
6. remove $l$ from index set
7. **endif**
8. Solve eq(8.23) for $\boldsymbol{h}$
9. Find $\tau$ from eq(8.24)
10. $\boldsymbol{f}^{(k+1)} = \boldsymbol{f}^{(k)} + \tau \boldsymbol{h}$
11. if $\tau < 1$ add $p$ to $\mathcal{A}$
12. $k = k + 1$, go to 2.

---

# 9 Stochastic Optimisation

Consider an ill-posed inverse problem where a physical quantity needs to be recovered from an indirect measurement $\boldsymbol{g}$ modelled by

$$\boldsymbol{g} = Af + \epsilon, \tag{9.1}$$

where $\boldsymbol{g} \in \mathbb{R}^k$ is the measurement data vector, $f \in \mathbb{R}^n$ is a discrete representation of the physical quantity of interest, $A$ is a linear operator modelling the measurement and $\epsilon$ is random noise taking values in $\mathbb{R}^k$.

In Bayesian inversion the quantities $\boldsymbol{g}$ and $f$ and $\epsilon$ are considered as random variables. The complete solution to the inverse problem is the *posterior distribution* defined using the Bayes formula:

$$\pi_{\mathsf{post}}(f \,|\, \boldsymbol{g}) = \frac{\pi(f)\,\pi(\boldsymbol{g} \,|\, f)}{\pi(\boldsymbol{g})}, \tag{9.2}$$

where $\pi(f)$ is the *prior model* representing *a priori* information about the unknown and $\pi(\boldsymbol{g})$ is a normalizing constant. The conditional probability $\pi(\boldsymbol{g}|f)$ is the *likelihood model* describing the measurement. In the case of independent white noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ we have

$$\pi(\boldsymbol{g} \,|\, f) = \pi_\epsilon(\boldsymbol{g} - Af) = C \exp\big(-\frac{1}{2\sigma^2}\|Af - \boldsymbol{g}\|_2^2\big), \tag{9.3}$$

where $C$ is a normalization constant.

Practical measurement gives us a realization $\widehat{\boldsymbol{g}} \in \mathbb{R}^k$ of the random variable (9.1), which is substituted into (9.2):

$$\pi_{\mathsf{post}}(f \,|\, \widehat{\boldsymbol{g}}) = \frac{\pi(f)\,\pi(\widehat{\boldsymbol{g}} \,|\, f)}{\pi(\widehat{\boldsymbol{g}})}. \tag{9.4}$$

We recover $f$ from $\widehat{\boldsymbol{g}}$ as the conditional mean (CM) estimate

$$f^{\mathsf{CM}} := \int_{T_n(X)} f\,\pi_{\mathsf{post}}(f \,|\, \widehat{\boldsymbol{g}})\, d\mu(f). \tag{9.5}$$

We also compute confidence intervals for uncertainty quantification.

For general references on Bayesian inversion, see [9].

## 9.1 Posterior Sampling and MCMC

Numerical evaluation of the estimate $f^{\mathsf{CM}}$ using (9.5) involves integration over $n$-dimensional space with large $n$. The curse of dimensionality prevents us from using integration quadratures. Instead, we approximate $f^{\mathsf{CM}}$ by generating a set of random samples $\{f^{(\ell)}, \ell = 1, 2, \ldots, N\}$ from the posterior density $\pi_{\mathsf{post}}(f \,|\, \widehat{\boldsymbol{g}})$ in such a way that

$$\frac{1}{N}\sum_{\ell=1}^{N} f^{(\ell)} \approx \int_{\mathbb{R}^n} f\,\pi_{\mathsf{post}}(f \,|\, \widehat{\boldsymbol{g}})\, df = f^{\mathsf{CM}}. \tag{9.6}$$

Confidence intervals can be approximately calculated from the variances of the sequence $\{f^{(\ell)}, \ell = 1, 2, \ldots, N\}$.

As a typical example Markov chain Monte Carlo (MCMC) method for creating the sequence $\{f^{(\ell)}, \ell = 1, 2, \ldots, N\}$ we describe the Metropolis-Hastings algorithm. The initial state $f^{(1)}$ is arbitrary. Assume that we know $f^{(\ell-1)}$ and need to costruct $f^{(\ell)}$. We create a candidate $c \in \mathbb{R}^n$ by drawing it randomly from a so-called *proposal distribution*, for example from the Gaussian distribution $\mathcal{N}(f^{(\ell-1)}, \beta^2)$ with a suitable standard deviation $\beta > 0$. Now we compare the posterior probability $\pi_{\mathsf{post}}(c \,|\, \widehat{\boldsymbol{g}})$ of the candidate to $\pi_{\mathsf{post}}(f^{(\ell-1)} \,|\, \widehat{\boldsymbol{g}})$. If the candidate is more probable than $f^{(\ell-1)}$, we set $f^{(\ell)} := c$. Otherwise we choose on of the following two options with certain probabilities: $f^{(\ell)} := f^{(\ell-1)}$ or $f^{(\ell)} := c$.

Other popular MCMC methods include the Gibbs sampler and various adaptive methods, such as DRAM and SCAM.

**References for this section :** [9].

# 10 Example Problems

## 10.1 Denoising

Consider an image corrupted by noise

$$\boldsymbol{g} = \boldsymbol{f} + \boldsymbol{n} \tag{10.1}$$

This takes the same general form as our linear inverse problems, with the forward mapping being the identity, if we look for a solution

$$\boldsymbol{f}_* = \arg\min_{\boldsymbol{f}} \left[ \frac{1}{2} ||\boldsymbol{g} - \boldsymbol{f}||^2 + \frac{\alpha}{2} ||\boldsymbol{f}||^2_{\Gamma} \right] \tag{10.2}$$

The direct solution is found by solving

$$(\mathsf{I} - \alpha\Gamma)\, \boldsymbol{f}_* = \boldsymbol{g} \tag{10.3}$$

### 10.1.1 Comment on Fourier Denoising

Suppose that the regulariser is a linear convolution operator, i.e.

$$\Psi(\boldsymbol{f}) = ||K * f||^2 = \int_\Omega |K(x) * f(x)|^2 \mathrm{d}x = \int_\Omega \hat{K}(k)\hat{f}(k)\mathrm{d}k \tag{10.4}$$

Then eq.(10.3) is directly solved by

$$\hat{f}_* = \frac{\hat{g}(k)}{1 + \alpha\hat{K}^2(k)} \tag{10.5}$$

E.g. for Tikhonov 1st order regularisation

$$\hat{f}_* = \frac{\hat{g}(k)}{1 + \alpha k^2}$$

which is the classical Weiner filter.

### 10.1.2 Series Solutions

Provided that the matrix norm $\alpha||\Gamma|| < 1$ we can develop a Neumann series for eq.(10.3) :

$$\boldsymbol{f}_* = \boldsymbol{g} + \alpha\Gamma\boldsymbol{g} + (\alpha\Gamma)^2\boldsymbol{g} + \ldots \tag{10.6}$$

Compare this to the Krylov subspace series

$$\mathbb{K}_\alpha = \left\{ \boldsymbol{g}, (\mathsf{I} - \alpha\Gamma)\, \boldsymbol{g}, (\mathsf{I} - \alpha\Gamma)^2\, \boldsymbol{g}, \ldots \right\} \tag{10.7}$$

Clearly the space spanned by $\mathbb{K}_\alpha$ is the same as the space spanned by

$$\mathbb{K}_{\mathsf{Neum}} = \left\{ \boldsymbol{g}, \Gamma \boldsymbol{g}, \Gamma^2 \boldsymbol{g}, \ldots \right\} \tag{10.8}$$

### 10.1.3 Iterative Solutions

The gradient of $\Psi(\boldsymbol{f})$ in eq.(10.3) is given by

$$\Psi'(\boldsymbol{f}) = (\mathsf{I} + \alpha\Gamma)\, \boldsymbol{f} - \boldsymbol{g}\,.$$

We can therefore form an iterative gradient descent solution by

$$\begin{aligned} \boldsymbol{f}^{(1)} &= \boldsymbol{g} \\ \boldsymbol{f}^{(n+1)} &= \boldsymbol{f}^{(n)} - \tau\, (\mathsf{I} + \alpha\Gamma)\, \boldsymbol{f}^{(n)} + \tau \boldsymbol{g} \end{aligned}$$

The step length $\tau$ can be chosen optimally, but we can also consider this iteration to be an evolution in time:

$$\frac{\partial \boldsymbol{f}}{\partial t} = (\boldsymbol{g} - \boldsymbol{f}) - \alpha \Gamma f \tag{10.9}$$

We note the comparison to a simple image scale space evolution scheme:

$$\frac{\partial \boldsymbol{f}}{\partial t} = -\Gamma \boldsymbol{f} \tag{10.10}$$

thus the gradient descent scheme can be considered as an image eveolution scheme, followed by adding back of the residual error $(\boldsymbol{g} - f)$.

$$\begin{aligned} \boldsymbol{f}^{(n+1/2)} &= \boldsymbol{f}^{(n)} - \Delta t \Gamma \boldsymbol{f}^{(n)} \\ \boldsymbol{f}^{(n+1)} &= \boldsymbol{f}^{(n+1/2)} + (\boldsymbol{g} - \boldsymbol{f}^{(n)}) \end{aligned}$$

This is an example of a *Split Operator* method. Note that the fictious time step $\Delta t$ takes the place of a regularisation parameter $\alpha$. The time evolutiuon concept may be particularly useful if we are considering non-linear regularisation. For example Total Variation will take the form

$$\frac{\partial \boldsymbol{f}}{\partial t} = \nabla \cdot \kappa(\boldsymbol{f}) \nabla \boldsymbol{f} \tag{10.11}$$

and we make take arbitrarily small steps in time to capture the non-linearity.

Finally we should note that image evolution schemes using an explicit scheme need to take account of the Courant-Freidrich-Levy (CFL) stability conditions which control stability. This sets an upper bound on the time step that can be taken. This limit is superceded if an explicit or semi-implicit evolution scheme is used.

## 10.2 The Radon Transform

The forward projection operator of the Radon transform in 2D as an integral operator mapping a function $f(x,y)$ to a function $g(s,\theta)$, by integrating along the parallel lines $\mathbf{r}\cdot\hat{\mathbf{n}} = s$.

$$g(s,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(s-(y\cos\theta - x\sin\theta))\mathrm{d}x\mathrm{d}y \quad = \int_{\mathbf{r}\cdot\hat{\mathbf{n}}=s} f(x,y)\mathrm{d}\ell \tag{10.12}$$

The adjoint of the Radon Transform maps a function in $(s,\theta)$ to one in $(x,y)$; it is known as *Back-projection*

$$h(x,y) = \int_{-\infty}^{\infty}\int_{0}^{\pi} b(s,\theta)\delta(s-(y\cos\theta - x\sin\theta))\mathrm{d}\theta\mathrm{d}s \tag{10.13}$$

The 1D Fourier Transform $\mathcal{F}_{s\to k}$ of the Radon Transform of a 2D function $f(x,y)$ is the same as the 2D Fourier Transform $\mathcal{F}_{x\to k_x, y\to k_y}$ of $f(x,y)$ sampled along the line $k_x = k\cos\theta, k_y = k\sin\theta$.

$$\int_{-\infty}^{\infty}\mathrm{e}^{-\mathrm{i}ks}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(s-(y\cos\theta - x\sin\theta))\mathrm{d}x\mathrm{d}y\mathrm{d}s \quad = \tag{10.14}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathrm{e}^{-\mathrm{i}ks} f(x,y)\delta(s-(y\cos\theta - x\sin\theta))\mathrm{d}s\mathrm{d}x\mathrm{d}y \quad = \tag{10.15}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathrm{e}^{-\mathrm{i}k(x\sin\theta - y\cos\theta)} f(x,y)\mathrm{d}x\mathrm{d}y \quad = \quad \hat{F}(k\sin\theta, k\cos\theta) \tag{10.16}$$

As an example, consider the Radon Transform of a 2D Gaussian function $f(x,y) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$ which is a Gaussian function $g(s,\theta) = \frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-\frac{s^2}{2\sigma^2}}$ that does not depend on $\theta$. In this particular case $\hat{G}(k,\theta) = \frac{\sigma}{\sqrt{2\pi}}\mathrm{e}^{-\frac{k^2\sigma^2}{2}}$ which obviously is also the radial form of the 2D Fourier Transform $\hat{F}(k_x,k_y) = \frac{\sigma^2}{2\pi}\mathrm{e}^{-\frac{(k_x^2+k_y^2)\sigma^2}{2}}$

If we back project we get

$$h(x,y) = \int_0^{\pi}\frac{1}{\sqrt{2\pi\sigma^2}}\mathrm{e}^{-\frac{(y\cos\theta - x\sin\theta)^2}{2\sigma^2}}\mathrm{d}\theta \tag{10.17}$$

and we clearly see that the back projection operator is not itself the inverse of the forward projection.

An exact reconstruction formula can be found by either i) back projection followed by convolution with a 2D filter or ii) convolution with a 1D filter of the Radon transform, followed by back-projection. To see this, we notice that integration of the back projection over all angles gives a weighting of $\frac{1}{|k|}$ in the Fourier domain. We may therefore filter by the inverse Fourier transform of $|k|$ in either 2D or 1D i) $h(x,y) = \mathcal{F}^{-1}[1/\sqrt{(k_x^2+k_y^2)}]$ or ii) $h(s) = \mathcal{F}^{-1}[1/|k|]$. These are approximately Laplacian functions (differentiating). The 2D case is hard to implement, since it is

circularly symmetric and requires interpolation. The 1D case is easier. In both cases they need to be adopised by smoothing otherwise the high frequencies will explode.

# A Further Notes on Differentiation

## A.1 Simple derivatives

Simple derivatives are examples of the general definition when X and Y are both $\mathbb{R}$ and therefore the mapping $A$ is a simple function $f$. Notice that the most general linear operator from $\mathbb{R}$ to $\mathbb{R}$ can be described by a simple scalar (the slope of the linear function), and the operation of $A'(x)h$ is just scalar multiplication. Thus, the derivative should be the scalar $f'(x)$ where

$$f(x+h) = f(x) + f'(x)h + \mathcal{O}(h)\,.$$

From this definition, we can find this scalar $f'(x)$ for each value of $x$ using limits:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}\,.$$

This is probably a more familiar definition but you should see now how it can be thought of as a computational device for the more general definition. Notice that the key to getting a handle on the Fréchet derivative is to first understand what a general linear operator looks like when going from X to Y and then trying to figure out how to describe the derivative linear operator.

## A.2 Derivatives of vector-valued functions

Suppose now that $\mathrm{X} = \mathbb{R}$ and $\mathrm{Y} = \mathbb{R}^n$. We usually write $\boldsymbol{f} : \mathbb{R} \mapsto \mathbb{R}^n$. A general linear mapping from $\mathbb{R}$ to $\mathbb{R}^n$ could be denoted with $\boldsymbol{c}x$ where $\boldsymbol{c}$ is a vector in $\mathbb{R}^n$ and $\boldsymbol{c}x$ denotes scalar-vector multiplication. Therefore, from the general definition we ought to be able to write

$$\boldsymbol{f}(x+h) = \boldsymbol{f}(x) + \boldsymbol{f}'(x)h + \mathcal{O}(h)\,.$$

Using limits we can as before come up with

$$\boldsymbol{f}'(x) = \lim_{h\to 0} \frac{\boldsymbol{f}(x+h) - \boldsymbol{f}(x)}{h}\,.$$

which is a straightforward extension of the simple case. In practice we usually recognize that

$$\boldsymbol{f}'(x) = [f_1'(x), f_2'(x), \ldots f_n'(x)]^{\mathrm{T}}$$

where $f_k'(x)$ are simple derivatives so we can compute them using the familiar rules.

## A.3 Derivatives with respect to vectors (matrices)

Now, suppose the mapping is $f : \mathbb{R}^n \mapsto \mathbb{R}$ (a multivariable function). The most general linear mapping from $\mathbb{R}^n$ to $\mathbb{R}$ is described by a constant length-$n$ vector, $\boldsymbol{c}$, so that

$$\boldsymbol{c}\boldsymbol{x} = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \langle \boldsymbol{x}, \boldsymbol{c} \rangle$$

is the value of the linear mapping. Thus, we ought to be able to define a derivative $f'(x)$ using a length $n$ vector. It is common in this case to use alternative notations to describe this vector.

$$f'(\boldsymbol{x}) \equiv \frac{\partial f}{\partial \boldsymbol{x}} \equiv \nabla f$$

and we sometimes call this linear operator the *gradient* of the vector. Thus, we can write

$$f(\boldsymbol{x} + \boldsymbol{h}) = f(\boldsymbol{x}) + \nabla^{\mathrm{T}} f(\boldsymbol{x})\boldsymbol{h} + \mathcal{O}(||\boldsymbol{h}||)\,.$$

In practice how can we express this gradient vector? We can find it projected against any direction of interest by choosing $\boldsymbol{h}$ to be a scaled vector: $\boldsymbol{h} = h\boldsymbol{q}$ where $\boldsymbol{q}$ is a unit-vector in the direction of interest:

$$f(\boldsymbol{x} + h\boldsymbol{v}) = f(\boldsymbol{x}) + h\,\langle \boldsymbol{v}, \nabla f \rangle + \mathcal{O}(h_i)$$

and using limits we can see that

$$(\nabla f, \boldsymbol{v}) = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})}{h}\,.$$

If $\boldsymbol{v}$ is chosen to be one of the standard Cartesian basis vectors, $\boldsymbol{e}_i$ for $\mathbb{R}^n$ (i.e. a one in one component and zeros in the rest) then we obtain the standard definition of partial derivative

$$(\nabla f, \boldsymbol{e}_i) = (\nabla f)_i = \frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{e}_i) - f(\boldsymbol{x})}{h}\,.$$

Here, the computation of the partial derivative is obtained from a more general perspective which shows that $\nabla f$ is the more general object and $\frac{\partial f}{\partial x_i}$ is just a projection of this vector against a standard basis. Under-standing the generality will help you make sense of the derivative when we get to mappings on uncountably infinite-dimensional spaces.

**Einstein notation** Suppose we are given a formula involving a vector or a matrix and we are trying to find the derivative with respect to the vector or matrix. This is usually best handled by using summation notation (Einstein notation) and using the rules for simple differentiation. Some texts define a collection of special rules for vector (matrix) differentiation. These can be difficult to remember and keep straight. Fortunately, they are superfluous and unnecessary if you get a little comfortable with Einstein notation.

For example, consider the following formula

$$f(\boldsymbol{x}) = (\boldsymbol{y} - \mathsf{A}\boldsymbol{x})^{\mathrm{T}}\mathsf{B}(\boldsymbol{y} - \mathsf{A}\boldsymbol{x}) + \boldsymbol{x}^{\mathrm{T}}\mathsf{C}\boldsymbol{x} + \boldsymbol{y}^{\mathrm{T}}\mathsf{D}\boldsymbol{y}\,.$$

To find $\frac{\partial f}{\partial \boldsymbol{x}}$ we could try to remember the various rules associated with differentiation with respect to vectors or, alternatively, we just write the expression using summation notation

$$f(x) = \sum_{i,k} \left( y_i - \sum_j A_{ij} x_j \right) B_{ik} \left( y_k - \sum_l A_{kl} x_l \right) + \sum_{i,j} x_i C_{ij} x_j + \sum_{i,j} y_i D_{ij} y_j \, .$$

Now, we can use the rules of simple differentiation to find ($\delta_{ij} = 0$ unless $i = j$ when it is 1).

$$\begin{aligned} \frac{\partial f}{\partial x_n} &= \sum_{i,k} \left( y_i - \sum_j A_{ij} x_j \right) B_{ik} - (A_{kl}\delta_{ln}) + \sum_{i,k} \left( - \sum_j A_{ij} \delta_{jn} \right) B_{ik} \left( y_k - A_{kl} x_l \right) \\ &\quad + \sum_{i,j} \left( x_i C_{ij} \delta_{jn} + \delta_{in} C_{ij} x_j \right) \, . \end{aligned} \tag{A.1}$$

After you do this several times, you start to notice that the $\sum$ sign is getting in the way. A simplified notation can be obtained by eliminating the sum and just remembering that repeated indices in the formula imply a sum over that index variable. Thus, the formula becomes

$$f(\boldsymbol{x}) = (y_i - A_{ij} x_j) B_{ik} (y_k - A_{kl} x_l) + x_i C_{ij} x_j + y_i D_{ij} y_j$$

and

$$\begin{aligned} \frac{\partial f}{\partial x_n} &= (y_i - A_{ij} x_j) B_{ik} (-A_{kl}\delta_{ln}) + (-A_{ij}\delta_{jn}) B_{ik} (y_k - A_{kl} x_l) + x_i C_{ij} \delta_{jn} + \delta_{in} C_{ij} x_j \\ &= -(y_i - A_{ij} x_j) B_{ik} A_{kn} - A_{in} B_{ik} (y_k - A_{kl} x_l) + x_i C_{in} + C_{nj} x_j \\ &= -y_i B_{ik} A_{kn} - A_{in} B_{ik} y_k + A_{ij} B_{ik} A_{kn} x_j + A_{in} B_{ik} A_{kl} x_l + x_i C_{in} + C_{nj} x_j \\ &= -A_{kn} (B_{ik} + B_{ki}) y_i + (A_{kn} (B_{ik} + B_{ki}) A_{ij} + C_{nj} + C_{jn}) x_j \, . \end{aligned}$$

Notice that in the last line we freely changed the indexes of summation to get the form into something we could more easily map back into matrix-vector notation. Now, if desired we can translate this back into matrix-vector notation.

$$\nabla f(\boldsymbol{x}) \equiv \frac{\partial f}{\partial \boldsymbol{x}} = -\mathsf{A}^{\mathrm{T}} (\mathsf{B}^{\mathrm{T}} + \mathsf{B}) \boldsymbol{y} + \left[ \mathsf{A}^{\mathrm{T}} (\mathsf{B}^{\mathrm{T}} + \mathsf{B}) \mathsf{A} + \mathsf{C} + \mathsf{C}^{\mathrm{T}} \right] \boldsymbol{x} \, .$$

To effectively use Einstein notation you have to get quite comfortable with the following table.

| Matrix Notation | Index Notation |
|---|---|
| $\mathsf{F}$ | $F_{ij}$ |
| $\mathsf{G}$ | $G_{jk}$ |
| $\mathsf{FG}$ | $F_{ij}G_{jk}$ |
| $\mathsf{F}^{\mathrm{T}}\mathsf{G}$ | $F_{ji}G_{jk}$ |
| $\mathsf{F}\mathsf{G}^{\mathrm{T}}$ | $F_{ij}G_{kj}$ |
| $\mathsf{F}^{\mathrm{T}}\mathsf{G}^{\mathrm{T}}$ | $F_{ji}G_{kj}$ |

The first column gives a result in matrix-notation and the second column provides the equivalent in index notation. The important thing to look at is the location of the repeated index for help in understanding whether a particular sum translates to a matrix or its transpose.

## A.4 Derivatives with respect to mappings from $\mathbb{R}^m$ to $\mathbb{R}^n$

Using our general definition we can make clear sense of what is meant by the derivative of a mapping from $\mathbb{R}^m$ to $\mathbb{R}^n$. Generally, we denote this mapping by $\boldsymbol{f} : \mathbb{R}^m \mapsto \mathbb{R}^n$ with $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ where $\boldsymbol{y}$ is an n-dimensionalvector and $\boldsymbol{x}$ is an m-dimensional vector. The most general linear mapping between these spaces can be written as $L\boldsymbol{x}$ where $L$ can be described by an $n \times m$ matrix $\mathsf{L}$. For example, if $\boldsymbol{x}$ is projected against an orthonormal basis $\{\boldsymbol{v}_j; j = 1 \ldots m\}$ of $\mathbb{R}^m$

$$x = \sum_{j=1}^{m} x_j \boldsymbol{v}_j$$

and if $\mathsf{L}\boldsymbol{x}$ is projected against an orthonormal basis $\{\boldsymbol{u}_i; i = 1 \ldots n\}$ of $\mathbb{R}^n$ then

$$L\boldsymbol{x} = \sum_{i=1}^{n} y_i \boldsymbol{u}_i = \sum_{j=1}^{m} x_j L\boldsymbol{v}_j$$

If we take the inner product of this result with $\{\boldsymbol{u}_i\}$ then we get for each $i = 1 \ldots n$

$$y_i := (L\boldsymbol{x}, \boldsymbol{u}_i) = \sum_{j=1}^{m} x_j (L\boldsymbol{v}_j, \boldsymbol{u}_i) = \sum_{j=1}^{m} L_{ij} x_j$$

This is a matrix-vector equation $\boldsymbol{y} = \mathsf{L}\boldsymbol{x}$ where $L_{ij} = (L\boldsymbol{v})j, \boldsymbol{u}_i)$ . Very often we choose $v_j$ and $u_i$ to be the standard Euclidean bases and simply describe $\mathsf{L}$ using its standard matrix representation.

Thus, according to the general definition we should be able to find a derivative operator $\mathsf{D}_{\boldsymbol{x}} \equiv \boldsymbol{f}'(\boldsymbol{x})$ such that

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{h}) = \boldsymbol{f}(\boldsymbol{x}) + \mathsf{D}_{\boldsymbol{x}}\boldsymbol{h} + \mathcal{O}(||h||) \,.$$

We can describe this linear operator $\mathsf{D}_{\boldsymbol{x}}$ using $n \times m$ scalars. Most commonly, the standard Euclidean bases are chosen to represent both $\boldsymbol{x}$ and $\boldsymbol{y}$ and so the matrix entries can be expressed in terms of specific partial derivatives. Deriving a specific form for these matrix components using limits is left as an exercise for the reader.

There are two commonly used multivariable derivatives that are related to the general Fréchet derivative. These are explained in the next two sections. These two derivatives are quite useful to help explain Newtonian physics but from an optimization standpoint they are not particularly useful and so we dont deal with them. They are included here only to show you how they fit in to the overall structure we are building.

**Divergence**

For the special case of a mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$ it is sometimes convenient (particularly for Physics applications) to define the divergence of the mapping

$$\nabla \cdot \boldsymbol{f} = \mathrm{Tr}(\mathsf{D}_{\boldsymbol{x}}) \,,$$

where

$$\mathrm{Tr}(\mathsf{A}) = \sum_j (\boldsymbol{u}_j, \mathsf{A}\boldsymbol{u}_j)$$

where $\boldsymbol{u}_j$ is any orthonormal basis. For a standard Cartesian bases this is

$$\nabla \cdot \boldsymbol{f} = \quad \sum_{i=1}^{n} \frac{\partial f_i}{\partial x_i} = \quad f_{i,i} = \quad D_{ii}(\boldsymbol{x}) \,,$$

where $f_{i,j} = \frac{\partial f_i}{\partial x_j}$ is another part of Einstein notation along with the implied sum over $i$. Notice also that we have used the Fréchet derivative matrix: $\{\mathsf{D}_{\boldsymbol{x}}\}_{ij} = D_{ij}(\boldsymbol{x})$.

### Curl

Another commonly used derivative that is applicable only for mappings from $\mathbb{R}^3$ to $\mathbb{R}^3$ is the curl. In Einstein notation in Cartesian coordinates the curl of $\boldsymbol{f}$ is a vector $\boldsymbol{g}$ defined as

$$\begin{aligned} \boldsymbol{g} &= \nabla \times \boldsymbol{f} \\ g_i &= \epsilon_{ijk} \frac{\partial f_k}{\partial x_j} = \epsilon_{ijk} f_{k,j} \\ \boldsymbol{g} &= \boldsymbol{\epsilon} : \mathsf{D}_{\boldsymbol{x}} \end{aligned}$$

where there is an implied sum over both $j$ and $k$, $f_{k,j} = \frac{\partial f_k}{\partial x_j} = D_{kj}(\boldsymbol{x})$ and $\epsilon_{ijk}$ is the $3 \times 3 \times 3$ Levi-Civita tensor which is defined to be 0 if any of the $i$,$j$, or $k$ are repeated indices, 1 if $(i, j, k)$ is an even permutation of (1, 2, 3) and -1 if $(i, j, k)$ is an odd permutation of (1, 2, 3). For example, $\epsilon_{113} = 0$ and $\epsilon_{231} = 1$ but $\epsilon_{132} = -1$. In particular, $\epsilon_{123} = \epsilon_{231} = \epsilon_{312} = 1$ while $\epsilon_{321} = \epsilon_{213} = \epsilon_{132} = -1$ and the rest of the components of $\epsilon_{ijk}$ are 0.

The notation $\mathsf{A} : \mathsf{B}$ to denote a tensor-matrix multiplication that involves two sums. This kind of notation gets bulky (thus the use of $\nabla \times \boldsymbol{f}$ in vector notation).

# B Variational Methods

## B.1 Generalised Differentiation

The reason derivatives arise in practice is that all derivatives try to capture the idea of an "instantaneous" rate-of-change of a mapping with respect to its input. In other words, the derivative of a mapping is another mapping that shows how the original mapping changes when the input change. The word "instantaneous" is precisely defined using the concept of limits. If we have some mapping and we are trying to find some optimum value for that mapping then at the optimum value, the instantaneous rate of change of the mapping will be zero. Thus, finding the zeros of the derivative of the mapping will be a method of finding optima. This procedure is the basic principle of calculus for mappings from real-numbers to real-numbers (regular functions), but also extends to all kinds of mappings such as those based on vectors, functions (variational calculus), complex-valued variables, and even operators.

### B.1.1 Definitions

Here we give the general definition of a (Fréchet) derivative of an arbitrary mapping from a normed vector space X to a normed vector space Y. The next sections will try to make better sense of this definition for the more common mappings we encounter. Suppose we have the mapping $A : \mathrm{X} \mapsto \mathrm{Y}$. The derivative of this operator at $x$ is denoted $A'(x)$ and is defined as the linear (and continuous, thus bounded) operator from X to Y such that

$$A(x+h) = A(x) + A'(x)h + \mathcal{O}(||h||) \tag{B.1}$$

where by definition of little-o notation:[2]

$$\lim_{||h||\to 0} \frac{\mathcal{O}(||h||)}{||h||} = 0 \tag{B.2}$$

(note that this limiting value of 0 is the 0-vector of the space Y). The notation $A'(x)h$ means operate on $h$ with the (linear) derivative operator $A'(x)$. Exactly what this operation is depends on the specifics of $A$ and the spaces X and Y. For this general definition $x$ and $h$ are elements of X and could be for example scalars, vectors (matrices), or functions. In other words, $x$ and $h$ could be one-dimensional, countably-dimensional, or uncountably-dimensional. Also, the addition on the left-hand side of the definition occurs in X while the addition in the right-hand side of the definition occurs in Y.

Technically this definition defines the right-derivative and we should also define the left-derivative using a backward difference and then say that the derivative exists only if these two values are the same. Normally we will deal with smooth mappings and thereby avoid this technicality.

[2] Little-o notation is useful as a container for arbitrarily complex expressions which vanish at a limit. For our purposes, $\mathcal{O}(g)$ is such that $\mathcal{O}(g)/g \to 0$ when $g \to 0$.

Notice what happens if $A$ is a linear operator. Then, we already know that

$$A(x+h) = Ax + Ah$$

and thus we see that $A'(x) = A$ for all x.

### B.1.2 Mappings from X to $\mathbb{R}$

Suppose X is an infinite-dimensional space (like the space of square-integrable functions) and the mapping is from X to the real numbers. This is what we have called a functional, but it lives happily in our general notion of a mapping from one normed vector space to another. Finding a derivative of a functional is sometimes called variational calculus, but again you can see that it is just an example of a single definition of derivative.

The general derivative definition says we should be able to define the (Fréchet) derivative of this functional with

$$\Psi(f+h) = \Psi(f) + \Psi'(f)h + \mathcal{O}(||h||)\,,$$

where $\Psi'(f)$ is some linear operator. How can we describe all the linear operators that map from X to $\mathbb{R}$. The simplest way is to choose a basis of X and tabulate what happens to the basis vectors through the linear operator. So, for example, write

$$f(\boldsymbol{x}) = \int \delta(\boldsymbol{x}-\boldsymbol{y}) f(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}$$

and we can see that the linear operator $P$ will map $f$ to the number

$$\begin{aligned} q &= Pf = \int P\delta(\boldsymbol{x}-\boldsymbol{y}) f(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \\ &= \int p(\boldsymbol{y}) f(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} \quad = \langle p, f \rangle \ , \end{aligned}$$

where $p(\boldsymbol{y}) = P\delta(\boldsymbol{x}-\boldsymbol{y})$ is the number that $P$ maps $\delta(\boldsymbol{x}-\boldsymbol{y})$ to, for a given $\boldsymbol{y}$. Thus, any linear operator, $P$ from X to $\mathbb{R}$ can be described using a function $p(\boldsymbol{y})$. As a result we could represent the derivative operator $\Psi'(f)$ using a function $\delta\Psi_f(\boldsymbol{y})$ with

$$\Psi'(f)h \equiv \int \delta\Psi_f(\boldsymbol{y}) h(\boldsymbol{y}) \mathrm{d}\boldsymbol{y}\,.$$

Therefore,

$$\Psi(f+h) = \Psi(f) + \int \delta\Psi_f(\boldsymbol{y}) h(\boldsymbol{y}) \mathrm{d}\boldsymbol{y} + \mathcal{O}(||h||)\,.$$

To find $\delta\Psi_f(\boldsymbol{y})$ we choose $h(\boldsymbol{y}) = \epsilon\delta(\boldsymbol{x}-\boldsymbol{y})$ and by taking limits note that

$$\begin{aligned} \delta\Psi_f(\boldsymbol{y}) &= \lim_{\epsilon\to 0} \frac{\Psi(f(\boldsymbol{x}) + \epsilon\delta(\boldsymbol{x}-\boldsymbol{y})) - \Psi(f(\boldsymbol{x}))}{\epsilon} \\ &\equiv \frac{\partial\Psi}{\partial f}\,. \end{aligned}$$

Notice that $\Psi'(f)$ represents the operator while $\delta\Psi_f(\boldsymbol{y})$ or simply $\delta\Psi$ represents the function that implements the operator so that

$$\Psi'(f)h = \langle \delta\Psi, h \rangle \ .$$

In calculations it is especially important to distinguish these notations.

### B.1.3 General Mappings

We have now shown how the definition of the (Fréchet) derivative of a mapping $A : \mathrm{X} \mapsto \mathrm{Y}$ can be defined as the linear operator satisfying the equation:

$$A(f+h) = A(f) + A'(f)h + \mathcal{O}(||h||) \qquad \text{as } ||h|| \to 0 \,.$$

We have seen how this general definition specializes for the more familiar mappings we are accustomed to and how the Fréchet derivative corresponds to our more familiar notions of derivative. We are now ready to get our minds around $A'(f)$ in the most general sense.

The first question we ask is how do we describe a general linear operator mapping X to Y. We already have a couple of representations. The first representation uses the kernel $K(\boldsymbol{x}; \boldsymbol{x}')$ so that the linear operator $P$ is described as

$$(Pf)(\boldsymbol{x}) = \int K(\boldsymbol{x}; \boldsymbol{x}') f(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}' \tag{B.3}$$

where $K(\boldsymbol{x}; \boldsymbol{x}')$ is the response of the operator to an impulse $\delta(\boldsymbol{x} - \boldsymbol{x}_0)$. We also have the spectral representation where we note that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_i w_i \boldsymbol{u}_i(\boldsymbol{x}) \boldsymbol{v}_i(\boldsymbol{x}')$$

so that

$$(Pf)(\boldsymbol{x}) = \sum_i w_i \left\langle f, \boldsymbol{v}_i \right\rangle u_i(\boldsymbol{x}) \,. \tag{B.4}$$

Thus, describing a general linear operator amounts to finding the kernel $K(\boldsymbol{x}; \boldsymbol{x}')$ and/or its spectral representation. Consequently, we could write the defining equation for the (Fréchet) derivative as

$$[A(f+h)](\boldsymbol{y}) = [A(f)](\boldsymbol{y}) + \int \delta A(\boldsymbol{y}; \boldsymbol{x}') h(\boldsymbol{x}') \mathrm{d}x' + \mathcal{O}(||h||) \,,$$

where $\delta A$ is the kernel of the linear operator $A'(f)$. By choosing $h(\boldsymbol{x}) = \epsilon\delta(\boldsymbol{x} - \boldsymbol{z})$ and taking limits we could find

$$\delta A(\boldsymbol{y}; \boldsymbol{z}) = \lim_{\epsilon \to 0} \frac{[A(f(\boldsymbol{x}) + \epsilon\delta(\boldsymbol{x} - \boldsymbol{z}))](\boldsymbol{y}) - [A(f(\boldsymbol{x}))](\boldsymbol{y})}{\epsilon} \,.$$

The Fréchet derivative is important largely for its role in optimization theory, but it also plays a role in inverse problems in general as it shows us the local linear behavior of an arbitrary non-linear

map. Consequently, the local SVD of the Fréchet derivative gives us a clear picture of the amount of information loss occurring for a particular input.

### B.1.4 Higher Order Derivatives

Higher order (Fréchet) derivatives are defined recursively so that $A''(x)$ is the linear operator mapping from X to Z (where Z is the space of all linear operators from $\boldsymbol{X}$ to $\boldsymbol{Y}$ ) such that

$$A'(x+k) = A'(x) + A''(x)k + \mathcal{O}(||h||)\,.$$

You can also think about $A''(x)$ as defining a bilinear map from X $\times$ X to Y so that $(h,k) \mapsto (A''(x)k)h$. It can be shown that this mapping is symmetric:

$$(A''(x)k)h = (A''(x)h)k\,.$$

We can continue this game and define a countably infinite number of higher-order Fréchet derivatives. For fun, try to come up with a general Taylor-series expansion of an arbitrary mapping in terms of these higher-order Fréchet derivatives. To be specific, try to show that we could represent an arbitrary mapping as

$$A(x+h) = A(x) + A'(x)h + c_2(A''(x)h)h + c_3((A'''(x)h)h)h + \ldots,$$

where $c_k = \frac{1}{k!}$.

### B.1.5 Calculus of FréchetDerivatives

**Example Rules** This definition as a limit is quite similar to the simple limit definition for the derivative. This means that all the familiar rules of calculus apply. A maximum or minimum (or in ection point) occurs when $\delta\Psi_f(\boldsymbol{y}) = 0$ and we can check between these by looking at $\Psi''$.

Other rules that are useful:

- Chain Rule: Suppose a new functional is $\Psi_1 = g(\Psi(f))$ then

$$\delta\Psi_1 = \delta g(\Psi(f)) = g'(\Psi(f))\delta\Psi$$

- Quotient rule

$$\delta\left[\frac{\Psi_1}{\Psi_2}\right] = \frac{\Psi_2\delta\Psi_1 - \Psi_1\delta\Psi_2}{\Psi_2^2}$$

- Product rule

$$\delta[\Psi_1\Psi_2] = \delta\Psi_1\Psi_2 + \Psi_1\delta\Psi_2$$

- Sum rule

$$\delta[a\Psi_1 + b\Psi_2] = a\delta\Psi_1 + b\delta\Psi_2$$

**Specific Derivatives** The derivative of a constant is zero

$$\delta a = 0\,.$$

In addition when $\Psi(f) = f(y)$ for a particular value of y then

$$\delta\Psi = \delta(y)\,.$$

When $\Psi(f) = \int f(x)g(x)\mathrm{d}x = \langle f, g\rangle$ we get

$$\begin{aligned}
\delta\Psi &= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\left[\int\left(f(x)g(x) + \epsilon\delta(x-y)g(x)\right)\mathrm{d}x - \int f(x)g(x)\mathrm{d}x\right] \\
&= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\int \epsilon\delta(x-y)g(x)\mathrm{d}x \\
&= g(y)\,.
\end{aligned}$$

Similarly, when

$$\Psi(f) = \int A(f(x))B(f(x))\mathrm{d}x = \langle A(f), B(f)\rangle \tag{B.5}$$

we get

$$\begin{aligned}
\delta\Psi &= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\left[\langle A(f+\epsilon\delta), B(f+\epsilon\delta)\rangle - \langle A(f), B(f)\rangle\right] \\
&= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\left[\langle A(f) + \epsilon A'(f)\delta, B(f) + \epsilon B'(f)\delta\rangle - \langle A(f), B(f)\rangle\right] \\
&= \langle A'(f)\delta, B(f)\rangle + \langle A(f), B'(f)\delta\rangle \\
&= \langle \delta, [A'(f)]^* B(f)\rangle + \langle [B'(f)]^* A(f), \delta\rangle \\
&= [A'(f)]^* B(f) + [B'(f)]^* A(f)\,.
\end{aligned} \tag{B.6}$$

Finally, when

$$\Psi(f) = \int g[f(x)]\mathrm{d}x$$

then

$$\begin{aligned}
\delta\Psi(y) &= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\int\left[g(f(x) + \epsilon\delta(x-y))\mathrm{d}x - \int g(f(x))\mathrm{d}x\right] \\
&= \lim_{\epsilon\to 0}\frac{1}{\epsilon}\left\{\int\left[g(f(x)) + \epsilon\delta(x-y)g'(f(x)) + \mathcal{O}(\epsilon) - g(f(x))\right]\mathrm{d}x\right\} \\
&= \int \delta(x-y)g'(f(x))\mathrm{d}x \\
&= g'[f(y)]\,.
\end{aligned}$$

Using just these rules quite a few calculations can be done.

**Example : Entropy** The Shannon entropy of a function is defined

$$\Psi_{\text{Ent}}(f) = \int (f \log f - f) \mathrm{d}x$$

using the above rules, e.g. Eq. B.6 we get

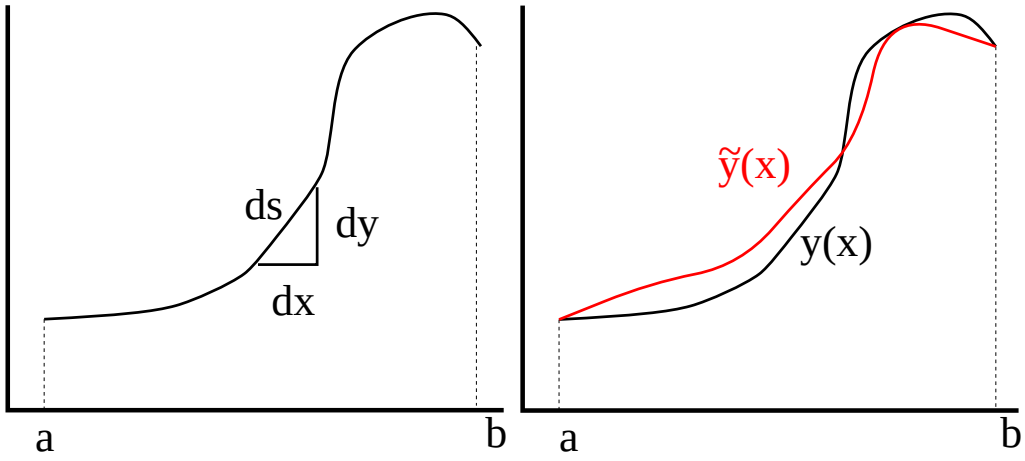
$$\delta\Psi_{\text{Ent},f} = \log f$$

So that the Fréchet derivative operator is given by

$$\Psi'(f)h = \int h \log f \mathrm{d}x \quad = \langle h, \log f \rangle$$

## B.2 Variational Forms and Euler-lagrange Equations

Whereas optimisation of functions (e.g. norms) over finite vector spaces may be quite familiar, it is also possible to consider infinite dimensional spaces of functions, and to look for the minimisation (or maximisation) of *functionals*.




Let us look for the stationary values (minimum, maximum, points of inlection) of the following functional

$$\Psi = \int_a^b \psi(y, y', x) \mathrm{d}x \tag{B.7}$$

Consider a perturbation from a given function $y(x)$ in an arbitary "direction" $h(x)$ having the same boundary conditions

$$\tilde{y}(x) := y(x) + \epsilon h(x)$$

where $h(a) = h(b) = 0$. If $\Psi$ is stationary, we define

$$\Psi(\epsilon) = \int_a^b \psi(\tilde{y}, \tilde{y}', x) \mathrm{d}x \tag{B.8}$$

and we require

$$\left.\frac{\mathrm{d}\Psi}{\mathrm{d}\epsilon}\right|_{\epsilon=0} = 0 \quad \forall h(x)\,. \tag{B.9}$$

If we write $\Psi(\epsilon)$ as a Taylor series we have

$$\begin{aligned}\Psi(\epsilon) &= \int_a^b \psi(y+\epsilon h, y'(x)+\epsilon h', x)\mathrm{d}x \\ &= \Psi(0) + \epsilon\int_a^b \left[\frac{\partial\psi}{\partial y}h + \frac{\partial\psi}{\partial y'}h'\right]\mathrm{d}x + \mathcal{O}(\epsilon^2)\end{aligned}$$

The requirement for stationarity is therefore

$$\Psi'(y)h = \int_a^b \left[\frac{\partial\psi}{\partial y}h + \frac{\partial\psi}{\partial y'}h'\right]\mathrm{d}x = 0\,.$$

By integration by parts and using $h(a) = h(b) = 0$ we find

$$\Psi'(y)h = \underbrace{\left[\frac{\partial\psi}{\partial y'}h\right]_a^b}_{=0} + \int_a^b \underbrace{\left[\frac{\partial\psi}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y'}\right]}_{=\delta\Psi} h\mathrm{d}x = \langle\delta\Psi, h\rangle = 0$$

and since this must be true for any $h$ we derive the *Euler-Lagrange* equation :

$$\delta\Psi := \frac{\partial\psi}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y'} = 0 \tag{B.10}$$

An expression such as *Eq. B.7* is known as a *Variational Form*, or sometimes as an *Energy* term because of its relation to principles such as Free Energy in classical thermodynamics. The Euler-Lagrange equation establishes an important principal in that optimisation of a variational form is always equivalent to solving a partial differential equation (PDE). A useful concept that we will see later, is that the solution to some (not all) PDEs can be made equivalent to optimising a variational form.

An alternative expression is sometimes useful :

$$\frac{\partial\psi}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\psi - y'\frac{\partial\psi}{\partial y'}\right) = 0\,. \tag{B.11}$$

To see this, expand all the derivatives in Eq. B.11 explicitly,

$$\begin{aligned}\frac{\partial\psi}{\partial x} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\psi - y'\frac{\partial\psi}{\partial y'}\right) &= \frac{\partial\psi}{\partial x} - \underbrace{\left(\frac{\partial\psi}{\partial x} + y'\frac{\partial\psi}{\partial y} + y''\frac{\partial\psi}{\partial y'}\right)}_{\frac{\mathrm{d}\psi}{\mathrm{d}x}} + \underbrace{\left(y''\frac{\partial\psi}{\partial y'} + y'\frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y'}\right)}_{\frac{\mathrm{d}}{\mathrm{d}x}y'\frac{\partial\psi}{\partial y'}} \\ &= -y'\left(\frac{\partial\psi}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y'}\right) = 0\end{aligned}$$

**E**xample 1 : Shortest path. What is the shortest path between two points? I.e. if we consider the path as a function $y(x)$ how do we find the one with minimum length ?

$$L = \int_a^b \mathrm{d}s = \int_a^b (\mathrm{d}x^2 + \mathrm{d}y^2)^{1/2} = \int_a^b \left(1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2\right)^{1/2} \mathrm{d}x \tag{B.12}$$

Obviously it is a straight line, but how do we actually prove that ? This leads us to the topic of *calculus of variations*.

For the example above, we have

$$\begin{aligned} \psi(y, y', x) &= \left(1 + y'^2\right)^{1/2} \\ \Rightarrow \quad \frac{\partial \psi}{\partial y'} &= \frac{y'}{(1 + y'^2)^{1/2}} \end{aligned}$$

From the Euler-Lagrange equation we have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial \psi}{\partial y'} &= 0 \\ \Rightarrow \quad \frac{\partial \psi}{\partial y'} &= \frac{y'}{(1 + y'^2)^{1/2}} = \text{constant} = A \end{aligned}$$

Rearranging,

$$y' = \frac{A}{(1 - A^2)^{1/2}} = m \Rightarrow \quad y = mx + c$$

and the values of the slope $m$ and offset $c$ are found from the boundary conditions. [Note also that $m = \frac{A}{(1-A^2)^{1/2}} \Rightarrow A = \cos\theta$].

**E**xample 2 : Light path in variable refractive index medium. Fermat's principle says that light follows the path that minimises the *time* spent in the medium

$$T = \frac{1}{c_0}\int_a^b n(x, y)(1 + y'^2)^{1/2}\mathrm{d}x \tag{B.13}$$

where $c_0$ is the speed of light in a vacuum. In this case we make use of the second Euler-Lagrange

form, Eq. B.11 to get

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}x}\left(\psi - y'\frac{\partial\psi}{\partial y'}\right) &= 0 \\ \Rightarrow \quad \left(\psi - y'\frac{\partial\psi}{\partial y'}\right) &= \text{constant} = K \\ \Rightarrow \quad n(x,y)(1+y'^2)^{1/2} - \frac{n(x,y)y'^2}{(1+y'^2)^{1/2}} &= K \\ \Rightarrow \quad \frac{n(x,y)}{(1+y'^2)^{1/2}} &= K \end{aligned}$$

For example if $n = e^y$ and $a = -1$, $b = 1$, then we can find the explicit solution

$$y(x) = \frac{1}{2}\log\left[1 + \tan(x)^2\right] + \text{constant} \tag{B.14}$$

where the constant is chosen to satisfy the boundary conditions

## B.3 Generalisations

We may generalise the basic variational method in several ways

**Higher order derivatives .** Suppose that

$$\Psi(y) = \int_a^b \psi(x, y, y', y'', \ldots y^{(n)})\mathrm{d}x\,. \tag{B.15}$$

We obtain

$$\frac{\delta\psi}{\delta y} = \frac{\partial\psi}{\partial y} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y'} + \frac{\mathrm{d}^2}{\mathrm{d}^2x}\frac{\partial\psi}{\partial y''}\ldots + (-1)^n\frac{\mathrm{d}^n}{\mathrm{d}^nx}\frac{\partial\psi}{\partial y^{(n)}} = 0 \tag{B.16}$$

**More than one dependent variable.** Suppose that the function sought is a vector function of one variable (e.g. a curve in space)

$$\Psi(y_1, y_2, \ldots y_n) = \int_a^b \psi(x, y_1, y_1', y_2, y_2', \ldots y_n, y_n')\mathrm{d}x = \int_a^b \psi(x, \boldsymbol{y}, \boldsymbol{y}')\mathrm{d}x \tag{B.17}$$

with specified boundary conditions $\boldsymbol{y}(a) = \boldsymbol{\alpha}$, $\boldsymbol{y}(b) = \boldsymbol{\beta}$, then the above procedure leads to the Euler-Lagrange equations

$$\frac{\delta\psi}{\delta y_i} = \frac{\partial\psi}{\partial y_i} - \frac{\mathrm{d}}{\mathrm{d}x}\frac{\partial\psi}{\partial y_i'} = 0 \tag{B.18}$$

**More than one independent variable.** Now consider a scalar function of several variables (e.g. a surface S)

$$\Psi(y) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \dots \int_{a_m}^{b_m} \psi(x_1, x_2, \dots x_m, y, y_{x_1}, y_{x_2}, \dots y_{x_m}) \mathrm{d}x_1 \mathrm{d}x_2 \dots \mathrm{d}x_m = \int_S \psi \left( \boldsymbol{x}, y, \frac{\partial y}{\partial \boldsymbol{x}} \right) \mathrm{d}^m \boldsymbol{x} \tag{B.19}$$

with specified boundary conditions $y(\boldsymbol{x})|_S = h$. The Euler-Lagrange equations are

$$\frac{\delta \psi}{\delta y} = \frac{\partial \psi}{\partial y} - \sum_{j=1}^{m} \frac{\mathrm{d}}{\mathrm{d}x_j} \frac{\partial \psi}{\partial y_{x_j}} = 0 \tag{B.20}$$

**Multivariate functions of multiple variables.** Finally, consider a vector function of a vector of variables

$$\Psi(y_1, y_2, \dots y_n) = \int_S \psi \left( \boldsymbol{x}, \boldsymbol{y}, \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \right) \mathrm{d}^m \boldsymbol{x} \tag{B.21}$$

with specified boundary conditions $y(\boldsymbol{x})|_S = \boldsymbol{h}$. The Euler-Lagrange equations are

$$\frac{\delta \psi}{\delta y_i} = \frac{\partial \psi}{\partial y_i} - \sum_{j=1}^{m} \frac{\mathrm{d}}{\mathrm{d}x_j} \frac{\partial \psi}{\partial y_{i,x_j}} = 0 \tag{B.22}$$

# References


[1] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, second edition, 1987.

[3] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, New York, 1999.

[4] S. Boyd and L. Vandenberghe. *Convex Optimisation*. Cambridge University Press, 2004.

[5] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, 2002.

[6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University, Cambridge, England, 2nd edition, 1992.

[7] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[8] M. Bertero and M. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing Ltd, 1998.

[9] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer, New York, 2005.