Fitting Functional Data Using Derivative-Free Methods

November 7, 2019

Abstract

Traditional algorithms for modelling functional data use derivative-based optimisation methods to fit parameters. The process of finding the derivatives of the fitting criterion with respect to the parameters is complex. In some cases, the derivatives might not even exist everywhere, as is the case when the Mean Absolute Deviation criterion is used instead of the usual Least Squares approach. Accordingly, the use of derivative-free methods for Functional Data Analysis was investigated in this thesis. It was found that the derivative-free methods perform satisfactorily on simple FDA problems and that the implementation effort was much less than for the derivative based methods. Furthermore, it is possible to fit models non-smooth loss functions such as the Mean Absolute Deviation criterion using derivative-free methods. It was also possible to fit a variety of parametric problems using a modified version of the derivative-free methodology developed in this thesis.

Contents

Chapter 1

Introduction

1.1 Preliminaries

1.1.1 Functional Data Analysis

Functional data analysis (FDA) is a field of statistics where it is assumed that the data observed at a given set of independent observation times (or coordinates etc.) represent noisy observations of some underlying function [?]. The approach taken here is to assume that an unknown differential equation can adequately, though not necessarily exactly, describe the process producing the data.

Specification of Function Spaces

The functions in question are assumed to be members of some countably infinite dimensional vector space, such as the set of all functions $f(\cdot)$ such that $\int_0^T |f''(t)|^2 dt < \infty$ over some interval $[0, T]$. This assumption implies that any given function can be represented as a countably infinite combination of basis elements, which are themselves functions. This means for a chosen set of basis elements $\{\phi_1(t), \phi_2(t), \dots\}$ and any given function $f(t)$, there is a set of coefficients $\{c_1, c_2, \dots\}$ such that:

$$f(t) = c_1\phi_1(t) + c_2\phi_2(t) + \dots.$$

Functional Data Analysis can thus be regarded as a generalisation of multivariate statistics where the number of dimensions is potentially infinite.Substantial complications are introduced into the statistical analysis because functions are much richer objects than real numbers or vectors. A function will generally have a different value for each input value, and the number of non-integer numbers on any interval (and hence potential inputs) is infinite. Functions therefore cannot be trivially represented on paper or in computer memory in a similar fashion as real numbers or vectors.

For the purposes of this thesis, it is assumed that the functions we are interested in are continuous mappings. In practice, the problem of potential infinite-dimensionality is resolved by constructing a discrete problem that resembles or approximates the functional problem, and then solving this approximate problem instead.

Statistical models that involve differential equations, such as those discussed in this thesis, are particularly difficult. A naive approach is to force the practitioner to solve the ordinary differential equation (ODE) numerically every time a goodness-of-fit is required for a given choice of parameters. For these situations, it is necessary by definition to use numerical analysis techniques to construct a proxy problem that (1) resembles the original problem sufficiently well, and (2) that is sufficiently easy to tackle computationally.

For example, consider the problem of parametric estimation for a stochastic differential equation (SDE) of the form:

$$dX = f(X;\theta)dt + \sigma dW.$$

Here $X(t)$ is the stochastic process being modelled, $f(\cdot;\theta)$ is a known function with a parameter θ to be estimated, σ is a volatility parameter, and $W(t)$ is a standard Brownian motion. This SDE is equivalent to asserting that, for any time t and increment h:

$$X(t+h) = X(t) + \int_t^{t+h} f(X(s);\theta)ds + \sigma[W(t+h) - W(t)].$$

Suppose there are observations $X_1, X_2, \dots, X_N$ of $X(t)$ at evenly spaced times, and that h is the distance between the time points. The integral formulation of the SDE suggests that if h is small enough, then

$$X_{k+1} \approx X_k + hf(X_k;\theta) + \sigma\sqrt{h}Z_k,$$

where the Z_k are i.i.d standard Normal random variables. The $\sqrt{h}$ term appears because $W(t+h) - W(t)$ has a variance of h. This is known as the *Euler-Maruyama Approximation* [?]. Thus, instead of attempting to estimate parameters for the original SDE, we can estimate parameters for a non-linear AR(1) process that acts as a proxy problem for the original SDE. This is a much more tractable problem than the original SDE. [1]

In FDA, the assumption is usually made that all the functions can be represented as a linear combination from some chosen *finite* set of basis functions. Rather than discretise the differential operator as in the above example, the space of functions is discretised instead. Similarly, a differential equation (or a similar problem) over some finite dimensional space of functions with n dimensions can be represented as a problem over the Euclidean space $\mathbb{R}^n$, which is now a discrete problem. The modelling process for functional data is described in Figure 1.1.

[1]Those interested in more detail are referred to [?].

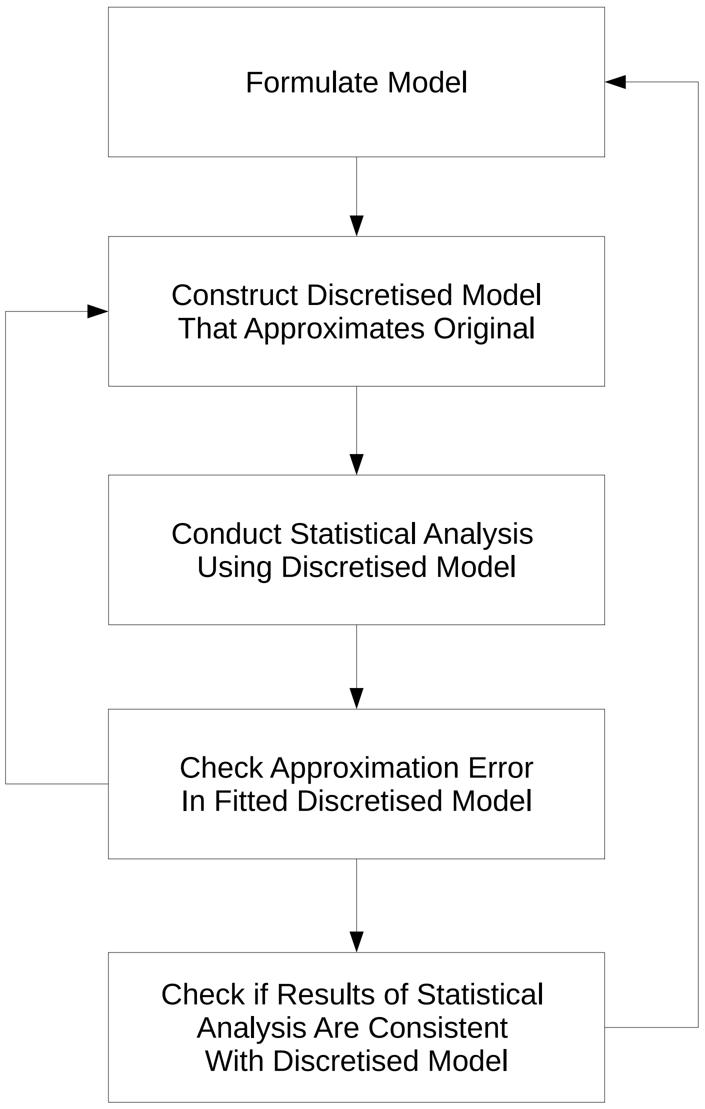

Formulate Model
Construct Discretised Model That Approximates Original
Conduct Statistical Analysis Using Discretised Model
Check Approximation Error In Fitted Discretised Model
Check if Results of Statistical Analysis Are Consistent With Discretised Model

Figure 1.1: Statistcal Modelling Process For Functions

Formulate a Model: As is the case for any statistical problem, the first step is to formulate a model. In the context of FDA, this often entails specifying an ODE model [? ?]. One must be certain that the model at used is sufficiently well-specified to capture the phenomenon under investigation.

Construct a Discretised Model that Approximates the Original Model: Unless the statistical model is trivial, the next step is to construct a proxy model.

Conduct Statistical Analysis Using the Discretised Model: While the discretised model tends to be simpler than the original model, this task is not necessarily trivial. For FDA problems, R packages such as the `FDA` package (discussed in Section 1.4) and the `Data2LD` package (discussing in Section 1.5) have been designed to conduct such analyses, however these packages can be complex to use.

Check the Approximation Error in Discretised Model: If the discretised model is too poor an approximation, then the results of any statistical analysis conducted could be biased as a result of the approximation error introduced. If the original model is biased, then the approximate one might be even more so.[2]

Therefore, conducting post hoc checks should be considered. For example, the analysis could be run again with an alternative approximate model and the results compared with the original model. If both analyses agree, it is evidence the approximate models are both reasonably accurate. In the context of FDA, this generally entails increasing the number of basis functions used to represent the functions so that the associated approximation error is smaller.

In an ODE context, suppose that the parameters of an ODE were estimated via least squares, and a finite difference solver was used to compute the fitted values, and hence determine the goodness-of-fit. Once the fitting algorithm had converged, the solver might be run again with a smaller stepsize (or more basis functions) with the same parameters and the goodness-of-fit statistic examined to determine if this had made a substantial change in the fit. If there has been a substantial change as a result of the stepsize reduction, the entire fitting procedure would need to be run again starting from the previously computed parameter estimate, now with the smaller stepsize. If reducing the stepsize a second time doesn't produce a substantial change in the goodness-of-fit statistic, one can be confident that no further reductions in the stepsize are necessary.

[2]A discussion of how numerical approximation error can introduce bias into parameter estimates for SDE models is provided in [?]. As a rule, the coarser the stepsize, the greater the asymptotic bias in parameter estimation for SDEs as the sample size goes to infinity.

This procedure can be automated. For example, the Implicit Filtering algorithm which is sometimes used for parametric ODE fitting and is discussed in Section B is an example of such automation. On each iteration, Implicit Filtering computes an approximate gradient using finite differences and uses this to perform optimisation [**?**]. If the Implicit Filtering algorithm cannot produce a decrease in the objective function, or it cannot be certain that the true gradient isn't in fact zero, it reduces the stepsize. The algorithm terminates when the change in the objective function between changes in the stepsize has fallen below a chosen tolerance level. However, such approaches can potentially be very slow due to the need to solve the same problem over and over again at increasing levels of precision.

Fortunately, FDA does not always require the re-computation of the curve in such a fashion whenever the parameters are changed. Instead of being implicitly represented as solutions of an ODE, functions are explicitly represented as elements in some finite dimensional vector space. As shall be seen, the objective function is generally a mapping from some vector space $\mathbb{R}^n$ to $\mathbb{R}$ that can often be evaluated more easily than running an ODE solver repeatedly as described above.

Check If Results of Statistical Analysis Are Consistent With Discretised Model. In the previous step, one checked that the approximate model was actually acting as a proxy for the original model. One must then check that the statistical analysis conducted using the approximate model is valid in its own right. For example, it will be seen throughout this thesis that many statistical problems involving functions can be approximated by non-linear regression models. These constructed non-linear regression models should be checked for statistical validity.

1.2 Exact Penalised Regression

Suppose we have N noisy observations y_i at times t_i, $i = 1, \ldots, n_i$ from some function $f(t)$, and we wish to estimate $f(t)$ from the data. One approach would be to estimate $f(t)$ by minimising a least squares criterion such as:

$$SSE(f) = \sum_{i=1}^{N} [y_i - f(t_i)]^2$$

Here, $SSE(\cdot)$ is a function that assigns a real number to every real-valued function that is defined for all the t_i. There is an obvious problem with this criterion - it does not have a unique minimiser. Any function $g(t)$ such that $g(t_i) = y_i$ will minimise $SSE(\cdot)$. Therefore there are an infinite number of degrees of freedom, but only a finite number of observations. To ensure uniqueness and to choose between different functions that interpolate a given set of points, it is necessary to impose further conditions to discriminate between different candidate functions.

1.2.1 Smoothing Splines

One potential criterion is to introduce a second order penalty, i.e. penalise the curvature of the estimated function $f(t)$. Introducing this penalty will ensure that if two functions fit the observed data equally well, the more regular or less 'wiggly' function is chosen. There are several ways of translating this intuition into a formal fitting procedure. A common choice is to measure the degree of irregularity (or 'wiggliness') by using the integral of the second derivative over a chosen interval $[0, T]$, with the upper limit T chosen to allow for all observation times to be included. Thus the penalty is of the form:

$$\int_0^T |f''(t)|^2 dt.$$

For a given set of points, the smooth interpolating curve that minimises the integral above is given by an interpolating cubic spline. Choosing the most regular interpolating curve is not necessarily a very good estimation strategy however, because it strongly prioritises goodness-of-fit above all other considerations. If the data is noisy, there is a risk of over-fitting and poor predictive power. Hence there is a trade-off between bias and variance and a joint estimation strategy is pursued to find a good balance between fidelity to the observed data and reasonably regular behaviour. This involves minimising the following penalised least squares criterion:

$$PENSSE(f; \lambda) = \sum_{i=1}^{N} [y_i - f(t_i)]^2 + \lambda \int_0^T |f''(t)|^2 dt \tag{1.1}$$

The λ term dictates the trade-off between fidelity to the data and regularity. As λ tends towards zero, the fitted curve gets closer to interpolating the data. As λ gets larger, priority is given to smoothness. λ can be chosen via cross-validation or generalised cross-validation.

Suppose there were a candidate function $g(t)$, then by taking the cubic spline such that its value at t_i is equal to $g(t_i)$, we can produce a curve $s(t)$ that has the same least-squares error as $g(t)$, but with $\int [s''(t)]^2 dt \leq \int [g''(t)]^2 dt$. Thus, the curve that minimises $PENSSE$ in Equation (1.1) can be assumed to be a cubic spline. To find the minimiser of $PENSSE(\cdot;\lambda)$, first assume that $f(t)$ can be represented as a linear combination of K cubic spline functions, $\phi_k(t)$, $k = 1, \ldots, K$, that can represent any cubic spline with knots at the t_i. This implies that

$$f(t) = \sum_{k=1}^{K} c_k \phi_k(t).$$

Next let the design matrix $\mathbf{\Phi}$ be defined by $\mathbf{\Phi}_{kj} = \phi_k(t_j)$, where k indexes the basis functions and j indexes the observations. Let the weight matrix $\mathbf{R}$ be defined by $\mathbf{R}_{kj} = \int_0^T \phi_k''(t)\phi_j''(t) dt$. Then $PENSSE$ can be written in terms of the vector of coefficients $\mathbf{c}$ and observations $\mathbf{y}$ as:

$$PENSSE(\mathbf{c};\lambda) = \|\mathbf{y} - \mathbf{\Phi}\mathbf{c}\|^2 + \lambda \mathbf{c}^\top \mathbf{R}\mathbf{c},$$

and the problem of minimising Equation (1.1) has been replaced with a discretised problem over $\mathbb{R}^K$. The optimal value of $\mathbf{c}$ is given by

$$\hat{\mathbf{c}} = (\mathbf{\Phi}'\mathbf{\Phi} + \lambda\mathbf{R})^{-1}\mathbf{\Phi}^\top \mathbf{y}.$$

This is an exact solution to the original problem because the span of the $\{\phi_k(t)\}$ contains the function that minimises $PENSSE$. The coefficient vector $\hat{\mathbf{c}}$ is the set of coordinates of the optimal function within this finite-dimensional vector space.

1.2.2 Piecewise Trigonometric Interpolation

The penalised regression approach can be extended to include more general penalty terms. Consider a more difficult penalised regression problem of the form:

$$PENSSE(f;\lambda) = \sum_{i=1}^{N} [y_i - f(t_i)]^2 + \lambda \int_0^T |f''(t) - f(t)|^2 dt \qquad (1.2)$$

The penalty $f''(t)$ has now been replaced with the more complex penalty, $f''(t) - f(t)$. $PENSSE$ can be minimised in this case taking by a piecewise function consisting of linear combinations of $\sin(t)$ and $\cos(t)$ pairs over each interval between t_i and t_{i+1}, and requiring that the points (y_i, t_i) are interpolated exactly such that $f(t_i) = y_i$, and that this piecewise function must be continuous.

Since a function of the form

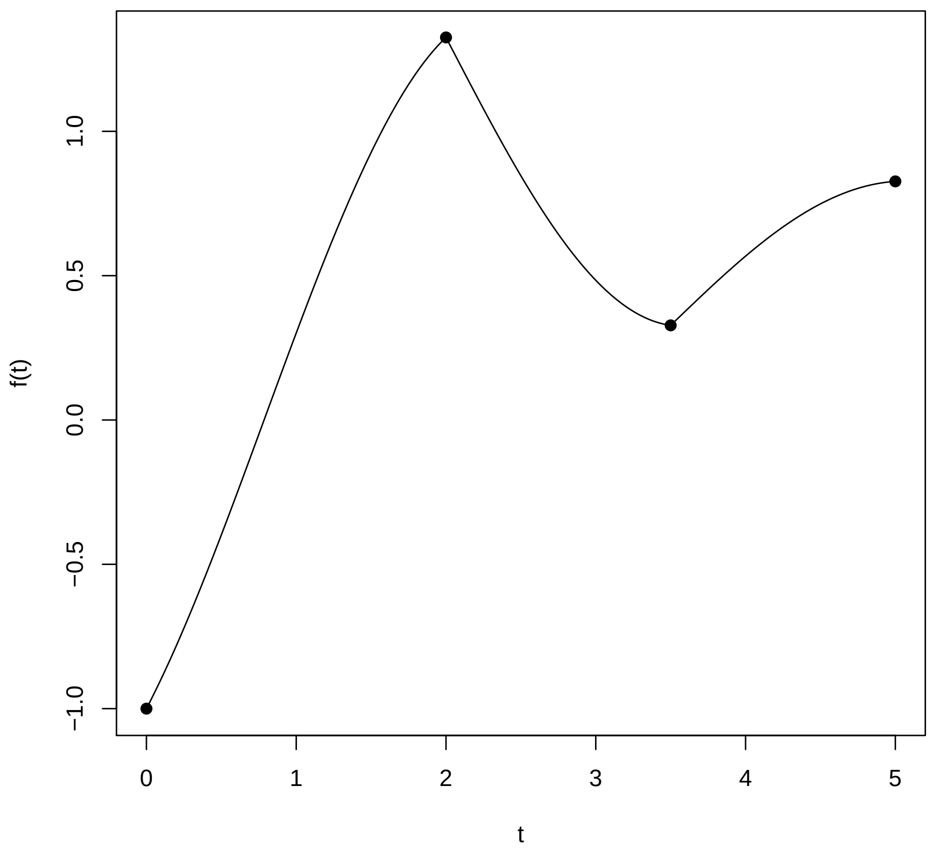


Figure 1.2: Plot of a Piecewise Trigonometric Curve. Note the kinks between segments.

$$a_0 + a_1 \cos(t) + b_1 \sin(t) + a_2 \cos(2t) + b_2 \sin(2t) + \dots$$

can be written as a polynomial in e^{it} and e^{-it}, such a piecewise trigonometric function can also be referred to as a piecewise trigonometric polynomial or a piecewise trigonometric spline [**?**].

As can be seen in Figure 1.2, a piecewise trigonometric polynomial of second degree generally fails to be smooth at the boundary points, and thus has a kinked appearance. For the purposes of statistical modelling, it is strongly desirable to impose the additional constraint that $f(t)$ must be everywhere differentiable. However this cannot be achieved for a piecewise basis formed from the functions $\{\sin(t), \cos(t)\}$ because there are only two free parameters on each segment and they are needed to ensure continuity.

1.3 Penalised Regression Using Finite Dimensional Approximations

In the previous section, the two problems in Sections 1.2.1 and 1.2.2 were solved exactly by constructing a finite dimensional function space that contained the optimal function. However, coonstructing such a space of functions is not always possible. In practice, one would hope that the optimal function can be approximated sufficiently well by a linear combination of some chosen set of basis functions. Spline bases functions are effectively the default choice as they provide a good balance between being well-behaved and having good approximating power.

Functional Data Analysis thus consists of the following steps, as illustrated in Figures 1.3 and 1.4:

1. Formulate a model for $f(t)$. Usually, this takes the form of a penalised regression model, where $f(t)$ is defined as the function that minimises some kind of penalised error such as in Equations (1.1) and (1.2).

2. Assume that $f(t)$ can be written as a finite combination of chosen basis functions. In practice a finite basis can only ever approximate $f(t)$, so it is important to ensure the basis (i.e. K) is large enough to approximate the optimal $f(t)$ sufficiently well. The function $f(t)$ can thus be written:

$$f(t) = \sum_{i=1}^{K} c_i \phi_i(t) = \mathbf{c}^\top \boldsymbol{\phi}(t),$$

   such that $f(t)$ is now defined by the coefficient vector $\mathbf{c}$.

3. Formulate the statistical model in terms of the coefficient vector $\mathbf{c}$. A statistical problem over a given (infinite-dimensional) functional space has thus been transformed into a statistical problem over the (finite-dimensional) $\mathbb{R}^K$ space.

For every valid choice of $\mathbf{c}$, a statistic that measures the goodness-of-fit to the data can be computed, and the value of $\mathbf{c}$ that maximises the goodness-of-fit statistic is required. The problem of finding the coefficient vector $\mathbf{c}$ can thus be thought of as estimation of a non-linear regression problem since $\mathbf{c}$ is finite-dimensional. Besides formulating an FDA model, one needs to consider the questions of constructing a finite dimensional approximation and then solving the associated non-linear regression. The situation is sketched in Figure 1.4.

1.3.1 FDA With a Quadratic Basis

As carried out in [**?**], we will provide an example with a very small basis to illustrate these key steps. Consider the following penalised regression problem:

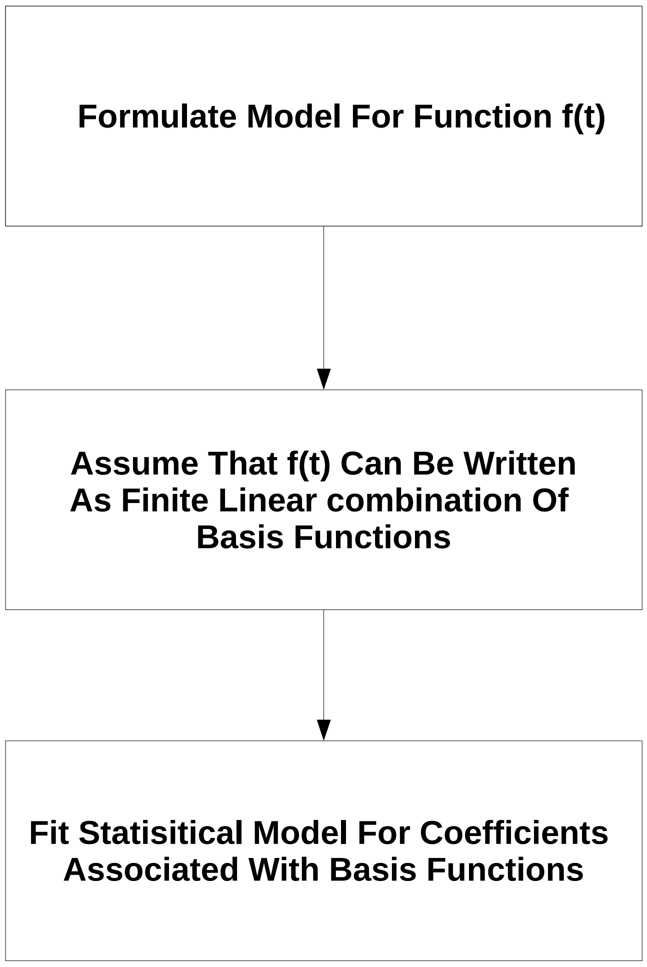

Formulate Model For Function f(t)
Assume That f(t) Can Be Written As Finite Linear combination Of Basis Functions
Fit Statisitical Model For Coefficients Associated With Basis Functions

Figure 1.3: Statistical Modelling Process For Functional Data Analysis

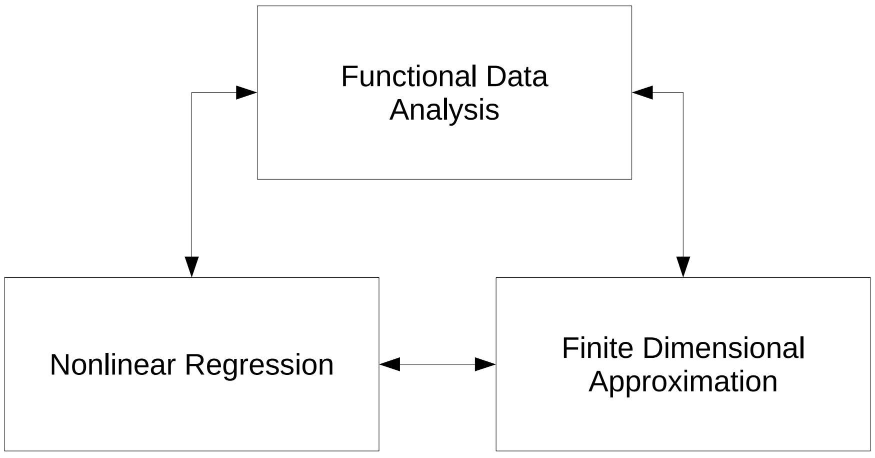


Figure 1.4: Elements of Functional Data Analysis

$$PENSSE(f;\lambda) = \sum_{i=1}^{N}[y_i - f(t_i)]^2 + \lambda \int_0^1 |t^2 f'' - 0.5f|^2 dt$$

The differential equation associated with the penalty term is known as an Euler's Equation. The solution is given by $f(t) = at^{r_1} + bt^{r_2}$, where r_1 and r_2 are the roots of the quadratic equation $r^2 - r - 0.5 = 0$. Thus, $r_1 \approx -0.36$ and $r_2 \approx 1.36$. For the sake of illustration it will be assumed that that $f(t)$ can be written as a quadratic using a linear combination of the basis functions $\{1, t, t^2\}$, i.e.

$$f(t) = at^2 + bt + c$$

Then:

$$\begin{aligned}
\int_0^1 |t^2 f'' - 0.5f|^2 dt &= \int_0^1 \left| at^2 - \frac{1}{2}(at^2 + bt + c) \right|^2 dt \\
&= \int_0^1 \left| \frac{1}{2}\left(at^2 - bt - c\right) \right|^2 dt \\
&= \frac{1}{4}\int_0^1 |at^2 - bt - c|^2 dt \\
&= \frac{1}{4}[a \ \ -b \ \ -c]^\top \mathbf{H}[a \ \ -b \ \ -c] \\
&= \frac{1}{4}[a \ b \ c]^\top (\mathbf{A}'\mathbf{H}\mathbf{A})[a \ b \ c] \\
&= [a \ b \ c]^\top \mathbf{K}[a \ b \ c]
\end{aligned}$$

Here $\mathbf{K} = \frac{1}{4}\mathbf{A}'\mathbf{H}\mathbf{A}$, the elements of the matrix $\mathbf{H}$ are defined by $\mathbf{H}_{ij} = \int_0^1 t^i t^j dt = 1/(i + j + 1)$, and elements of the matrix $\mathbf{A}$ are given by:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Thus, the penalised error is given by:

$$PENSSE(a, b, c; \lambda) = \sum_{i=1}^{N}(y_i - at_i^2 - bt_i - c)^2 + \lambda[a \ b \ c]^\top \mathbf{K}[a \ b \ c] \qquad (1.3)$$

We have therefore changed our estimation problem from a problem specified in terms of functions, to a penalised least squares problem in the three coefficients a, b and c. The quality of this approximate model as λ gets larger and larger depends on how well the functions $t^{-0.36}$ and $t^{1.36}$ can be respectively approximated by quadratics over the interval $[0, 1]$.

To illustrate this example further, the above approach was fitted to simulated data. A solution to the ODE $t^2 f'' - f = 0$ was generated over the interval $[0, 1]$, and samples were taken at various points before being corrupted by Gaussian noise. The quadratic that minimised Equation (1.3) with $\lambda = 100$ was then found. For comparison, the data was also fitted to a quadratic using ordinary least squares. The original function $f(t)$, the perturbed data, and the two fitted functions are all shown in Figure 1.5. As has already been noted, the quality of this model depends on how well $f(t)$ can be approximated by a quadratic over $[0, 1]$. Therefore, the quadratic $q(t)$ that minimises $\int_0^1 |f(t) - q(t)|dt$ was found numerically and is also plotted in Figure 1.5.

Figure 1.5 suggests that $f(t)$ can be approximated reasonably well by quadratics so long as one stays away from the point $t = 0$. The ODE $t^2 f'' - f = 0$ behaves degenerately at the origin. When $t = 0$, the ODE has a singular point, the term in front of f'' becomes zero so that the ODE reduces to $(0)^2 f'' - f = 0$. Additionally, it is always the case that the second derivative diverges to infinity at 0 if $f(t)$ is of the form $at^{-0.36} + bt^{1.36}$. As a result of both the singular point and infinite curvature at $t = 0$, polynomial approximation is predicted to be exceptionally tricky around this point [**?** **?**].

Comparing the two fits in Figure 1.5, the penalised regression model captures the shape of $f(t)$ better than ordinary least squares away from $t = 0$. Both models seem to have similar predictive power on average. However, the penalised fit is being heavily influenced by the singularity at $t = 0$, and fails to capture the behaviour of the original data close to this point.

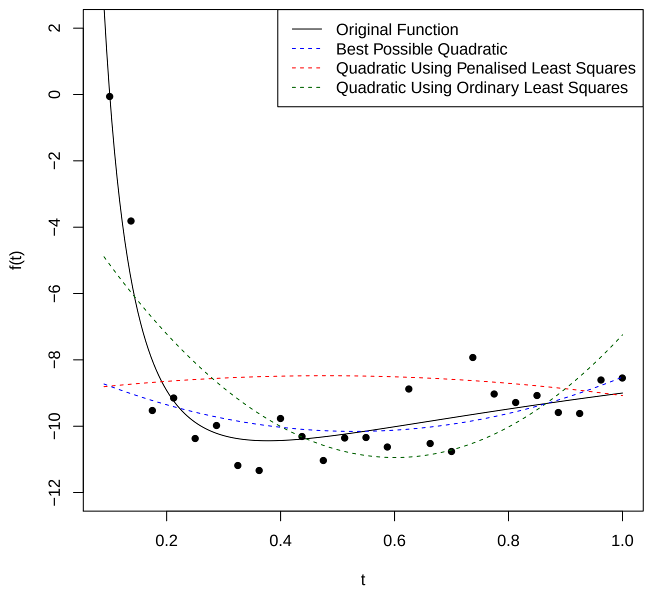


Figure 1.5: Performing FDA with the differential operator $Lf = t^2 f'' - 0.5f$ and the basis set $\{1, t, t^2\}$.

1.4 The `FDA` Package

Section 1.3 developed FDA algorithms for penalised fitting from first principles. However the `FDA` package in R ([? ?]) was developed to tackle penalised problems of the form:

$$PENSSE(f) = \sum_{i=1}^{N} [y_i - f(t_i)]^2 + \lambda \int |Lf(t)|^2 dt \tag{1.4}$$

Here Lf is a parameterised linear differential operator of the form $\sum_{j=0}^{n} \beta_j D^j$ where the β_j are constants. The authors of the `FDA` package introduce the melanoma dataset, which describes the incidence of skin cancer per 100,000 people in the state of Connecticut from 1936 to 1972 [?]. The result of smoothing the melanoma data with the differential operator $Lf = f - \omega^2 f^{(4)}$ with $\omega = 0.65$ is shown in Figure 1.6. This operator was chosen because a penalty of the form $f - \omega^2 f^{(4)}$ ignores functions of the form $c_1 + c_2 t + c_3 \sin(\omega t) + c_4 \cos(\omega t)$.

The `FDA` packages is not as powerful as the `Data2LD` package, which will be introduced in Section 1.5. It has the advantage of simplicity and ease of use, and is used throughout this thesis to fit FDA models unless `Data2LD` is essential. A deficiency of the `FDA` package is that it provides no guidance on the best choice of the ODE parameters β_j

nor the smoothing parameter λ. [3]

[3]The FDA package has a command called `lambda2gcv` whose documentation claims it '[finds] the smoothing parameter that minimises GCV' ([**?**]). Inspection of the code for this function shows that it only performs a fit based on the value of λ passed and then reports the GCV.

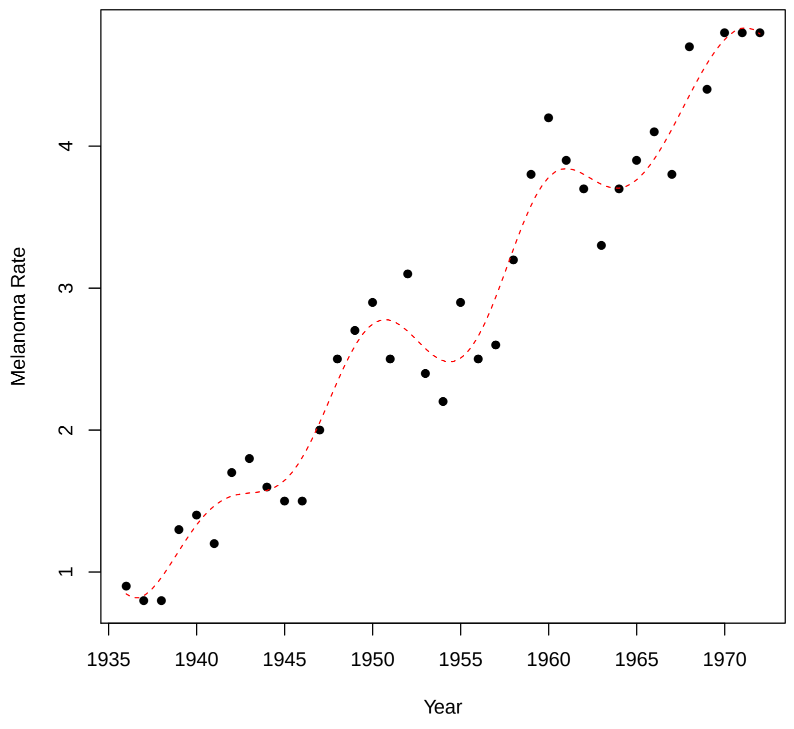


Figure 1.6: Using the `FDA` package to smooth the melanoma data with the differential operator $Lf = f - \omega^2 f^{(4)}$.

1.5 The `Data2LD` Package

The `Data2LD` package is an R package intended to perform smoothing using general linear differential operators with a forcing function, that is, ODEs of the form:

$$\sum \beta_i(t) D^i f(t) = u(t) \tag{1.5}$$

The $\beta_j(t)$ are parameter functions for the linear differential operator on the left hand side, and $u(t)$ is a forcing function. More generally, `Data2LD` can model a system of inhomogeneous linear differential equations:

$$\frac{d\mathbf{f}(t)}{dt} + \mathbf{B}(t)\mathbf{f}(t) = \mathbf{u}(t) \tag{1.6}$$

Each element of $\mathbf{B}(t)$ is a time-varying linear parameter function of the the form $\beta_{ij}(t)$ and each element of $\mathbf{u}(t)$ denotes the forcing function applied to the ith equation. A further advantage of `Data2LD` over the `FDA` package is that not only can it smooth ODEs with functional parameters, but it can estimate the associated parameters even if they are functions.

While `Data2LD` can estimate parameters for the differential operator, it does not provide a means for finding the optimal smoothing parameter λ.

1.6 Parametric estimation of ODE parameters versus `Data2LD`

The reflux data, plotted in Figure 1.7, describes the output of an oil refining system. A given fraction of oil is distilled into a specific tray, at which point it flows out through a valve. At a given time, the valve is switched off, and distillate starts to accumulate in the tray [**?**]. The reflux dataset was taken from the `Data2LD` package used for FDA mentioned above. The authors of the `Data2LD` package model the data using the following ODE:

$$\begin{cases} y'(t) = -\beta y(t) & t \le t_0 \\ y'(t) = -\beta y(t) + u_0 & t \ge t_0 \\ y(0) = 0 \end{cases} \tag{1.7}$$

Up to the point t_0, the function satisfies the ODE $y' = -\beta y$. At the breakpoint, a constant forcing function u_0 is turned on to model the valve being switched off, so that the ODE then becomes $y' = -\beta y + u_0$. This ODE admits an exact solution. Letting $\gamma = -u_0/\beta$ and C be an arbitrary constant, then the solution is given by:

$$y(t) = \begin{cases} 0 & t < t_0 \\ \gamma + Ce^{-\beta(t-t_0)} & t \ge t_0 \end{cases}$$

Without loss of generality the exponential term $Ce^{-\beta(t-t_0)}$ can be replaced with one that is of the form $Ce^{-\beta t}$. This is because $Ce^{-\beta(t-t_0)} = Ce^{-\beta t}e^{-\beta t_0} = [Ce^{-\beta t_0}]e^{-\beta t}$, and the $e^{-\beta t_0}$ term is thus absorbed into the constant term. In order to ensure that $y(t)$ is continuous at t_0 and monotone increasing, we require that $\gamma + C = 0$ and that $\beta > 0$.

1.6.1 Fitting the Data Parametrically by Solving the ODE Model

Instead of approximately solving an associated problem as discussed in Section 1.3, a purely parametric approach to fitting the ODE in Equation (1.7) will be employed, while the question of modelling the reflux data using the FDA approach will be discussed in the next chapter.

On closer inspection, the constraint $C = -\gamma$ is unsuitable for numerical parameter estimation because R's `nls` command reports errors when this constraint is imposed. However, if we allow t_0 to vary, we can allow C to assume any negative value while preserving monotonicity and continuity. Thus, let us assume that $y(t)$ is instead given by an alternative formulation of the problem such that:

$$\tilde{y}(t) = \max(0, \gamma + Ce^{-\beta(t-t_0)})$$

This change does not substantially effect the statistical model. The function $\tilde{y}(t)$ satisfies the same ODE and initial conditions as $y(t)$ except that the change point t_0 is shifted to t_0' which is defined by:

$$t_0' = \max\left(t_0, t_0 - \frac{1}{\beta}\ln\left(\frac{-\gamma}{C}\right)\right)$$

When the model is fitted, t_0' and t_0 are very close to each other in practice. The function $\tilde{y}(t)$ is a combination of two simpler functions, joined together using the maximum operator instead of the addition operator, an idealised version of which can be seen in Figure 1.8.

Parametric Fitting

We will now use this approach to model the reflux data. Assume that the breakpoint t_0 is known in advance. Then the model for $y(t)$ is:

$$y(t) = \begin{cases} 0 & t \leq t_0 \\ \beta_0 + \beta_1 e^{\beta_2 t} & t \geq t_0 \end{cases} \tag{1.8}$$

Note that this function might not be well defined at t_0, and requires matching of two functions at that point to ensure a continuous solution. How to carry out this matching will addressed later. It will not generally be the case that $\beta_0 + \beta_1 e^{\beta_2 t_0} = 0$, so ultimately a model of the form $y(t) = \max(0, \beta_0 + \beta_1 e^{\beta_2 t})$ will be used to ensure continuity at the breakpoint. We must estimate the three unknown coefficients $\beta_0, \beta_1, \beta_2$.

Estimating β_0 from the data: Figure 1.7 suggests that $\beta_2 < 0$, and $\beta_1 < 0$. Under this assumption, we have that:

$$\lim_{t \to \infty} y(t) = \beta_0,$$

where the convergence happens monotonically from below. An initial estimate for β_0 is thus given by $\hat{\beta}_0 = \max(y_i)$.

Estimating β_1 and β_2 from β_0 and the data: For $t \geq t_0$, the model in Equation 1.8 can be rearranged so that:

$$\log[\beta_0 - y(t)] = \log|\beta_1| + \beta_2 t \tag{1.9}$$

This equation is only valid as long as the left hand side is well defined however. It is necessary to exclude the largest observed value of y, because β_0 is estimated to be the largest observation at this point. If the largest value were included, there would be a term of the form $\log(0)$ in the rearranged model. The values of $\log|\beta_1|$ and β_2 can be estimated by performing simple linear regression of t against $\log[\beta_0 - y(t)]$, with the largest value of y observed excluded. It was assumed that $\beta_1 < 0$, so $\hat{\beta}_1$ can be found from the estimate of $\log|\beta_1|$.

Simultaneous Estimation of Parameters: Now that we have reasonable estimates for β_0, β_1, and β_2, we can use non linear regression to estimate all three jointly.

Matching: For $t < t_0$, it is estimated that $\hat{y}(t) = 0$. For $t \geq t_0$, the estimate is given by $\hat{y}(t) = \hat{\beta}_0 + \hat{\beta}_1 e^{\hat{\beta}_2 t}$. There are distinct estimates for $y(t)$ at $t \leq t_0$ and $t \geq t_0$, which do not necessarily agree at $t = t_0$. This is the case for the estimates produced here since $\hat{y}(t_0) = 0.029$. To stitch the two functions together, let $\hat{y}(t) = \max(0, \hat{\beta}_0 + \hat{\beta}_1 e^{\hat{\beta}_2 t})$. This is a continuous function that entirely satisfies the original ODE, except for the precise location of the breakpoint. The resulting fit is presented in Figure 1.9.

Breakpoint Estimation: The value of t_0 used for the fit is given by $t_0 = 68$. A statistical estimate of the breakpoint can be found from finding the point where $\hat{\beta}_0 + \hat{\beta}_1 e^{\hat{\beta}_2 t}$ is zero:

$$\hat{t_0} = \left| \frac{1}{\hat{\beta}_2} \log\left(-\frac{\hat{\beta}_0}{\hat{\beta}_1} \right) \right|.$$

Using this formula, it was estimated that $t_0 = 67.71$. This new value will produce the same results as for $t_0 = 68$ because it doesn't change the set of observation points used to estimate β_0, β_1, and β_2.

Discussion

The parametric approach taken to estimation here is somewhat *ad hoc*. Instead of devising a formal estimation strategy in advance, the fitting approach evolved organically alongside the problems of solving the ODE and fitting the data. Use was made of properties unique to the specific ODE model to compute estimates. While this has produced an effective fit, there are obvious concerns about generalising this approach to other ODEs. The issue is difficult to resolve if we restrict ourselves to solving ODE models explicitly and then fitting them by parametric methods. It is often the case in applied mathematics that one can't fully investigate an ODE model until one has a rough grasp of its behaviour. It has been demonstrated that the associated statistical fitting problem inherits this tendency.

1.6.2 Fitting the Reflux Data with `Data2LD`

While the parametric approach requires a considerable amount of domain-specific knowledge or solving complex constrained optimisation problems, the functional data analysis approach can be more generally employed. The FDA approach doesn't rely on individual features of the specific differential equation at hand,[4] and produces a similar fit to the reflux data as the parametric approach in Section 1.6.1.

The functional model asserts that

$$y'(t) \approx -\beta y(t) + u(t),$$

where $y(\cdot)$ and $u(\cdot)$ are functions to be estimated, and β is a single scalar parameter. It is assumed that $u(t)$ is a step function of the form

$$u(t) = a\mathbb{I}_{[0,t_0)}(t) + b\mathbb{I}_{[t_0,\infty)}(t).$$

[4]The FDA approach does rely on more general features of course, such as whether or not the differential equation is linear.

As in the parametric case, the breakpoint t_0 is fixed in advance. It is further assumed that $y(t)$ can be expanded as a linear combination of B-Splines. The knots are duplicated at t_0 so that the first derivative at the breakpoint is discontinuous. This model was fitted using the `Data2LD` package, and the results are plotted in Figure 1.10. It can be seen that the fit is quite similar to the parametric one presented in Figure 1.9. The main disadvantage of the FDA approach compared to the parametric one is that `Data2LD` can be complex and unintuitive to use.

1.7 Rates Of Convergence

Throughout this thesis, it will sometimes be desirable to consider the rates of convergence of different fitting and estimation methods. For the purposes of this section, it is assumed that there is a vector-valued sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ that converges to a value $\mathbf{x}^*$.

Linear Convergence: A convergent sequence is said to *converge linearly*[5] to $\mathbf{x}^*$ (with convergence rate μ) if there is a $0 < \mu < 1$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = \mu. \tag{1.10}$$

If a sequence $\mathbf{x}_n$ converges linearly with constant μ, then $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \approx \mu\|\mathbf{x}_n - \mathbf{x}^*\|$ for n sufficiently large. A simple example of a linearly converging sequence is given by $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$ If plotted on a log scale, the error terms $\|\mathbf{x}_{n+1} - \mathbf{x}^*\|$ will tend to lie on a straight line. A linearly convergent sequence has the property that if the number of iterations is doubled, then the number of digits of precision achieved is roughly doubled as well.

Sublinear Convergence: A sequence is said to converge *sublinearly* to $\mathbf{x}^*$ if:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = 1$$

Sublinear convergence is very slow. Every reduction in the order of magnitude of the error achieved takes more iterations than the previous reduction. The ur-example of a sublinearly convergent sequence is the reciprocals of the natural numbers: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots$.

Superlinear and Quadratic Convergence: A sequence is said to converge superlinearly if:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = 0,$$

[5] In [**?**], the case $\mu = 0$ is considered to be a case of linear convergence as well. This definition makes it harder to sharply discriminate between linear and superlinear convergence.

Convergence Class	Example	Iterations until $< 10^{-6}$	Iterations until $< 10^{-12}$
Sublinear	$x_n = \frac{1}{n}$	$10^6 + 1$	$10^{12} + 1$
Linear	$x_n = 2^{-n}$	20	40
Superlinear	$x_n = 2^{-2^n}$	5	6

Table 1.1: Illustrating the different classes of convergence.

and is said to converge superlinearly with order p if there exist positive constants $p > 1$ and $\mu > 0$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|^p} = \mu. \tag{1.11}$$

If $p = 2$, the sequence is said to converge quadratically. Note that there is no requirement that $\mu < 1$ in this case. The μ term will be dominated if $\|\mathbf{x}_n - \mathbf{x}^*\|$ is small enough in magnitude. Taking logs of Equation (1.11) yields that if n is sufficiently large, then:

$$\log(\|\mathbf{x}_{n+1} - \mathbf{x}^*\|) \approx \log(\mu) + p\log(\|\mathbf{x}_n - \mathbf{x}^*\|)$$

If $\|\mathbf{x}_n - \mathbf{x}^*\| < 1$ then $\log(\|\mathbf{x}_n - \mathbf{x}^*\|) < 0$. As already indicated in the previous paragraph, the $\log(\mu)$ term will be become increasingly negligible as the error $\|\mathbf{x}_n - \mathbf{x}^*\|$ becomes smaller and smaller.

For a linearly convergent sequence, the magnitude of the error declines exponentially, and the number of digits of precision gained increases linearly with the number of iterations. But for a superlinearly convergent sequence, the *order of magnitude of the error* declines exponentially, and the number of digits of precision gained grows geometrically with the number of iterations. For a quadratically converging sequence, each iteration tends to roughly double the number of digits of precision. For example, if the error in the first iterate is approximately 0.1, the next iterate will have error on the order of 10^{-2}, the next again will have error on the order of 10^{-4}, and so on.

Note that if $\log(\mu) + p\log(\|\mathbf{x}_n - \mathbf{x}^*\|)$ is large enough then $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| > \|\mathbf{x}_n - \mathbf{x}^*\|$, so that there is a failure to converge.[6] Methods that converge superlinearly are generally more sensitive to a poor starting point than methods that converge linearly. An example of superlinear convergence is given by the sequence $x_n = 2^{-2^n}$.

An Extended Definition of Convergence Rates: The above approach to defining rates of convergence can't handle every sequence however. For example, the sequence $1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \ldots$ does not converge linearly in the sense of (1.10). To cover these situations, a sequence is also said to converge linearly/sublinearly/superlinearly if there

[6]Consider for example the case where $\mu = 1, p = 2$, and $\|\mathbf{x}_0 - \mathbf{x}^*\| = 2$. It will be the case that $\|\mathbf{x}_1 - \mathbf{x}^*\| = 4$ and $\|\mathbf{x}_2 - \mathbf{x}^*\| = 16$.

is an associated auxiliary sequence ϵ_n such that $\|\mathbf{x}_n - \mathbf{x}^*\| \leq \epsilon_n$ for all $n \geq 0$, and the sequence ϵ_n converges linearly/sublinearly/superlinearly to zero.[7]

Linear Convergence and Iterated Mappings: Nearly all estimation algorithms used in statistics start with an initial estimate θ_0 and generate a sequence of estimates by $\theta_{n+1} = \mathbf{M}(\theta_n)$ for some mapping $\mathbf{M}(\cdot)$. The algorithm is stopped when the generated sequence has converged within a tolerance of the limit θ^*. Examples include the Newton-Raphson Method, Fisher's Method of Scoring, Gradient Descent, the EM Algorithm, Block Relaxation, and many imputation methods. As shall be seen, a statistically motivated fitting algorithm will nearly always converge linearly unless it has been specifically engineered so that $\mathbf{M}'(\theta^*) = 0$.

Linear convergence is common for convergent sequences defined by repeatedly applying a function $\mathbf{f}$ so that $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$. To see this, perform a Taylor expansion about the limit point $\mathbf{x}^*$:

$$\begin{aligned}
\mathbf{f}(\mathbf{x}_n) &\approx \mathbf{f}(\mathbf{x}^*) + \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{f}(\mathbf{x}_n) &\approx \mathbf{x}^* + \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{f}(\mathbf{x}_n) - \mathbf{x}^* &\approx \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{x}_{n+1} - \mathbf{x}^* &\approx \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*)
\end{aligned}$$

Taking norms of both sides yields that:

$$\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \lesssim \|\mathbf{f}'(\mathbf{x}^*)\| \|\mathbf{x}_{n+1} - \mathbf{x}^*\|$$

The situation here is a little subtle because $\mathbf{f}$ is a multivariate function. The exact rate is of convergence is controlled by the norm of the Jacobian matrix $\mathbf{f}'(\mathbf{x})$ at $\mathbf{x}^*$. So long as there is a matrix norm such that $\|\mathbf{f}'(\mathbf{x}^*)\| < 1$ the sequence will converge linearly at worst, though faster than linear convergence is potentially possible if 0 is an eigenvalue of $\mathbf{f}'(\mathbf{x}^*)$.[8] If $\mathbf{f}'(\mathbf{x}^*) = \mathbf{0}$, the convergence will be superlinear.

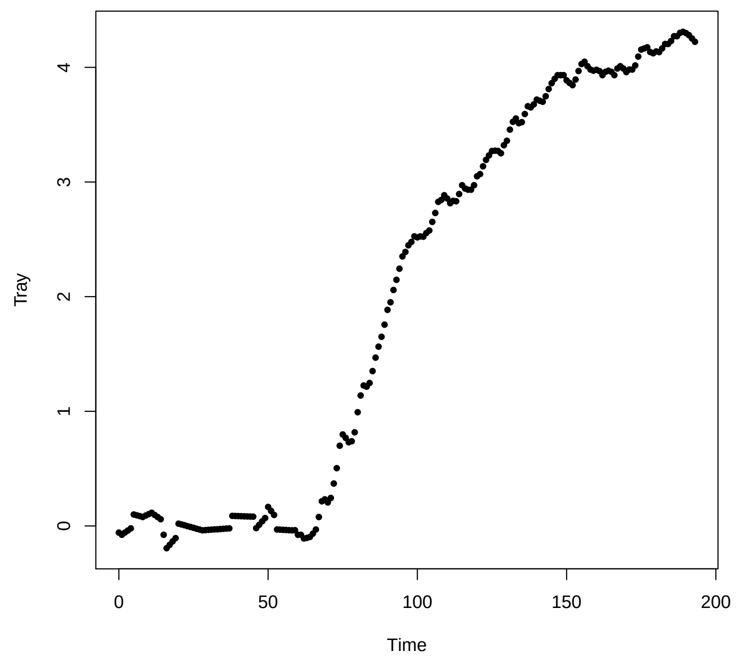
1.8 Overview of Appendices

Appendix A contains an overview of the optimisation methods used throughout this thesis. The material in Appendix A is a prerequisite for Chapter 2 in particular.

[7]The simple definition presented here is known as *Q-Convergence*, and the extended definition is known as *R-Convergence*.[**?**]

[8]Consider for example the multivariate sequence defined by $(x_{n+1}, y_{n+1}) = (x_n^2, y_n/2)$. The convergence towards zero is superlinear in the x direction, but only linear in the y direction. If $(x_0, y_0) = (0.5, 0)$, then the convergence will be superlinear. Usually however the y component will be nonzero and will drag the convergence rate down to linear convergence.

Appendix B discusses the Implicit Filtering method. Implicit Filtering was found to be inadequate for our purposes and hence does not play an active role in this thesis. This material is presented in the appendices for completeness.



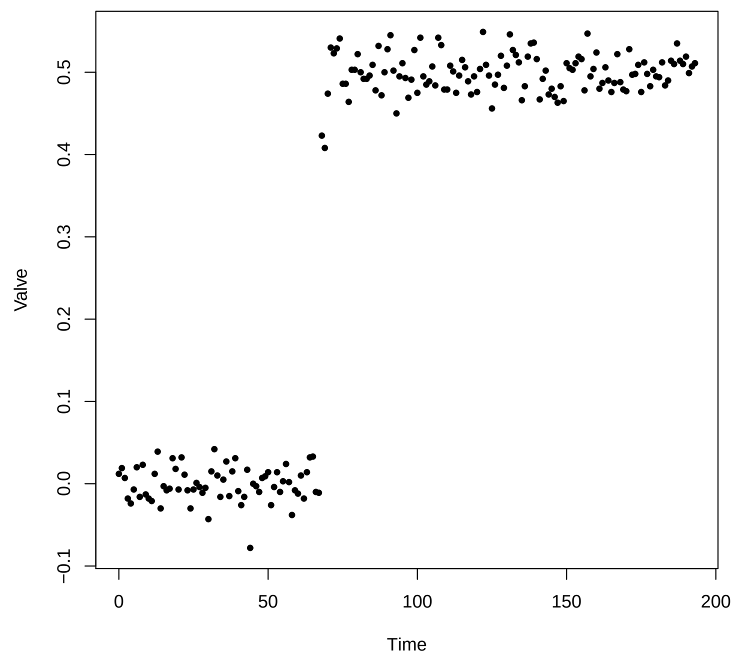


Figure 1.7: Reflux Data

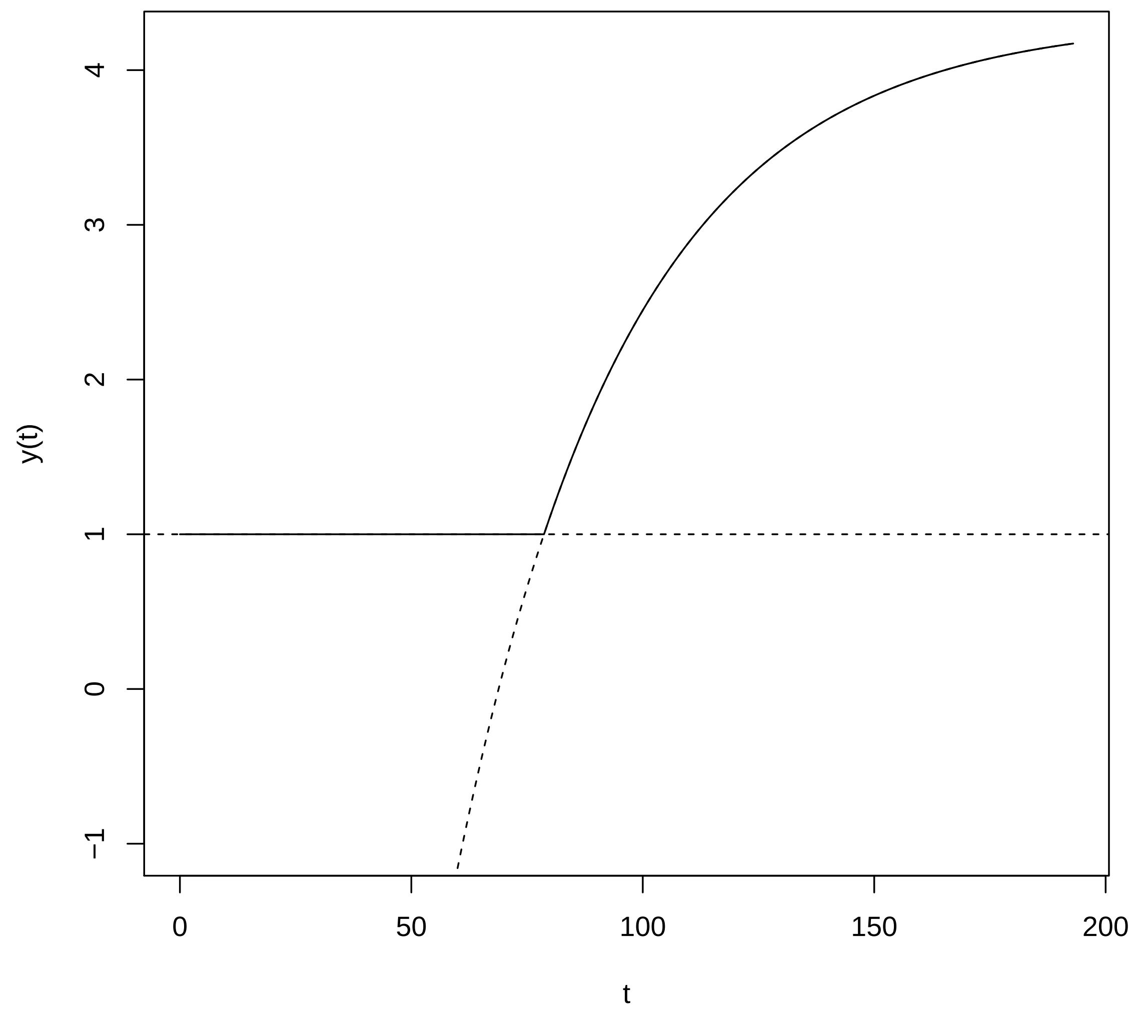


Figure 1.8: The fitted curve is constructed by combining two functions together using the maximum operator.

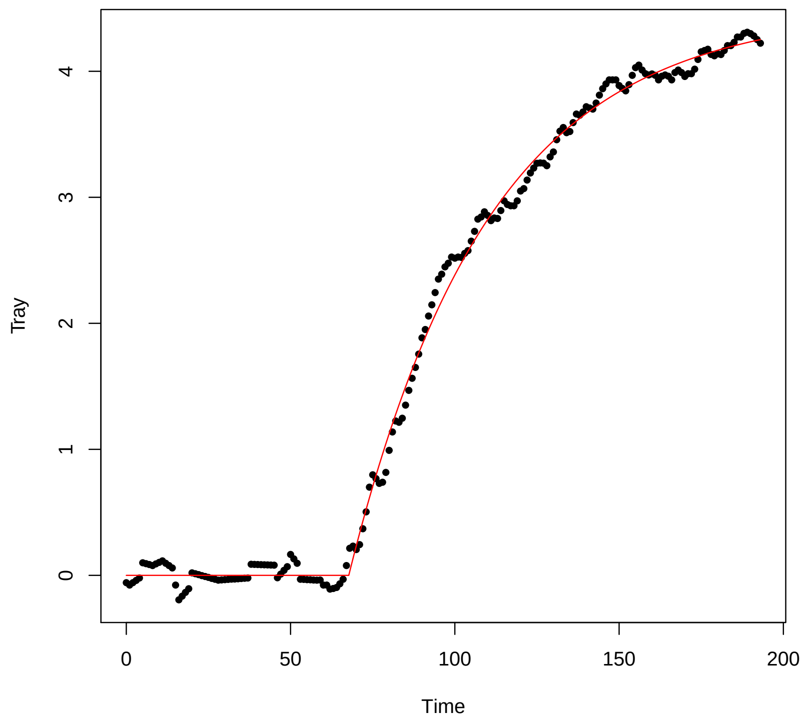


Figure 1.9: Plot of the parameteric fit to the the Reflux data.

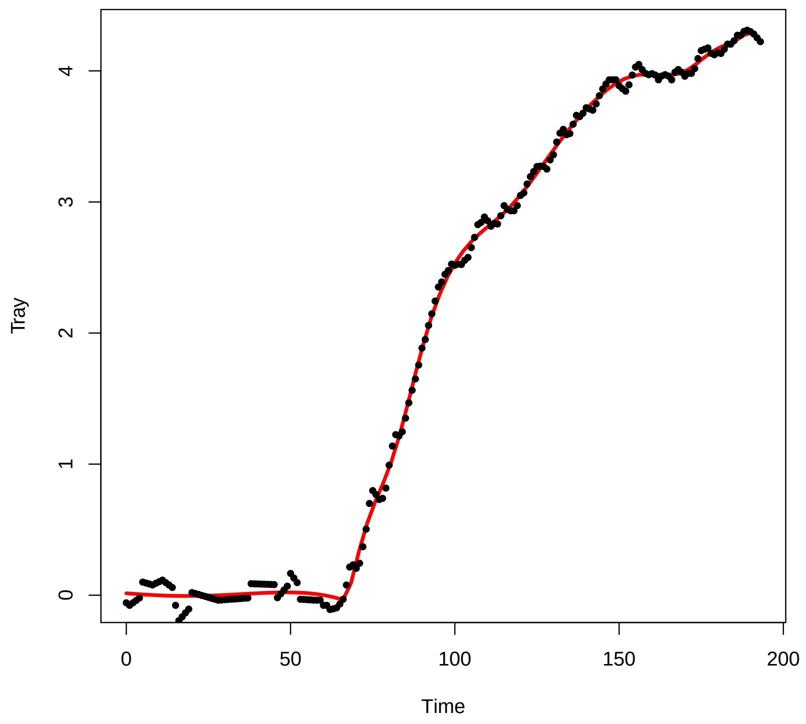


Figure 1.10: Modelling the Reflux data using `Data2LD`.

Chapter 2

Hierarchical Estimation and the Parameter Cascade

This chapter opens with a discussion on fitting a semi-parameteric generalisation of the ODE used to model the reflux data in Section 1.6.1. It is shown that a hierarchical method is needed, where an *inner problem* consists of fitting the ODE to the data given the parameters, and a *middle problem* in turn entails finding the best choice of parameters. The Parameter Cascade method for fitting penalised smoothing problems with unknown parameters is then introduced. A familiarity with numerical optimisation methods such as Gradient Descent, the Newton-Raphson Mathod, and Line Searches is assumed. Details are provided in Appendix A.

2.1 Fitting the Reflux Data Using a Semi-Parametric ODE Model

In Section 1.6 the reflux data was discussed. An ODE model $y' = \beta y + u_0$ as described in Equation 1.7 was fitted to the data. After performing a parametric fit by approximating $y(t)$ by basis functions, `Data2LD` was then used to fit a sophisticated semi-parametric model where the ODE $y'(t) = \beta(t) + u(t)$ holds approximately. While the parametric approach was explored in detail, the discussion of `Data2LD` was deferred to this chapter.

To introduce some of the ideas behind `Data2LD`, a semi-parametric ODE model that lies in between the purely parameteric approach discussed in Sections 1.6.1 and **??**, and the relatively non-parameteric employed by `Data2LD` will be briefly examined in this section. This ODE model takes the form:

$$\begin{cases} y'(t) & = \begin{cases} \beta y(t) & t < t_0 \\ \beta y(t) + u(t) & t \geq t_0 \end{cases} \\ \\ y(0) & = 0 \end{cases}$$

The forcing function $u(t)$ is unknown and must be estimated alongside β, therefore this is a semi-parametric model. As in Section 1.6, the breakpoint at t_0 will be assumed to be known in advance.

The solution to the homogeneous ODE $y'(t) = \beta y(t)$ is given by $y(t) = Ce^{\beta t}$ where C is an unknown constant. Since $y(0) = 0$, this implies that $y(t) = 0$ if $t < t_0$. To take into account the forcing function that is turned on for $t \geq t_0$, variation of parameters will be employed. Assume that the solution for the case with the forcing function can be written as $y(t) = e^{\beta t}v(t)$, with the time-varying coefficient $v(t)$ to be determined. Substitute this into the case for $t \geq t_0$:

$$\begin{aligned} e^{\beta t}v'(t) + \beta e^{\beta t}v(t) &= \beta e^{\beta t}v(t) + u(t) \\ \therefore \qquad v'(t) &= e^{-\beta t}u(t) \\ \therefore \qquad v(t) &= \int_{t_0}^{t} e^{-\beta t}u(t)dt \end{aligned}$$

Setting the limits of integration from t_0 to t ensures that $v(t_0) = 0$ so that the solution for $t \geq t_0$ is matched continuously with the solution for $t < t_0$. The solution will not be differentiable at t_0 however unless[1] $u(t_0) = 0$, so there will generally be a kink at the breakpoint in a similar fashion as in Figures 1.8 and 1.9.

The general solution can be written as:

$$y(t) = e^{\beta t}v(t)\mathbf{1}(t - t_0).$$

Here $\mathbf{1}(t)$ denotes the Heavyside step function:

$$\mathbf{1}(t) = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases}$$

The ODE model can thus be equivalently formulated as a functional linear model as in [**?**] of the form:

$$y(t) = L_\beta u(t),$$

[1]The left derivative at t_0 is identically zero. The right derivative at t_0 is found by substituting $y'(t_0) = \beta y(t_0) + u(t_0) = u(t_0)$, recall that $v(t)$ is chosen so that $y(t_0) = 0$. The left and right derivatives will not match unless $u(t_0) = 0$.

where the parameterised linear operator L_β is defined by:

$$L_\beta f(t) = \mathbf{1}(t-t_0)e^{\beta t}\int_{t_0}^{t} e^{-\beta t} f(t)dt,$$

where L_β is an operator that takes a function as an argument and returns another function. The parameter β and function $u(t)$ are unknown and must be estimated. For the special case discussed in Section 1.6 where $u(t) = u_0$ for all t, we find that:

$$v(t) = -\frac{u_0}{\beta}(e^{-\beta t} - e^{-\beta t_0})$$

The solution can then be straightforwardly found:

$$y(t) = \frac{u_0}{\beta}\left(1 - e^{\beta(t-t_0)}\right)\mathbf{1}(t-t_0)$$

This is the same solution as was found at the start of Section 1.6, except that the ODE is of the form $y' = \beta y + u_0$ for $t \geq t_0$ here instead of being of the form $y' = -\beta y + u_0$ as in Section 1.6.

Figure 2.1 plots a solution with $\beta = -1, t_0 = 4$, and $u(t) = 1 + 0.2\sin(10t)$. In this situation, the constant forcing term is perturbed by high frequency, low amplitude noise. It can be seen that the solution resembles the constant case where $u(t) = u_0$ except that $y(t)$ converges to a sinusoidal function instead of a constant.

Let I denote the set of indices for which $t_i > t_0$. If β were known, $v(t)$ could estimated by non-parameterically regressing the values $y_i/e^{\beta t_i}$ against t_i where $i \in I$. The forcing function $u(t)$ could then be estimated by differentiating the estimate for $v(t)$ and multiplying it by $e^{\beta t}$. If $v(t)$ were known on the other hand, β could be estimated by minimising the following non-linear least squares criterion:

$$SSE[\beta; v(t)] = \sum_{i \in I}[y_i - e^{\beta t_i} v(t_i)]^2.$$

A hierarchical estimation strategy thus seems natural. For a given choice of β, let $v(t|\beta)$ denote the estimate of $v(t)$ produced by regressing $y_i/e^{\beta t_i}$ against t_i for $i \in I$. Define the associated least squares error by:

$$\begin{aligned} H(\beta) &= SSE[\beta; v(t|\beta)] \\ &= \sum_{i \in I}[y_i - e^{\beta t_i} v(t_i|\beta)]^2 \end{aligned}$$

The sketch presented here fails to address two important questions: how to estimate $v(t)$ given β, and how to optimise $H(\beta)$. In this case the *inner problem* entails estimating $v(t)$ given the parameters and β, and the *middle problem* entails minimising $H(\beta)$. `Data2LD` similarly employs a powerful hierarchical fitting methodology for FDA problems known as the Parameter Cascade, which will be introduced in the next section.

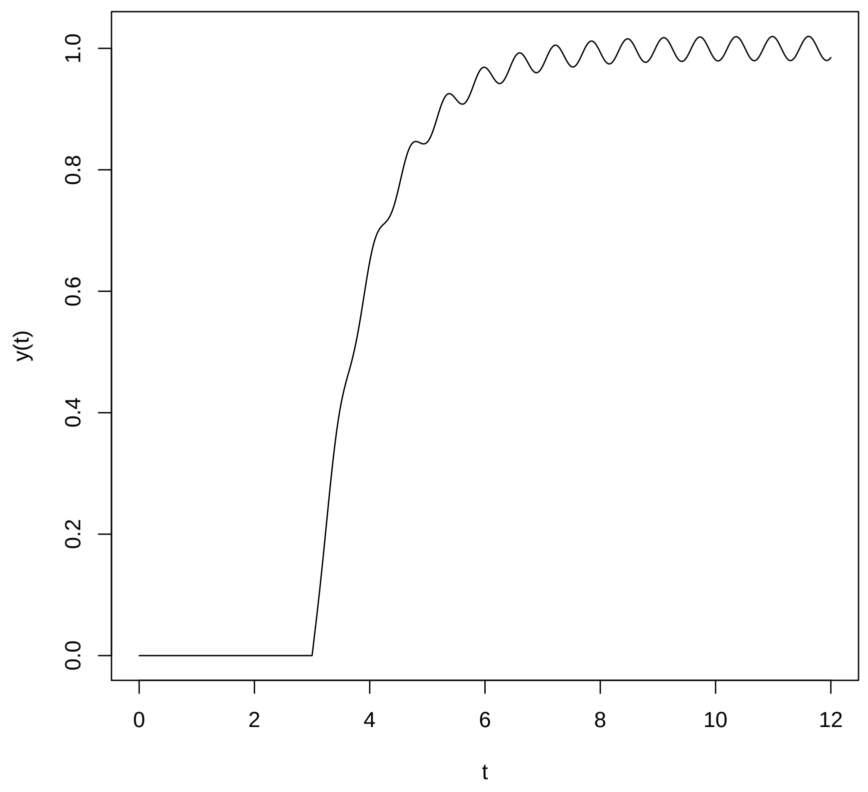


Figure 2.1: A solution to the general Reflux ODE with a constant plus sinusoidal forcing function.

2.2 The Two-Stage Parameter Cascade

Consider the following penalised regression problem:

$$PENSSE(f,\theta) = \sum_{i=1}^{N}(y_i - f(t_i))^2 + \lambda\int_0^1 |T_\theta f|^2 dt.$$

Here T_θ is a differential operator that is parameterised by an unknown θ that is to be estimated. T_θ can be an ordinary differential operator or a partial differential operator; linear, quasi-linear, or nonlinear. There are two statistical objects to be estimated: the parameter θ, and the function $f(t)$. Ramsay and Cao [**?**], propose the following hierarchical approach to estimation:

- Given a fixed value of θ, let $f(t|\theta)$ denote the function that minimises $PENSSE(f,\theta)$.
- For a given value of θ, its associated mean square error is then defined by:

$$MSE(\theta) = \frac{1}{N}\sum_{i=1}^{N}[y_i - f(t_i|\theta)]^2$$

By making $f(t)$ dependent on θ, the fitting problem has been reduced to a non-linear least squares problem. This leaves the issue of estimating the optimal value of θ - Ramsay and Cao propose the use of Gradient Descent. For a given value of θ, $f(t|\theta)$ is found. These two values are then used to compute $MSE(\theta)$ and $\nabla MSE(\theta)$. Finally, a new value of θ is computed by perturbing θ in the direction of the gradient. This scheme is sketched out in Figure 2.2.

It is assumed that $f(t)$ can be represented by a finite vector $\mathbf{c}$ associated with an appropriate basis. This leads to a pair of nested optimisation problems: the *inner optimisation* involves finding the value of $\mathbf{c}$ that minimises the penalised least squares criterion given θ, and the *middle optimisation* entails finding the value of θ that minimises $MSE(\theta)$. There is thus a 'cascade' of estimation problems, where the results of the lower level estimation problem feed back in to the higher level ones.

Note that every time a new value of θ is introduced, the associated function $f(t|\theta)$ must be computed from scratch. The middle optimisation can thus generate many inner optimisation subproblems as the parameter space is explored, and these in turn could require multiple iterations to complete if no explicit formula for $\mathbf{c}$ given θ is available.

Figure 2.2 is a highly idealised sketch of the Parameter Cascade. The main abstraction is that the step of computing $f(t|\theta)$ is presented as a single atomic and organic step, even though it could in fact be a complex process. This risks masking some of the computational work that is occurring within the algorithm. A more detailed description is provided in Figure 2.3. In this thesis, Parameter Cascade problems that cannot be differentiated easily or at all are considered.

2.3 The Three-Stage Parameter Cascade

The two-stage parameter cascade assumes that the structural parameter λ is treated as fixed. However it is possible to extend the Parameter Cascade to also estimate λ. In this case it is necessary to introduce an *outer criterion*, $F(\lambda)$, that determines how good a given choice of λ is. A common choice for the outer criterion is Generalised Cross Validation [**? ?**].

Just as the problem of fitting a function $f(\cdot|\theta)$ can generate an optimisation subproblem, that of fitting a third level in the cascade can generate a series of subproblems to find the best parameter choice associated with a given value of λ. This in turn generates a series of subproblems to find the fitted function as the parameter space is explored.

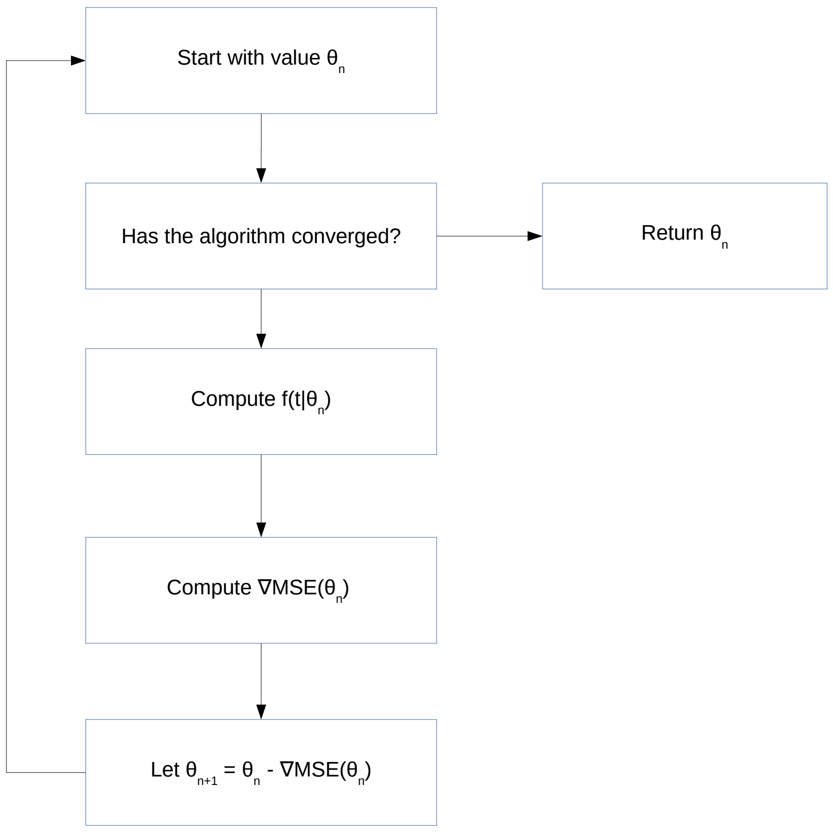
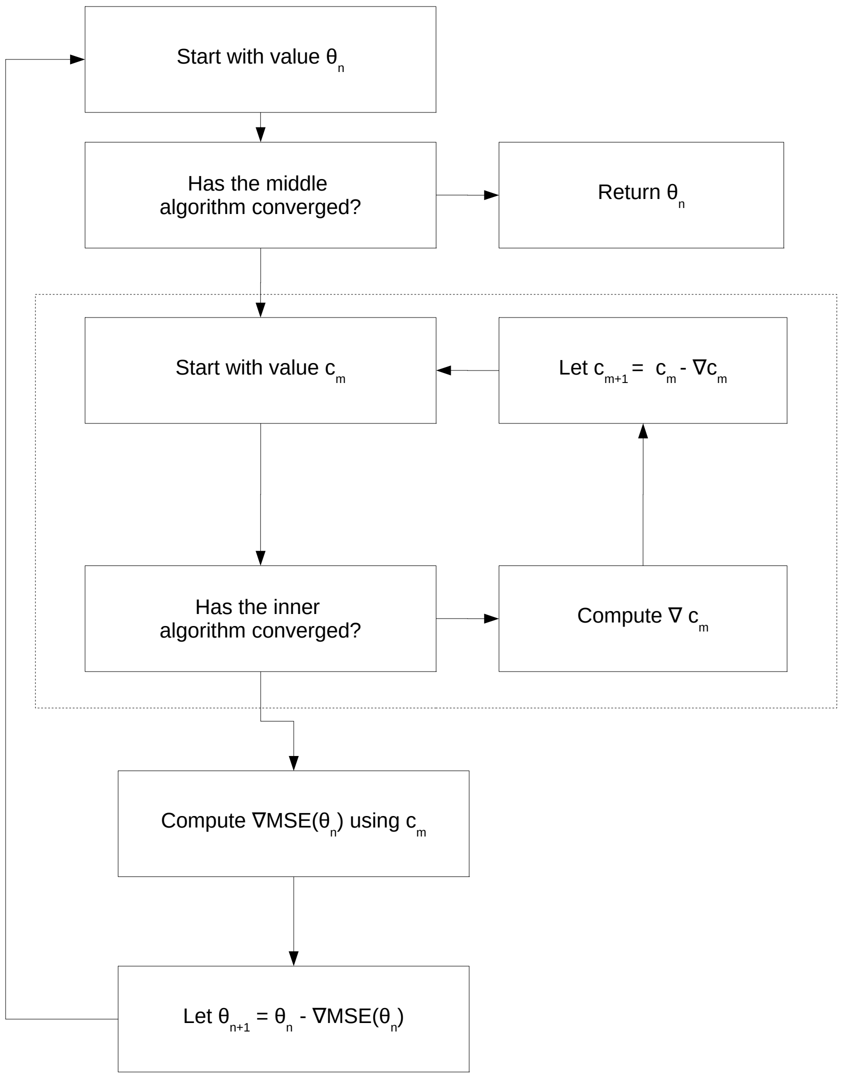


Figure 2.2: Two Stage Parameter Cascade (Simplified)



Start with value θ_n
Has the middle algorithm converged?
Return θ_n
Start with value c_m
Let $c_{m+1} = c_m - \nabla c_m$
Has the inner algorithm converged?
Compute ∇c_m
Compute $\nabla MSE(\theta_n)$ using c_m
Let $\theta_{n+1} = \theta_n - \nabla MSE(\theta_n)$

Figure 2.3: Schematic of the Two Stage Parameter Cascade With the Inner Optimisation Visible

Neither the FDA nor Data2LD packages implement the three stage parameter cascade. They instead require practitioners to determine the best choice of λ by cycling through a set of predetermined values and/or using manual adjustment. However, there is no guarantee that the optimal choice of λ will lie in a finite set of predetermined values and manual adjustment is slow and cumbersome compared to an automated optimisation routine.

2.4 Investigating the Data2LD Package

Data2LD uses a sophisticated two-level parameter cascade algorithm to fit parameters to the data, which is briefly described here. The inner level of the parameter cascade is implemented by the eponymous Data2LD routine. The middle level is implemented by the Data2LD.opt command [?]. The outer level of optimisation for choosing the trade-off parameter λ is not implemented.

The Data2LD function is written such that upon calling the method, it returns a list with a of number computed quantities and statistics. For example, the associated Mean Square Error (MSE),[2] the gradient of MSE and the Hessian of MSE will always be computed and returned whether one needs them or not. Only the Data2LD.opt command is investigated in detail here. The middle level of the Parameter Cascade is generally easier to implement than the inner level. When implementing the middle level, one can generally treat the lower level as a 'black box' that accepts a given choice of parameters as inputs, and then returns the value of the objective function (and sometimes derivatives) as outputs.

2.4.1 How Data2LD Estimates Parameters

Let $\mathbf{g}_n$ and $\mathbf{H}_n$ respectively denote the gradient vector and Hessian matrix of the objective function $f(\theta)$ at θ_n. The search directions used by the Data2LD.opt command are the Gradient Descent direction:

$$\mathbf{p}_n = -\mathbf{g}_n \tag{S1}$$

and the Newton Direction:

$$\mathbf{p}_n = -\mathbf{H}_n^{-1}\mathbf{g}_n \tag{S2}$$

Data2LD.opt makes use of line search methods, which are discussed in more detail in Section A.7. For line search methods, the algorithm takes a variable step α_n in the direction $\mathbf{p}_n$ on each iteration. Simpler algorithms such as Gradient Descent set α_n to a constant. Choosing the value of α_n is a subproblem that must be tackled on each

[2]The MSE returned is the MSE associated with the choice of parameters. The command Data2LD fits a function using the choice of parameters passed to it, and then reports the associated MSE alongside other related values.

iteration.

Data2LD uses four tests to determine how good a step is:[3]

- First Wolfe Condition (checks for sufficient decrease in the objective function):

$$f(\theta_n + \alpha_n \mathbf{p}_n) \leq f(\theta_n) + c_1 \alpha_n \mathbf{p}_n^\top \mathbf{g}_n \tag{T1}$$

- Second Wolfe Condition (checks for sufficient decrease in curvature):

$$|\mathbf{p}_n^\top \nabla f(\theta_n + \alpha_n \mathbf{p}_n)| \leq c_2 |\mathbf{p}_n^\top \nabla f(\theta_n)| \tag{T2}$$

- Has the function even decreased compared to the previous iteration?

$$f(\theta_n + \alpha_n \mathbf{p}_n) \leq f(\theta_n) \tag{T3}$$

- Has the slope along the search direction remained nonnegative?

$$\mathbf{p}_n^\top \nabla f(\theta_n + \alpha_n \mathbf{p}_n) \leq 0 \tag{T4}$$

The constants c_1 and c_2 are required to satisfy $0 < c_1 < c_2 < 1$[**?**]. These four tests are illustrated in Figure 2.4. Written in terms of $\phi(\alpha) = f(\theta + \alpha \mathbf{p}_n)$ the tests are:

$$\phi(\alpha_n) \leq \phi(0) + c_1 \alpha_n \phi'(0) \tag{T1'}$$

$$|\phi'(\alpha_n)| \leq c_2 |\phi'(0)| \tag{T2'}$$

$$\phi(\alpha) \leq \phi(0) \tag{T3'}$$

$$\phi'(\alpha) \leq 0 \tag{T4'}$$

If **T1** and **T2** are satisfied, then the line search has converged completely. If **T3** has failed, this represents a total failure because it means the line search has failed to produce any improvement in the objective function. A failure in **T4** means the function has overshot a critical point. The use of four tests is a little unusual here. The literature suggests that only the Wolfe Conditions **T1** and **T2** are needed as discussed in Section A.7. Data2LD.opt is designed to be robust against the possibility that the objective

[3]Data2LD actually tests for the negation of **T3** and **T4**. For the sake of consistency the logical negations of the two tests used by Data2LD are presented here so that passing a test is consistently a good thing and failing consistently represents unsatisfactory or pathological behaviour.

function might not behave as predicted by the computed gradient and Hessian.

Depending on the outcome of the tests, `Data2LD` chooses the stepsize as follows:

- If **T1**, **T2**, and **T3** are passed, the algorithm terminates.
- If **T1** and **T1** are passed, or **T4** is passed but **T3** is failed, it means that the slope is satisfactory, but the function has increased rather than decreased. Here, `Data2LD` reduces the step size.
- If all four tests are failed, then the newest point is entirely unsuitable and `Data2LD` falls back on interpolation to try to find a critical point of $\phi(\alpha)$, or quadratic interpolation methods if necessary.

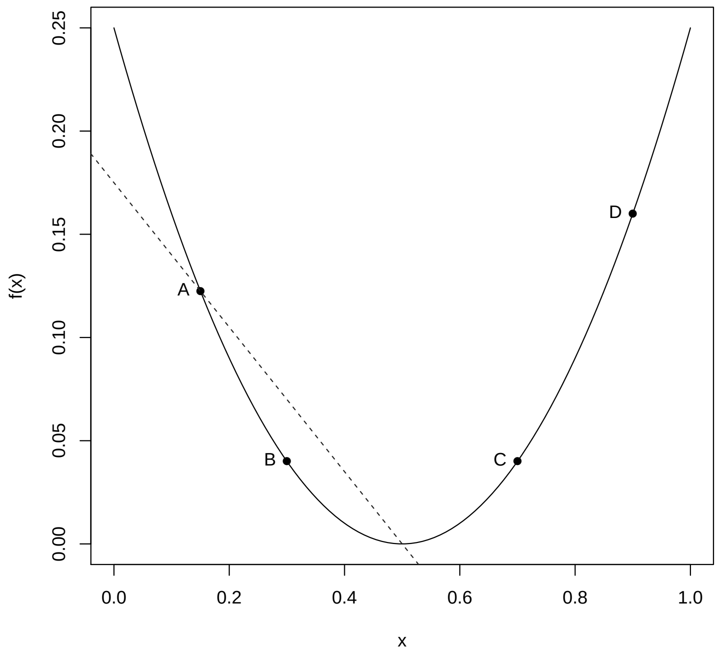


Figure 2.4: Point A is the initial point. Point B passes **T1** with $c_1 = 0.5$ and passes **T2** with $c_2 = 0.9$. Point C fails **T1** with $c_1 = 0.5$ and also fails **T3**, but passes **T4**, and passes **T2** with $c_2 = 0.9$. Point D fails all four tests.

If the line search succeeds in reducing the objective function, `Data2LD` uses the Newton search direction for the next iteration. If the line search makes the objective function worse, the Gradient Descent direction is used. In the event of the line search making the objective function worse twice in a row, `Data2LD` returns an error.

Somewhat peculiarly, `Data2LD` does not make use of $\phi''(\alpha)$ despite being able to compute it easily.[4] One would think that the Newton-Raphson Method would be the first approach to perform the line search attempted before resorting to interpolation-based methods since it's both simpler to implement and faster to converge. The effort required to compute $\phi''(\alpha)$ is mostly a sunk cost because `Data2LD` will return the Hessian matrix on each iteration whether it is needed or not.

The code for`Data2LD` can be difficult to understand. While software with such powerful features was inevitably going to be complex, the complexity is compounded by not heeding best practices recommended for making code readable and easy to maintain. For example, `Data2LD` hardcodes unnamed constants into the code. Allowing such 'Magic Numbers' is strongly discouraged because it makes code more error prone and difficult to understand [**?**].

2.5 Hierarchical fitting of a Partial Differential Equation

A linear PDE that is analogous to the linear ODE used to model the reflux data in Section 1.6 is the Transport Equation:

$$\frac{\partial u(x,t)}{\partial t} + \beta\frac{\partial u(x,t)}{\partial x} = 0$$

A general solution to the Transport Equation is given by [**?**]:

$$u(x,t) = f(x - \beta t)$$

The function $f(\cdot)$ is unspecified. The solution $u(x,t)$ is constant along the rays $x = \beta t + C$. The solution is an animation of the shape $f(x)$ moving to the right at fixed speed β. The ODE $y'(t) + \beta y(t) = 0$ can be thought of as a simplification of the Transport Equation, where it is assumed that $u(x,t)$ only varies with time, and not with space. It is apparent that this PDE has a much richer solution structure than is the case for the ODE, which only has solutions of the form $Ae^{-\beta t}$. Statistically speaking, fitting the Transport Equation to observed data is a semi-parametric problem because one of the parameters to be estimated is a function.

The problem of fitting the Transport Equation is also a transformation model such as that used for the Box-Cox transformation, since the plot of $u(x,t)$ with respect to x at a fixed time t is a transformed version of $f(x)$, the curve at $t = 0$. If the parameter governing the transformation process - β - is known, $f(\cdot)$ is reasonably easy to estimate. Suppose there were n observed values y_i at time t_i and location x_i. It has already been

[4]Differentiating the expression $\phi'(\alpha) = \mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha\mathbf{p}_n)$ with respect to α yields that $\phi''(\alpha) = \mathbf{p}_n^\top \mathbf{H}(\alpha)\mathbf{p}_n$, where $\mathbf{H}(\alpha)$ denotes the Hessian of f evaluated at $\mathbf{x}_n + \alpha\mathbf{p}_n$.

established that the value observed at a point x at time t depends only on $x - \beta t$. The function $f(\cdot)$ could thus be estimated by non-parametrically regressing the observed values at y_i against $x_i - \beta t_i$

What if β is unknown? The above discussion suggests a hierarchical approach to estimation: for a given choice of β, to fit an associated function $f(\cdot|\beta)$ using an appropriate non-parametric estimation method, and compute the associated least squares error. Let $H(\beta)$ be the function that associates each β with its sum of squared error:

$$H(\beta) = \sum_{i=1}^{n} [y_i - f(x_i - \beta t_i|\beta)]^2$$

The problem of minimising $H(\beta)$ is a non-linear least squares problem that is also a two level hierarchical estimation problem such as the two level parameter cascade discussed above. The inner level consists of non-parametrically fitting a function to the set of points $\{(y_i, x_i - \beta_i)\}$ given β. The associated sum of squared errors is then returned as $H(\beta)$. The outer level entails optimising the profiled objective function $H(\beta)$.

This is a broad fitting strategy where different statistical and optimisation approaches can be swapped in and out as needed. There are several ways to tackle the inner function - LOESS; Kernel Regression; Penalised Splines, etc. The least squares loss function could be replaced with another one as suits the problem. There are many methods for optimising $H(\beta)$ that might be attempted - subgradient methods if $H(\beta)$ is convex, Gradient Descent, Gauss-Newton Method, derivative-free methods and so on.

Chapter 3

Derivative-Free Optimisation and the Parameter Cascade

The Parameter Cascade as presented in Chapter 2 requires the computation of derivatives to perform optimisation. However, computing the necessary derivatives can be a time-consuming and complex task. This is especially the case for the higher levels of the Parameter Cascade [**?**]. In some cases the derivatives might not even exist, such as when the loss function used to measure goodness-of-fit is non-differentiable. In this chapter, the use of derivative-free methods for fitting FDA problems is explored. Derivative-free methods are used to tackle a series of increasingly complex problems, culminating in fitting one level of the Parameter Cascade without derivatives.

3.1 Overview of Quadratic Optimisation Methods

A large class of numerical optimisation methods rely on constructing a quadratic approximation to an objective function $f(\theta)$. Given an iterate θ_n and possibly some associated data, a quadratic approximation $m_n(\theta)$ to the objective function is constructed. The next iterate θ_{n+1} is then found by minimising $m_n(\theta)$. Constructing the approximate quadratic and then minimising it tends to be straightforward. If the next iterate $\theta_{n+1}(\theta)$ is unsatisfactory, a new quadratic model function $m_{n+1}(\theta)$ is again minimised, producing the next iterate θ_{n+2}. Ideally, the θ_n will approach the optimal point as the sequence of quadratic models become increasingly accurate approximations to $f(\theta)$ in the neighbourhood of the optimal point, so that the process can be repeated until convergence [**?**].

3.1.1 Newton's Method

The Newton-Raphson Method is a well-known member of this class. Newton's method constructs a quadratic approximation using a second-order Taylor expansion around θ_n :

$$f(\theta) \approx m_n(\theta) = f(\theta_n) + f'(\theta_n)(\theta - \theta_n) + \frac{1}{2}f''(\theta_n)(\theta - \theta_n)^2$$

It is not difficult to show that the critical point of $m_n(\theta)$ is given by $\theta_{n+1} = \theta_n - f'(\theta_n)/f''(\theta_n)$, which is the usual Newton formula [? ? ? ?].

For a point close to θ_n, the difference between $f(\theta)$ and $m_n(\theta)$ is roughly equal to $[f'''(\theta_n)/3!](\theta - \theta_n)^3$ so long as $f(\theta)$ is sufficiently well behaved [?]. This formula suggests that if θ_n is close to the optimal point θ^* so that $|\theta^* - \theta_n|$ is sufficiently small, then $|\theta_n - \theta^*|^3$ will be very small indeed and so the quadratic model will be a very accurate approximation of $f(\theta)$ around θ^*. As a result, θ_{n+1} will be quite close to θ^*. The next model $m_{n+1}(\theta)$ will thus be substantially better than $m_n(\theta)$ at approximating $f(\theta)$ around θ^*, and θ_{n+2} will be much closer to θ^* than θ_{n+1}. Newton's method converges very rapidly so long as one is sufficiently close to θ^* to start with. In fact, Newton's method converges quadratically, as discussed in Section A.1.

Newton's method is a very effective estimation algorithm so long as the derivatives $f'(\theta)$ and $f''(\theta)$ can be computed, and so long as the initial starting value is not too far from the optimal value. Choosing a good initial value is thus very important. For maximum likelihood estimation for example, a method of moments estimator could be used to provide an initial starting value.

3.1.2 Secant Method

If the second derivative is difficult to calculate, one can approximate it with a difference quotient instead [? ?]:

$$f''(\theta) \approx \frac{f'(\theta_n) - f'(\theta_{n-1})}{\theta_n - \theta_{n-1}} \tag{3.1}$$

This leads to the quadratic approximation:

$$m_n(\theta) = f(\theta_n) + f'(\theta_n)(\theta - \theta_n) + \frac{1}{2}\left(\frac{f'(\theta_n) - f'(\theta_{n-1})}{\theta_n - \theta_{n-1}}\right)(\theta - \theta_n)^2,$$

and the update formula:

$$\begin{aligned}\theta_{n+1} &= \theta_n - \left[\frac{f'(\theta_n) - f'(\theta_{n-1})}{\theta_n - \theta_{n-1}}\right]^{-1} f'(\theta_n) \\ &= \theta_n - \frac{f'(\theta_n)[\theta_n - \theta_{n-1}]}{f'(\theta_n) - f'(\theta_{n-1})} \\ &= \frac{\theta_{n-1}f'(\theta_n) - \theta_n f'(\theta_{n-1})}{f'(\theta_n) - f'(\theta_{n-1})}.\end{aligned}$$

The Secant Method is straightforward to implement, and only requires first derivatives. Relying on $f(\theta_n), f'(\theta_n)$ and $f'(\theta_{n-1})$ instead of $f(\theta), f'(\theta_n)$ and $f''(\theta_n)$ has a drawback however. The Secant Method's model is less accurate because θ_{n-1} tends to be further from θ^* than θ_n. More formally, the error for the model is roughly equal to $[f'''(\theta_n)/3!](\theta_n - \theta)^2(\theta_{n-1} - \theta)$. If the sequence is converging to θ^*, substituting in the $(\theta - \theta_{n-1})$ term inflates the error relative to Newton's Method and acts as a drag on convergence. It can be shown that the Secant Method convergences superlinearly with order 1.618. However avoiding the cost of computing a second derivative on each step means that more iterations can be completed in a given period of time. The Secant Method is comparable with Newton's Method, and can be faster if computing the second derivative is difficult.

The Secant Method is a widely used method that provides a good trade-off between convergence speed and ease of implementation [?]. Multivariate generalisations of the Secant Method when used for optimisation are usually referred to as *Quasi-Newton Methods,* and are discussed in Section A.3.

3.1.3 Successive Parabolic Interpolation

Parabolic interpolation goes one step further than the Secant Method and dispenses with derivatives entirely. Instead, a model function is constructed by interpolation through the points $(\theta_n, f(\theta_n))$, $(\theta_{n-1}, f(\theta_{n-1}))$, and $(\theta_{n-2}, f(\theta_{n-2}))$[? ?].

$$\begin{aligned} m_n(\theta) = &f(\theta_n)\frac{(\theta - \theta_{n-1})(\theta - \theta_{n-2})}{(\theta_n - \theta_{n-1})(\theta_n - \theta_{n-2})} \\ &+ f(\theta_{n-1})\frac{(\theta - \theta_n)(\theta - \theta_{n-2})}{(\theta_n - \theta_n)(\theta_n - \theta_{n-2})} \\ &+ f(\theta_{n-2})\frac{(\theta - \theta_{n-1})(\theta - \theta_n)}{(\theta_n - \theta_{n-1})(\theta_n - \theta_n)} \end{aligned}$$

This model has an approximate error of $[f'''(\theta_n)/3!](\theta - \theta_n)(\theta - \theta_{n-1})(\theta - \theta_{n-2})$. By relying on the past two iterates, the rate of convergence is slowed further. Parabolic interpolation has an order of convergence of 1.32 [?]. An issue with parabolic interpolation is providing enough initial points to seed the method [?]. This difficulty is more acute for the multivariate case. One approach is to provide enough points at the start, and run the algorithm from there. Alternatively, one can start off with just enough points needed to estimate an ascent or descent direction and construct a linear approximation, and then run the optimisation routine using a sequence of linear approximations until there are enough points to construct a parabola. If one is using a linear approximation, a limit must be imposed on the maximum distance that the routine can travel on each iteration since linear functions do not have a minimum or maximum and diverge off to infinity.

3.1.4 Discussion

All three approaches are governed by the same fundamental theory - approximating functions by polynomials. The difference between the methods centres on the precise inputs used to construct an approximation. This means that if a problem is suitable for Newton's Method, the other two methods will very likely perform well too. On the other hand, the methods all share the same fundamental handicap; these methods are not guaranteed to converge unless the starting point is close to the optimal value. Local convergence does not necessarily imply global convergence. The error terms in the quadratic approximations are all of the form $(\theta - \theta_n)^3$. If $|(\theta - \theta_n)|$ and any other error terms are small, the error in the approximation will be much smaller since it is proportional to the product of three such errors. If however the errors are large,their product might be so large that the method fails to converge [**? ? ?**].

This is less academic than it might seem. Suppose one had a complicated likelihood function $L(\theta)$. Perhaps to evaluate the likelihood it is necessary to numerically integrate a complex marginal distribution that depends on θ. Instead of attempting to find explicit formulae for the score and information functions, if crude estimates of $\hat{\theta}$ and of the error $\hat{\sigma}_\theta$ were available, then successive parabolic interpolation with $\{\hat{\theta}, \hat{\theta} - 2\hat{\sigma}_\theta, \hat{\theta} + 2\hat{\sigma}_\theta\}$ could be used as a set of starting points. If $L(\theta)$ is in fact a well behaved smooth function, then parabolic interpolation will find the value of θ that maximises $L(\theta)$ fairly quickly. It is necessary to provide plausible starting values for θ because the quadratic model is only certain to be valid if one is already near the optimal value.

3.2 Modifying the `Data2LD.opt` Routine

In Section 1.4, the following penalised regression model is used for fitting the melanoma data:[1]

$$PENSSE(f|\theta, \rho) = (1-\rho)\sum[y_i - f(t_i]^2 + \rho\lambda \int |f - \theta f^{(4)}|^2 dt$$

While `Data2LD` can fit a penalty of the form $f - \theta f^{(4)}$, and it can enforce linear equality constraints such as $\theta_1 - \theta_2 = 0$, it cannot enforce linear inequality constraints such as $\theta \geq 0$, nor can it work with penalties of the form $f - \theta^2 f^{(4)}$[**?**]. This means that it is possible, in principle, that the estimated value of θ returned by `Data2LD.opt` is negative even though this value is not valid for the modelling problem at hand since it is desired that $\theta \geq 0$.

Approximating the Derivative: We modified a single line in the code for the `Data2LD.opt` routine introduced in Section 2.4 to use a finite difference approximation

[1]In this section, for the sake of consistency with the notation used by `Data2LD`, θ is used to denote the frequency parameter instead of the usual ω, and $\rho = \lambda/(1+\lambda)$ is used to determine the trade-off between fit to the data and fit to the ODE model instead of λ.

to the first derivative of the objective function rather than Newton's method.

$$f'(\theta) \approx \frac{f(\theta_n) - f(\theta_{n-1})}{\theta_n - \theta_{n-1}} \tag{3.2}$$

Because only a single line was changed, `Data2LD.opt` still computes the second derivatives and makes use of them in the same fashion as before. Note also that this modified code will only work for problems with a single parameter to be estimated because it will only compute the derivative in one direction. Ignoring the line search, the optimisation routine thus now takes the following form:

$$\begin{cases} d_n & = \frac{f(\theta_n) - f(\theta_{n-1})}{\theta_n - \theta_{n-1}} \\ \theta_{n+1} & = \theta_n - d_n / f''(\theta_n) \end{cases} \tag{3.3}$$

The first derivative is approximated using a secant equation, but the second derivative is computed exactly.

Comparing the Modfied Method with the Original Method: The fitting algorithm used by `Data2LD.opt` is hierarchical because there are two levels of optimisation, an outer level that computes search directions, and an inner level that conducts line searches. This makes charting the course of the method a little tricky. The easiest approach is to reproduce the output of `Data2LD.opt`.

Below the output for the original method and the modified method when applied to fitting the melanoma data are presented. The workings of the two different levels can be seen. At the top level, `Data2LD.opt` computes the value of the objective function and the magnitude of the gradient used for the search direction. It then reports the values of the objective function and directional derivatives along the search direction computed in the course of conducting a line search. For the modified method, the computed gradients and directions are approximate. Both methods achieve a similar value of the objective function, but the estimated values of θ differ somewhat. The modified method is clearly much slower than the original, which is congruent with our theoretical analysis. The associated values of the objective function for the two estimates are so similar it is difficult to conclusively say which computed estimate is better.

Newton Method with Gradient Line Search (Data2LD.opt)

```
Iter.    Criterion   Grad Length
0        0.03974      0.002238
  theta = 0.400000, dtheta = -0.001867
  theta = -0.104395, dtheta = 0.000476
  theta = 0.002903, dtheta = -0.000044
1        0.039187      4.4e-05
```

```
  theta = -0.494194, dtheta = 0.002106
  theta = -0.006478, dtheta = 0.000002

2        0.039187      2e-06
Convergence reached.
```

Newton Method with Secant Approximation To First Derivative (Modified)

```
Iter.    Criterion   Grad Length
0        0.03974      0.002238
  theta = 0.400000, dtheta = 0.001898
  theta = 0.414612, dtheta = -0.000300
  theta = 0.401771, dtheta = 0.000230
  theta = 0.402048, dtheta = -0.000005
1        0.039554      0.001876
  theta = 0.304096, dtheta = 0.001555
  theta = 0.318509, dtheta = -0.000241
  theta = 0.305876, dtheta = 0.000184
2        0.039404      0.001472
  theta = 0.209704, dtheta = 0.001177
  theta = 0.224023, dtheta = -0.000176
  theta = 0.211530, dtheta = 0.000134
3        0.039293      0.001047
  theta = 0.117183, dtheta = 0.000772
  theta = 0.131623, dtheta = -0.000107
  theta = 0.119149, dtheta = 0.000081
4        0.039221      0.000609
  theta = 0.026769, dtheta = 0.000353
  theta = 0.042419, dtheta = -0.000035
5        0.039191      0.000237
  theta = -0.034310, dtheta = 0.000030
  theta = 0.005845, dtheta = 0.000061
  theta = 0.010409, dtheta = -0.000007
6        0.039187      8.1e-05
  theta = -0.021601, dtheta = -0.000015
  theta = -0.005596, dtheta = 0.000053
  theta = -0.000575, dtheta = 0.000002

7        0.039187      2.7e-05
Convergence reached.
```

3.3 Golden-Section Search

In contrast to the methods discussed above, methods that repeatedly split the interval of interest in two parts and pick one, which is in turn split into two parts and so on, tend to be slow. They have the advantage in that they are guaranteed to ensure consistent and steady progress towards the optimal point subject to technical conditions. A generic bisection algorithm starts with an interval $[a, b]$ and a third point c between a and b such that $f(c) < f(a)$ and $f(c) < f(b)$. A fourth point d within the interval $[a, b]$ is selected, and $f(d)$ is computed. If d is between a and c, and $f(d) < f(a)$ and $f(d) < f(c)$, then $[a, c]$ becomes the new interval and d becomes the new provisional minimum. If $f(c) < f(d)$, then the new interval becomes $[d, b]$, - c remains the provisional minimum, but the interval has been narrowed. A similar approach applies if d is between c and b. The bisection method used in practice is known as golden-section search, where the point d is chosen so that the width of the new interval is equal to that of the old one divided by the Golden Ratio $\phi \approx 1.618$ [**?** **?**]. The process is illustrated in Figure 3.1.

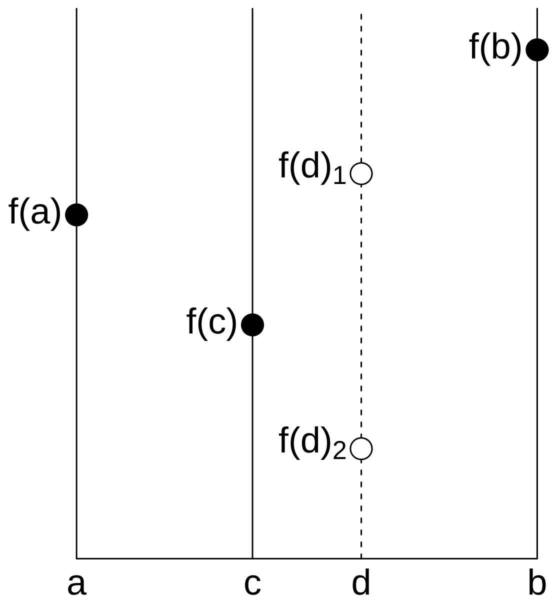


Figure 3.1: Golden-Section search. If $f(c) < f(d)$, the next triplet is given by $\{a, c, d\}$, otherwise $\{c, d, b\}$ is used.

3.4 Brent's Method

Brent's Method is a hybrid of successive parabolic interpolation and golden-section search [**?**]. If parabolic interpolation is failing to provide a sufficiently rapid decrease in the objective function, a bisection step is performed. While the bisection steps might not produce as much progress as the parabolic steps, they are certain to produce a consistent rate of improvement no matter how close the algorithm is to the optimal point. In contrast, parabolic interpolation is only certain to work if one is already within a neighbourhood of the optimal point as noted in Section 3.1.4. Brent's method will also perform a bisection step if the interpolating parabola is ill-conditioned, or if a bisection step has not been performed recently. The hybrid method is robust as a result of the golden-section steps, and the parabolic steps ensure it performs well when applied to smooth functions along with a decent starting value.

3.5 Estimation Of Parameters For a Standard Cauchy Distribution Using Brent's Method

To illustrate how Brent's method is employed in practice, it will be used on a straightforward estimation problem first. Consider the question of fitting a Cauchy distribution to some data. Given n observations $x_1, \ldots, x_n$ from an unknown Cauchy distribution, the likelihood function is given by:

$$L(\mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\pi\sigma \left[1 + \left(\frac{x-\mu}{\sigma}\right)^2\right]}$$

Attempting to maximise this likelihood entails solving a complex system of equations in μ and σ. Our purpose is to demonstrate that Brent's Method can tackle this problem without much difficulty. Brent's Method can only optimise a function in one dimension at a time, so it is necessary to attempt to optimise for μ and σ separately. The profile log-likelihood of σ is computed:

$$\ell(\sigma) = \sup_{\mu} \log(L(\mu, \sigma))$$

R can evaluate $\ell(\sigma)$ straightforwardly by using Brent's method to optimise $L(\mu, \sigma)$ with respect to μ and holding σ constant. The function $\ell(\sigma)$ can then in turn be optimised to find the optimal value of σ. This procedure is illustrated in Figure 3.2. One subtlety with optimising a Cauchy likelihood is that the likelihood function can have multiple local maxima since the likelihood function is the ratio of two multivariate polynomials in μ and σ. To ensure that the algorithm was sufficiently close to the MLE, the median was used as an initial estimate of μ, and half the interquartile range was used as an initial estimate for σ. Given these somewhat crude estimates, $\tilde{\mu}$ and $\tilde{\sigma}$, the the standard error of the median $\sigma_{\tilde{\mu}}$ is approximately given by:

$$\hat{\sigma}_{\tilde{\mu}} \approx \frac{1}{2f(\tilde{\mu}; \tilde{\mu}, \tilde{\sigma})\sqrt{n}},$$

where $f(x; \mu, \sigma)$ is the Cauchy density function with location parameter μ and scale parameter σ. The values $\tilde{\mu} \pm 2\hat{\sigma}_{\tilde{\mu}}$ are then used to provide the initial lower and upper bounds for the optimiser. The aim is to construct a confidence interval that is highly likely to contain the MLE for μ, but isn't so wide that the interval is in danger of containing multiple local maxima for the likelihood.

Not only can the likelihood be maximised without derivatives, asymptotic inference can be done without derivatives as well. Given the score function and the Fisher information at the maximum likelihood estimates, it is possible to compute an approximate confidence interval for σ and μ[**?**]. Instead of analytic methods, one can use finite differences to approximately compute the necessary derivatives to the desired degree of accuracy [**? ?**]. This was successful at producing a valid approximation for the profile likelihood, shown as a red dotted parabola in Figure 3.2. It is thus possible to compute a confidence interval using the score test. The test statistic $S(\sigma)^2/I(\sigma)$ could be accurately approximated using finite differences. One takes the value of σ for which the test statistic is less than or equal to the appropriate critical value from a chi-squared distribution. By inspecting the plot in Figure 3.3 and then solving for σ, an approximate confidence interval for σ can be computed such that σ lies in $(0, 2.20)$ with 95 percent confidence.

An important assumption underpinning such asymptotic confidence intervals is that the two term quadratic Taylor expansion based on the score and information functions is valid over the range of interest. This is not necessarily the case here as can be seen in Figure 3.3. There is a spike in the score statistic on the right caused by the Fisher information changing sign at approximately $\sigma = 2.35$. This indicates that the confidence interval might be wider than the range over which a quadratic approximation around the MLE is valid, and should perhaps be treated with some scepticism.

3.6 Robust ODE Parameter Estimation

If observations of values from an ODE are subject to heavy-tailed noise such as in the Cauchy case described above, least squares regression becomes unsuitable. An obvious candidate is L_1 regression, which attempts to minimise the sum of the absolute values of the residuals instead of the sum of the squared residuals. An important property of L_1 regression is that median is naturally associated with this approach; the sample median of a set of numbers is the value that minimises the L_1 error just as the sample mean is the value that minimises the least squares error [?][2]. L_1 regression can greatly complicate the process of estimation however, because the the function $|x|$ is not everywhere differentiable. This means that the usual gradient-based approaches to nonlinear regression such as gradient descent should not be applied. Even methods that attempt to numerically approximate the derivatives such as parabolic interpolation are either not guaranteed to converge quickly or entirely unsuitable.

Brent's Method can tackle such problems however, as it is robust against non differentiability. For nonlinear L_1 regression, the objective function tends to be piecewise smooth. Between the "kinks", the function is differentiable and amenable to parabolic interpolation. Once the bisection steps have reached a neighbourhood of the optimal value, parabolic interpolation will find the optimum reasonably quickly. Consider for example, the following ODE with $\beta = -0.5$:

$$\begin{cases} y'' - \beta(1-y^2)y' + y = 0 \\ y(0) = 1 \\ y'(0) = 0 \end{cases} \tag{3.4}$$

This ODE describes a non-linear oscillator, and is representative of quasi-linear mathematical models that can't be tackled by the `FDA` package or `Data2LD`. Note that this ODE is of the form $y'' + \beta(y)y' + y = 0$ with $\beta(y) = -\beta(1-y^2)$. By definition, the linear ODEs usually used in FDA cannot model systems where the $\beta(\cdot)$ terms have y as a dependent variable. Instead they can only model situations where the parameters vary with time alone (and/or space in the case of a linear PDE).

We wish to investigate the problem of estimating β from noisy observations. The `desolve` package [?] was used to numerically find the values of $y(t)$ at chosen time points $\{t_1, \dots, t_n\}$. The values of $y(t)$ at these points, were corrupted by random Cauchy noise and were independently sampled N times. This produced a set of nN observations: $\{y_{11}, y_{12}, \dots, y_{1N}, \dots, y_{n1} \dots, y_{nN}\}$. Because the data is heavy-tailed, least squares (L_2) regression is inappropriate. Instead, the goodness-of-fit associated with a given choice of β is measured by the sum of absolute errors associated with a given choice of β such that:

[2]This is discussed in more detail in Chapter 4.

$$SAE(\beta) = \sum_{i=1}^{K} \sum_{j=1}^{N} |y(t_i;\beta) - y_{ij}|$$

Here $y(t;\beta)$ denotes the solution of Equation 3.4 for a given choice of β. To evaluate $SAE(\beta)$ at a given value of β, it is necessary to use `desolve` to numerically find the values of $y(t_i|\beta)$. Brent's method was used to find the number $\hat{\beta}$ that minimised $SAE(\beta)$. The top panel of Figure 3.4 displays the realisation of $SAE(\beta)$, which is minimised at $\hat{\beta} \approx -0.25$. The bottom panel of Figure 3.4 shows the original curve, the generated data points corrupted with Cauchy noise, and the fitted curve generated by $\hat{\beta}$. It can be seen that the original curve and the original value of $\beta = -0.5$ are recovered with reasonable accuracy.

3.7 The Parameter Cascade and Brent's Method

Recall that the Parameter Cascade has three levels: (1) the inner problem, (2) the middle problem and (3) the outer problem. For the inner problem there is a given functional $J(f;\theta,\lambda)$ that takes a function f, and associated parameters θ and λ, and returns a real number. Usually, the function f is represented by a vector of coefficients $\mathbf{c}$, associated with a chosen set of basis functions. The function $\hat{f}(t|\omega,\lambda)$ that optimises $J(\cdot;\theta,\lambda)$ is then found. Outside of simple cases such as those discussed in Section 1.3.1, this problem cannot be solved analytically and is nearly always solved numerically by restricting the space of functions to the span of some set of chosen basis functions and optimising over that.

This in turn defines the middle problem, $H(\theta,\hat{f}(t|\omega,\lambda);\lambda) = H(\theta;\lambda)$, which is usually defined as the least squares error associated with the optimal f given θ and λ, i.e.

$$H(\theta;\lambda) = \sum [x_i - \hat{f}(t_i|\omega,\lambda)]^2$$

As suggested in Section 3.6 on fitting an ODE with Cauchy noise, the middle error function might be another loss function besides least squares (L_2) error such as the sum of absolute errors. As before, the value of θ that optimises $H(\cdot)$ holding λ constant, which is defined by $\hat{\theta}(\lambda)$, is computed.

Finally, the outer problem attempts to determine the value of λ that minimises the prediction error (generalisation error) by minimising another function that is defined by:

$$F(\lambda,\hat{\theta}(\lambda),\hat{f}(t|\hat{\theta},\lambda)) = F(\lambda). \tag{3.5}$$

There are several plausible choices for $F(\cdot)$. One could use leave-one-out cross-validation, one could partition the data set into a training set and a validation one and let $F(\lambda)$ be the associated error for the validation set, one could use generalised cross-validation. The chosen criterion is in turn optimised to find the optimal λ.

Note that the three levels are somewhat isolated from each other and only interact by exchanging parameters downwards and optimal values upwards. The middle function $H(\cdot)$ for example, only requires the value of the optimal $f(\cdot)$ evaluated at the chosen points t_i, and does not care about how these values were found or how $f(\cdot)$ is represented.

The inner problem consists of finding a function that minimises a chosen optimisation criterion for a given set of parameters. As previously discussed, the complexity of such problems can increase rapidly, requiring a considerable degree of expert knowledge and often must be developed from scratch if the differential penalty changes too much. It is thus desirable that the inner problem can be solved with already existing methods and tools such the FDA package or Data2LD package, as the effort associated with developing these tools from scratch is extensive. Ideally, it should be possible to utilise existing code that can compute $H(\cdot)$ and the optimal function as required.

There is thus a considerable degree of modularity present in the Parameter Cascade that is not fully investigated or exploited in Ramsay and Cao's paper [?], and research that inherits that framework. The Parameter Cascade can be adapted to solve problems for data exhibiting heavy-tailed noise for example, by using alternative loss functions for the various levels of the cascade. Not only is it good research practice to have mostly independent components that can be tackled and verified separately before being combined, it is also good practice from a software engineering perspective because the potential for complex interactions between different parts of the algorithm is reduced. This saves on debugging and testing requirements, which can be excessively high when implementing algorithms for FDA. In contrast, the Data2LD package is fairly tightly coupled. Rather than use R's built-in routines for example to optimise the associated middle problem, the authors wrote their own code. With the use of Brent's method as proposed in this thesis however, there is more separation between the levels of the cascade, which makes it very easy to build optimisation routines using already existing code. This substantially elides the cost and effort of tackling the inner problem and allows one to concentrate on the statistical questions such as fitting the model to data.

Melanoma Data

This derivative free optimisation strategy was applied to the melanoma dataset with a parameterised linear differential operator of the form:

$$L_\omega = D^2 - \omega^2 D^4. \tag{3.6}$$

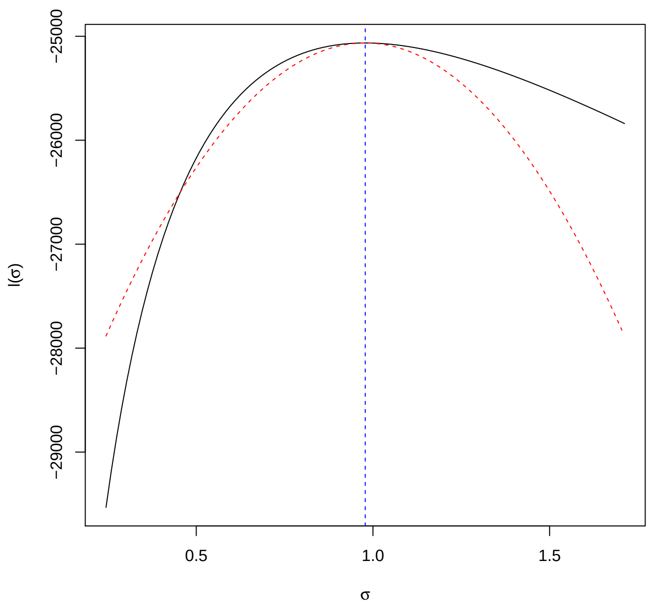
Both ω and λ will be estimated, so this is a three-level Parameter Cascade estimated without the need for calculating derivatives. As already noted, the `Data2LD` package can only implement a two-level Parameter Cascade, hence the proposed approach is an extension of the current state-of-the-art. The inner problem consists of finding the function $f(t)$ that minimises a penalised regression problem of the form:

$$PENSSE(f;\omega,\lambda) = \sum(x_i - f(t_i))^2 + \lambda \int |L_\omega f(t)|^2 dt.$$

The penalty term measures the extent to which a function lies outside of the span of the functions $\{1, t, \cos(\omega t), \sin(\omega t)\}$. The `FDA` package has optimisation routines that do the numerical work of fitting the data with the differential penalty given in Equation (3.6) for given choices of λ and ω, and then reporting the associated mean square error. Using Brent's method for the middle problem, the function $H(\omega; \mathbf{x}, \lambda)$ can be optimised with respect to ω for a given fixed λ. In the spirit of Equation (3.5), this defines a profiled objective function $F(\lambda, \hat{\omega}(\lambda), \hat{f}(t|\hat{\omega}, \lambda)) = F(\lambda)$. The outer objective function $F(\lambda)$ can be in turn optimised a second time using Brent's Method to estimate the optimal value of λ. Figure 3.5 plots $H(\omega; \mathbf{x}, \lambda)$ for a fixed value of λ, and plots $F(\lambda)$.

For ω, the top panel of Figure 3.5 shows that the SSE in this example is not particularly sensitive to small deviations from the optimal value even for fairly high values of λ. This suggests that the fitted curve will be adjusted to ensure no substantial increase in the error so long as ω isn't altered too much from the optimal value. Heuristically speaking, a flat objective function in the neighbourhood of the optimal point as can be seen in in this figure, increases the uncertainty in estimation because it is more difficult to argue that the optimal value is definitively better than adjacent ones. The loss function associated with a given fitting problem only approximates the 'true' loss function as the sample size goes to infinity. The bottom panel of Figure 3.5 shows that GCV curve has two critical points, with an asymptote as λ tends to infinity. The optimum value of λ is approximately 10^{-1}. When ω is optimised for the value of λ that minimises the GCV, Brent's method reports zero as the optimal value to within its default tolerance. Thus if λ is set too low, the optimal value of ω is numerically indistinguishable from zero.

A huge advantage of the proposed approach compared to Data2LD's use of derivative-based methods is that it allows for the use of more robust loss functions to be used at the various levels of the parameter cascade since no use at all is made of derivatives. As an example, suppose one wanted to choose ω to minimise the Median Absolute Deviation - median$(|y_i - \hat{f}(t_i|\omega, \lambda)|)$ - instead of the least squares (L_2) error. This loss function is chosen instead of the L_1 error for the sake of demonstration because the L_1 error might sometimes be tackled using a generalised version of gradient descent known as the sub-gradient method,[**?**] while getting any kind of a derivative for MAD is extremely difficult. To do the estimation with the MAD loss function becomes quite simple using Brent's method. One replaces the code that computes the L_2 error with a few lines of R code that computes the MAD and the optimisation routine is run again. The result of using Brent's method and the MAD loss function when fitting the melanoma data is seen in Figures 3.6 and 3.7. In this example, the MAD gives similar results to the usual least squares criterion, which suggests that both estimators are mutually consistent with each other.



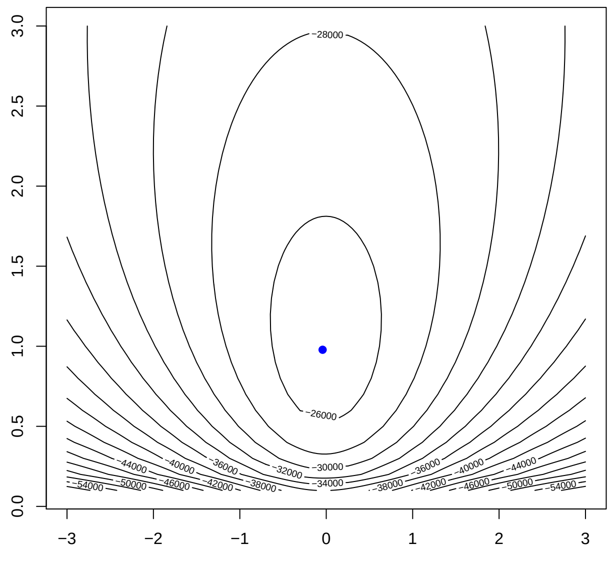


Figure 3.2: Profile log likelihood in σ, and contour plot of the joint log likelihood.

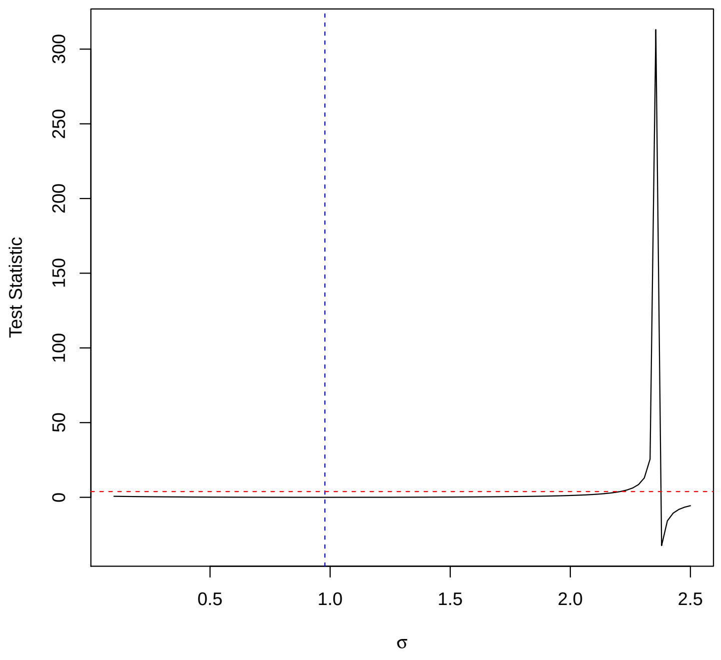
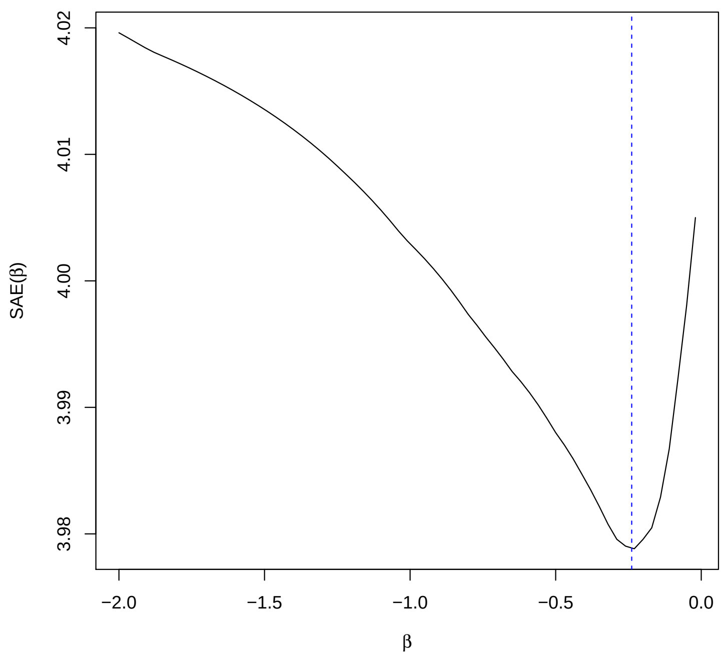


Figure 3.3: Plot of profile score statistic.



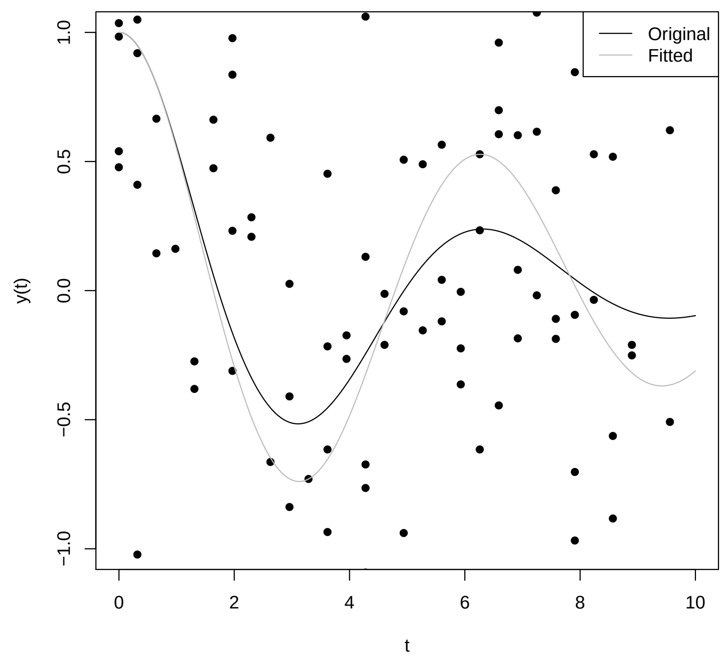


Figure 3.4: Top: Displays the SAE objective function; Bottom: Displays the true curve, the generated data corrupted with Cauchy noise, and the resulting fitted curve using Brent's method.

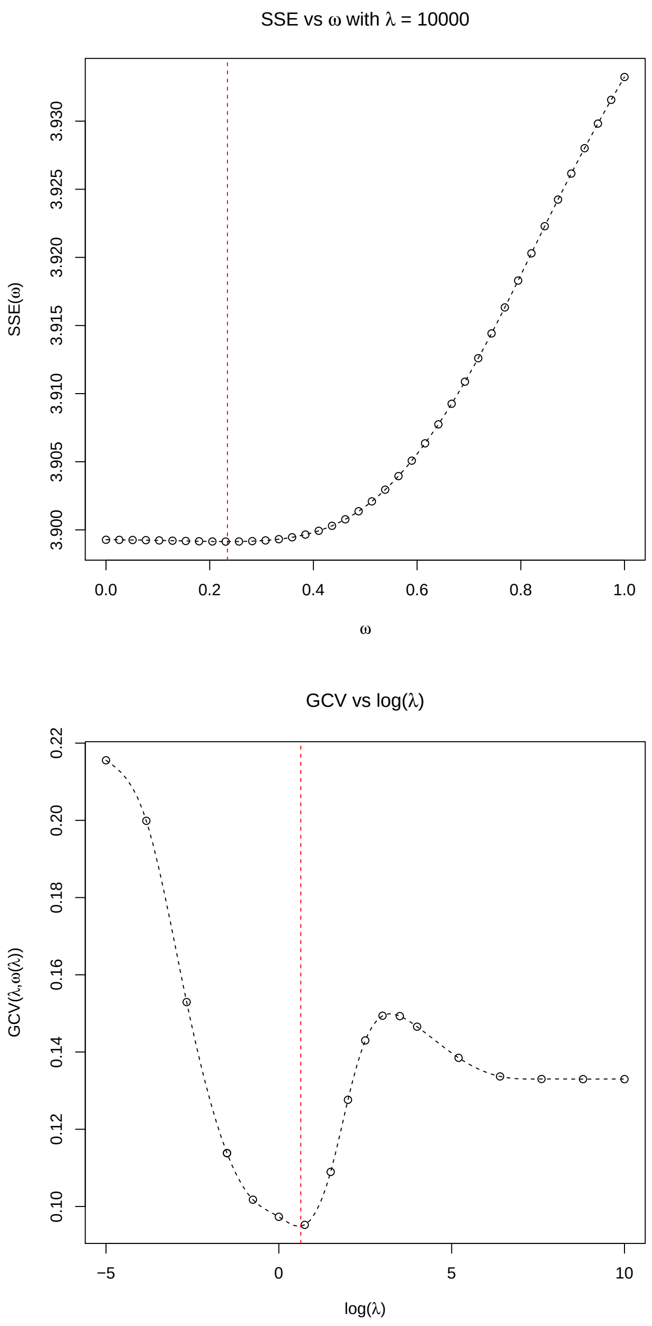


Figure 3.5: Plots of the middle and outer optimisation problems.

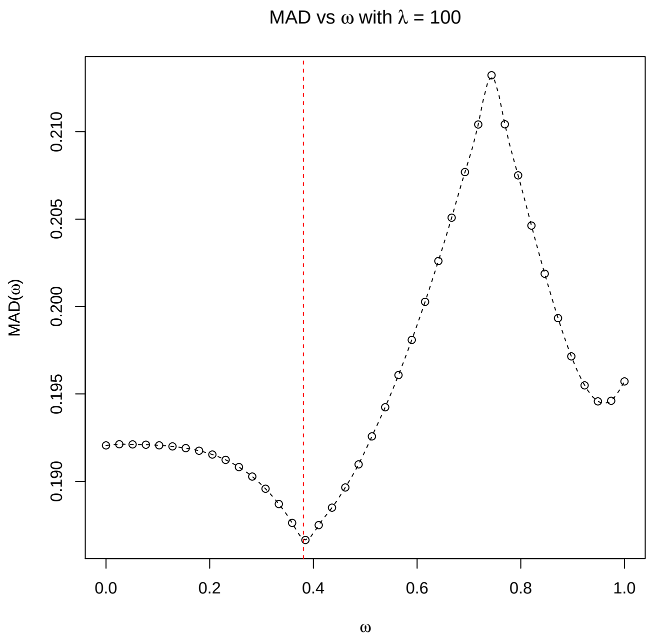


Figure 3.6: Plot of the middle optimisation problem with MAD used as a loss function

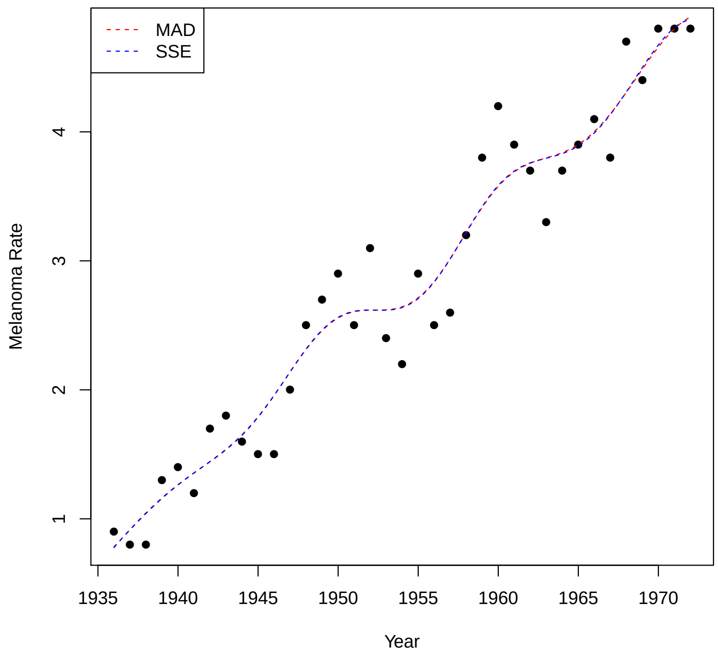


Figure 3.7: Comparison of fits for MAD and SSE criteria for middle problem

Chapter 4

A Two-Stage L_1 Parameter Cascade Using the MM Algorithm

In Chapter 3, Brent's method was introduced and was used to tackle Parameter Cascade problems where the middle level uses a loss function which is difficult to differentiate or has no well-defined derivative everywhere. It was remarked that Brent's method ensures that the different elements of the Parameter Cascade are loosely coupled from each other and this allows one to combine different fitting methodologies for different levels straightforwardly. In this chapter, these ideas are developed further. First, the L_2 based penalised fitting method is extended to the L_1 case. This new method is then used alongside Brent's method to implement a two level Parameter Cascade with L_1 loss functions at both levels.

4.1 L_1 Estimation for the Inner Problem

Brent's method is designed to optimise real-valued functions over a real interval. In Section 3.5, when the Cauchy likelihood was optimised, it was extended to functions that take more than one real argument by optimising over each coordinate (μ and σ) separately. However, there is no guarantee that this approach will perform well, and it can fail entirely for functions that have an exotic topography or multiple local optima arranged unusually[1]. Even in the best case, optimising over each coordinate generates its own optimisation sub-problem, which has the cumulative effect of increasing the running time of the algorithm. Brent's method further requires the specification of a bounding box that contains the optimal point since it uses bisection, which gets increasingly difficult as the number of dimensions increases. All of these considerations mean that Brent's method is highly unsuitable for performing L_1 fitting over a space of functions which tend to have a large number of dimensions. For the inner problem

[1]Consider for example the problem of finding the minimum of the function $f(x) = x\sin(x)$ over the interval $[0, 13]$. It is easy to see that the minimum is not on the boundary points of the interval because $f(0) = 0, f(6) = -1.67$, and $f(13) = 5.45$. Brent's method fails to find the minimum. It claims the optimal value is given by $f(4.9) = -4.81$ though $f(11) = -10.99$.

in the Parameter Cascade, by definition, there is one dimension introduced for each basis function used to represent f.

As a result a different approach will be employed for L_1 fitting at the inner level of the cascade. This approach a generalisation of the Iteratively Re-weighted Least Squares (IWLS) algorithm for computing the L_1 median of a set of N items $\{x_1, \dots, x_N\}$ to which an L_1 norm can be associated. The L_1 median is defined as the object x that minimises $\sum_{i=1}^{N} |x - x_i|$. We will start by describing how IWLS can be used to compute the L_1 median of a set of real numbers. We will further show that this as an example of what is known as an Majorise-Minimise (MM) algorithm [? ? ?], and then proceed to extend this MM algorithm to produce a modified Penalised Sum of Squares problem that can be iteratively solved and re-weighted to find the function that minimises a penalised L_1 norm.

4.1.1 An MM Algorithm For Computing the Median

Suppose that given a set of numbers $\{x_1, \dots, x_N\}$, the number x that minimised the L_1 distance between them is required. Then the fitting criterion, the Sum of Absolute Errors (SAE), is:

$$SAE(x) = \sum_{i=1}^{N} |x_i - x|$$

It is well known that $SAE(x)$ is minimised by the sample median of the numbers [?]. The usual approach to computing the sample median - sorting the numbers and taking the one in the middle - cannot be generalised to FDA problems. Therefore we will use a different approach. The main difficulty is that the function $SAE(x)$ is not everywhere differentiable, which means that the usual derivative-based techniques such as gradient descent or Newton's method can't be used. Instead an approach known as Majorise-Minimise or the MM Algorithm will be used to minimise the SAE [? ? ?]. For a given iterate x_n, a function $M(x|x_n)$ is required with the following properties:

$$\begin{aligned} M(x|x_n) &\geq SAE(x) \\ M(x_n|x_n) &= SAE(x_n) \end{aligned}$$

The function $M(x|x_n)$ is said to majorise $SAE(x)$. The next iterate x_{n+1} is then found as the value of x that minimises $M(x|x_n)$. Thus:

$$\begin{aligned} SAE(x_{n+1}) &\leq M(x_{n+1}|x_n) \\ &\leq M(x_n|x_{n+1}) \\ &= SAE(x_n) \end{aligned}$$

If such a function, $M(x|y)$, could be determined such that $M(x|y)$ is straightforward to minimise, it is then possible to easily produce a sequence of iterates x_n such that $SAE(x_{n+1}) \leq SAE(x_n)$ for all n. This pattern of monotone improvement in the objective function is similar to the EM Algorithm. In fact, the EM algorithm is a special case of the MM algorithm [?] [2].

The most important question associated with the MM algorithm is the construction of the majorising function. Once the majoriser has been found, the algorithm is easy to implement [? ?]. Verifying a potential majoriser is usually straightforward, however finding one in the first place is more difficult. The EM algorithm for example takes advantage of the probabilistic structure of the problem and Jensen's inequality. For an L_1 problem, the usual approach is to employ the Arithmetic Mean-Geometric Mean (AM-GM) inequality [?]. Only the AM-GM inequality in its simplest form is required here, i.e. that the geometric mean of two numbers is less than or equal to their arithmetic mean:

$$\sqrt{xy} \leq \frac{x+y}{2}$$

The AM-GM inequality is in fact a special case of Jensen's Inequality since the log function is concave:

$$\begin{aligned}\log(\frac{x+y}{2}) &\geq \frac{\log x}{2} + \frac{\log y}{2} \\ &= \log\sqrt{x} + \log\sqrt{y} \\ &= \log\sqrt{xy}\end{aligned}$$

It is possible to exploit the AM-GM inequality to majorise an L_1 regression problem by a weighted L_2 problem. The L_1 norm can be represented as a geometric mean, which then allows for the L_1 norm to majorised and separated by a weighted sum of squares. Given an iterate x_n, the AM-GM inequality implies that:

$$\begin{aligned}|y-x| &= \sqrt{(y-x)^2} \\ &= \sqrt{\frac{(y-x)^2}{|y-x_n|}|y-x_n|} \\ &\leq \frac{1}{2}\left(\frac{(y-x)^2}{|y-x_n|} + |y-x_n|\right)\end{aligned}$$

This in turn implies that:

[2] When applied to maximisation problems, MM instead stands for Minorise-Maximise. This case is the same except the surrogate function is required to be less than or equal to the objective function and it is maximised on each iteration. Thus, each iteration drives the objective function upwards.

$$\begin{aligned}\sum |x_i - x| &\leq \frac{1}{2}\sum\left(\frac{(x_i - x)^2}{|x_i - x_n|} + |x_i - x_n|\right)\\ &= \frac{1}{2}\sum\frac{(x_i - x)^2}{|x_i - x_n|} + \frac{1}{2}\sum\left(|x_i - x_n|\right)\end{aligned}$$

The L_1 problem is thus majorised by a weighted least squares problem. The $\frac{1}{2}\sum|x_i - x_n|$ term is constant with respect to x, so neglecting it makes no difference to the choice of x that is optimal. Likewise, multiplying the weighted least squares problem by a positive constant doesn't change the optimal value, so the $\frac{1}{2}$ term can be eliminated by multiplying by 2. The optimal value x_{n+1} can thus be found by minimising this weighted least squares score:

$$\sum\frac{(x_i - x)^2}{|x_i - x_n|}$$

The algorithm consists of finding the value of x that minimises the least squares error, inversely weighted by the residuals from the previous iteration.

4.1.2 Penalised L_1 Fitting

For the case of penalised regression, the Penalised Sum of Absolute Errors (PENSAE) is defined by:

$$PENSAE(f|\theta, \lambda) = \sum |x_i - f(t_i)| + \lambda\int |Tf|^2 dt$$

Here T is used instead of L to denote a differential operator that might not necessarily be linear.As before, this can be majorised by a weighted sum of a residual-weighted penalised sum of squared errors, and a $\sum|x_i - f_n(t_i)|$ term that can be ignored in the course of the optimisation. Then:

$$\begin{aligned}PENSAE(f) &\leq \frac{1}{2}WPENSSE(f|f_n, \theta, 2\lambda) + \frac{1}{2}\left(\sum|x_i - f_n(t_i)|\right) &(4.1)\\ &= \frac{1}{2}\left(\sum\frac{[x_i - f(t_i)]^2}{|x_i - f_n(t_i)|} + 2\lambda\int|Tf|^2dt\right) + \frac{1}{2}\left(\sum|x_i - f_n(t_i)|\right). &(4.2)\end{aligned}$$

To find the function that minimises the penalised L_1 error, one repeatedly finds the function that minimises $WPENSSE$ with the previous set of residuals used as inverse weights. This produces a sequence of fitted functions for which the penalised sum of absolute errors is monotonically forced downwards.

4.1.3 Discussion

The sequence of penalised errors $PENSAE(f_n)$ is monotone decreasing and but cannot be less than zero, so it is a bounded monotone sequence. As a result, the Monotone Convergence Theorem for sequences of real numbers ([**?**]) thus guarantees that a given generated sequence $PENSAE(f_n)$ will always converge to a limit. There are two caveats. First, the sequence might converge to a different point depending on the starting values and there is no guarantee that the sequence will converge to the lowest possible value of $PENSAE$. Second, there is no guarantee that the underlying sequence of functions will converge, and may just oscillate between several points. For example, the sequence $-1, 1, -1, \ldots$ does not converge but the associated sequence of absolute values $1, 1, 1, \ldots$ does.

This approach of associating the objective function with more standard problem that acts as a surrogate is employed in the literature on the EM algorithm. For example, in the introductory chapter of [**?**], the authors discuss how a multinomial estimation problem can be transformed into a binomial problem with missing data by artificially splitting one of the cells. They then construct a simple iterative EM scheme that can then be repeatedly iterated to estimate parameters for the original multinomial. They even remark that the surrogate problems associated with EM algorithms tend to be easy to solve using existing tools in the field. In a similar vein, the L_1 problem has been replaced here with a surrogate sequence of weighted L_2 problems that can easily solved using the `FDA` package. Since the `FDA` package does much of the heavy lifting, the necessary code for implementing penalised L_1 regression is brief.

Although the MM algorithm is simple to implement and good at tackling high dimensional penalised regression, convergence can be slow [**?**]. These claims are borne out when the convergence of the method is examined in Section 4.1.4 below.

4.1.4 Testing the Algorithm on the Melanoma Data

Since minimising $PENSAE$ is an optimisation problem over many dimensions, plotting the objective function to verify that the optimal function has been found isn't possible. Instead the MM algorithm described in Section 4.1.2 will be tested by applying it to the melanoma data perturbed by random noise. Further, the convergence of the algorithm for the original melanoma dataset will be examined. Figure 4.1 presents the L_1 and L_2 inner level fits to the melanoma data, which has been corrupted by Cauchy distributed noise. The value of ω was held fixed at a value of 0.3, which was chosen as the average of the two different estimates of ω presented in Figures 3.5 and 3.6 from the previous chapter. It is apparent from the Figure 4.1 that the MM fit is robust against outliers, tends to ignore more deviant points, and even manages to remain similar to the original fit. The L_2 fit tends to chase the heavy-tailed noise on the other hand. This is strong evidence that the curve that minimises $PENSAE$ has

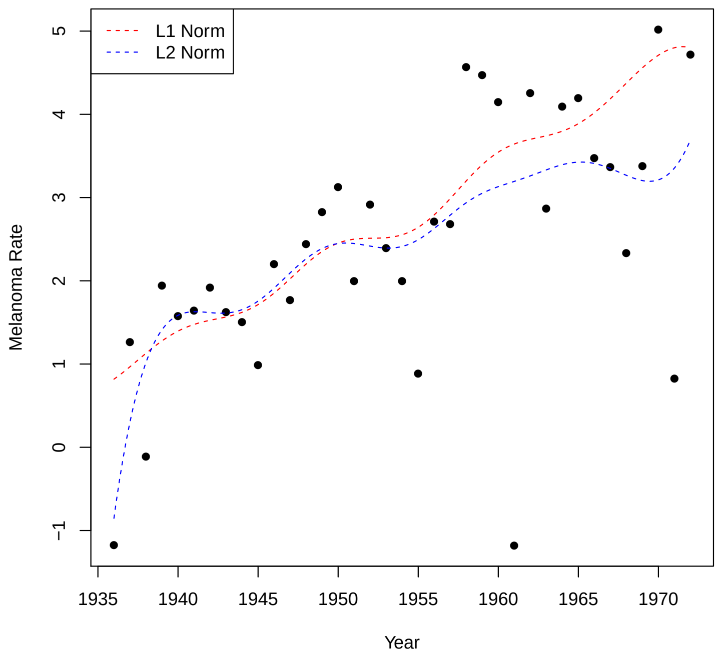
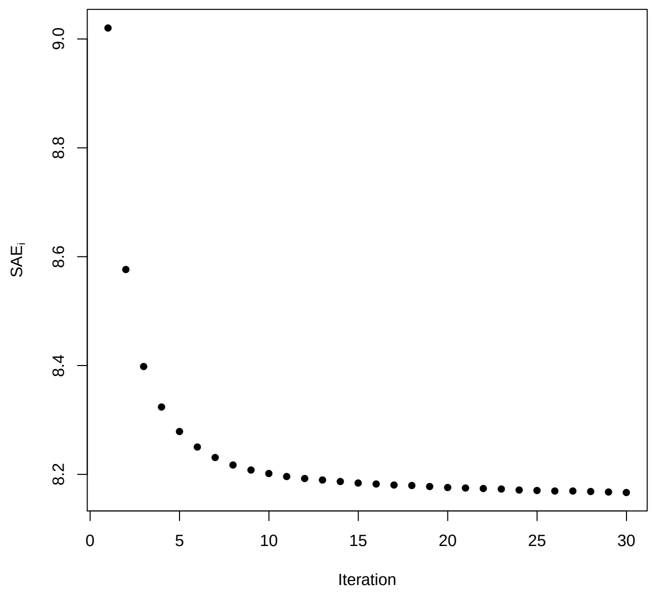


Figure 4.1: Comparison of L_1 and L_2 inner fits to Cauchy-perturbed data with ω fixed at 0.3

been found and that the method has been implemented correctly.

Figures 4.2 and 4.3 plot the convergence of SAE and $PENSAE$ over the course of the algorithm. Note that the PENSAE statistic doesn't quite converge monotonically as the theoretical analysis predicted. Instead, it fluctuates before settling down to the typical and expected pattern of monotone decline. Upon investigation, it was determined that over the first few iterations, the range of the weights applied to the observations on each iteration that were computed using the residuals from the previous iteration, grew very rapidly. By the fourth iteration, the lowest weight was equal to 1.48, and the highest was equal to 4.8×10^6. It seems that this rapid and large change produces qualitative changes in behaviour before the algorithm manages to 'burn in'. It is likely that observations with low weights are effectively being censored after a few iterations due to roundoff error. It was found that imposing a minimum threshold for the weights by adding a constant to all the residuals before proceeding to computing the weights smooths out this behaviour, but doesn't eliminate it entirely.

Figure 4.4 plots the convergence of the coefficient vectors $\mathbf{c}_n$. This log-plot suggests that the sequence of fitted coefficient vectors $\mathbf{c}_n$ converges linearly since $\|\mathbf{c}_{n+1} - \mathbf{c}_n\| \approx C\|\mathbf{c}_n - \mathbf{c}_{n-1}\|$ as $n \to \infty$.



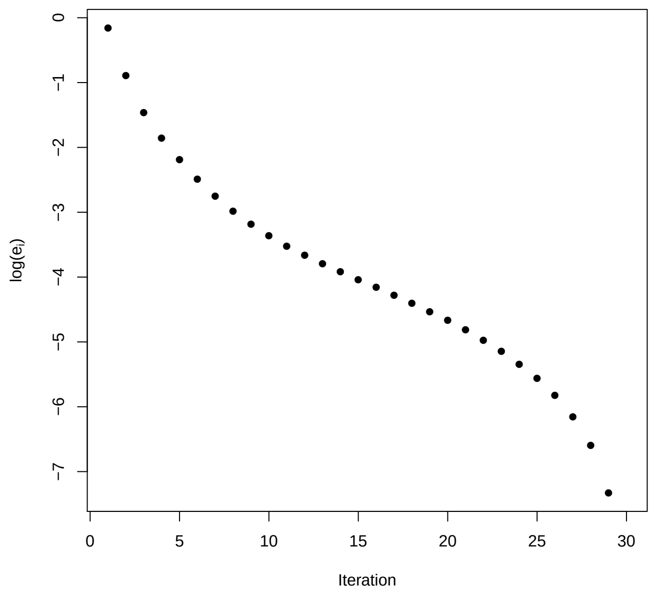


Figure 4.2: Plot of values and log differences for SAE Statistic

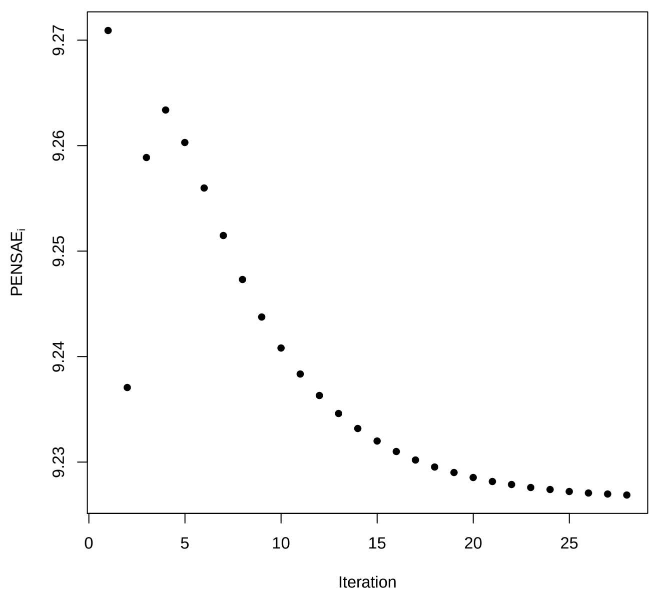


Figure 4.3: Plot of PENSAE statistic as the algorithm proceeds.

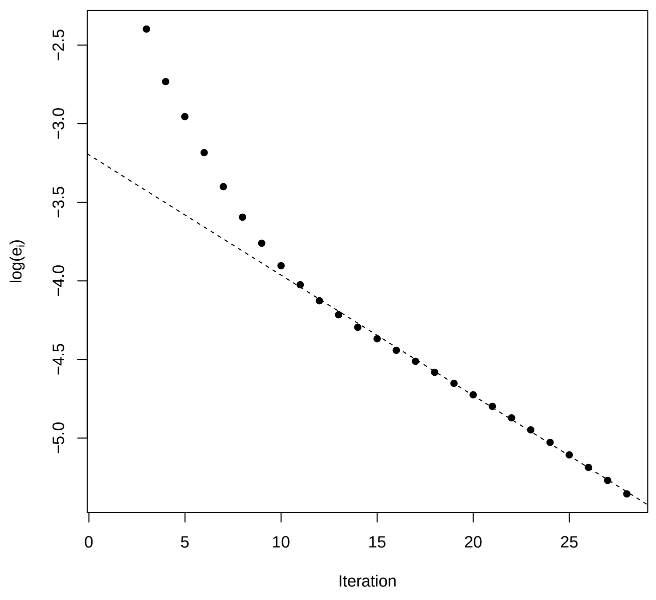
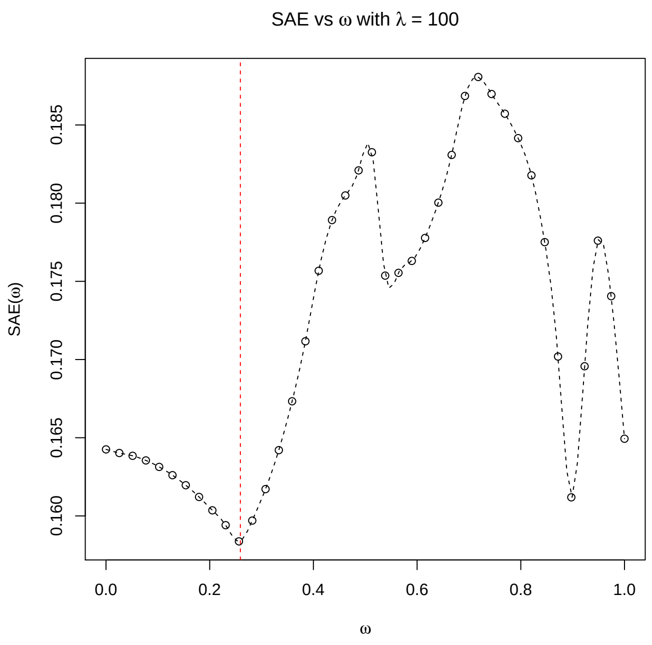


Figure 4.4: Plot of log norm differences for coefficients. Note that they tend to settle on a line.

4.2 The Two-Stage Parameter Cascade with L_1 Estimation at Inner and Middle Levels

The inner problem of the parameter cascade is a semi-parametric least squares regression model. The fitted function is modelled as a weighted sum of a solution to the differential equation (parametric), and a non-parametric residual. The λ term governs how big the residual is allowed to be relative to the the least squares error term. If the usual least-squares error function is used, the inner problem will struggle with outliers and heavy tailed errors as is the case for any form of least-squares regression. For high order differential operators like that used to model the melanoma data, there are many degrees of freedom associated with the differential operator's solution set. The ω and λ parameters don't strongly constrain the lower level of the cascade. There is thus little capacity for the higher levels of the cascade to restrain the lowest level through altering the λ and ω parameters and the Parameter Cascade algorithm must therefore use robust estimation at every level.

Chapter 3 discussed how Brent's Method can be used to tackle the middle problem without derivatives and then used this approach to optimise a highly irregular loss function. In Section 4.1, the MM algorithm was used to optimise the inner problem with an L_1 norm. Combining the two methods, it is very straightforward to implement a two-level Parameter Cascade with L_1 errors at both levels. Figure 4.5 displays the result of fitting a two level L_1 Parameter Cascade using the proposed approach to the original melanoma data. It can be seen that the $SAE(\omega)$ function is highly irregularly shaped, so we must be more careful that the optimisation routine doesn't get stuck on a local minimum. In Figure 4.6 both the L_1 and L_2 fits to the melanoma data perturbed with Cauchy noise are shown, where the same loss function is used respectively for both levels. Figure 4.7 plots the results of applying the L_1 and L_2 Parameter Cascades to the original and perturbed melanoma data, alongside mixed versions where the L_1 loss function is used for the inner fitting and L_2 loss function for the middle fitting and vice versa. It can be seen that using the L_1 loss function for the inner criterion is more robust than the L_2 loss function. The penalty term applied to the inner fitting only has limited capacity to stop the least squares loss function from chasing outliers.



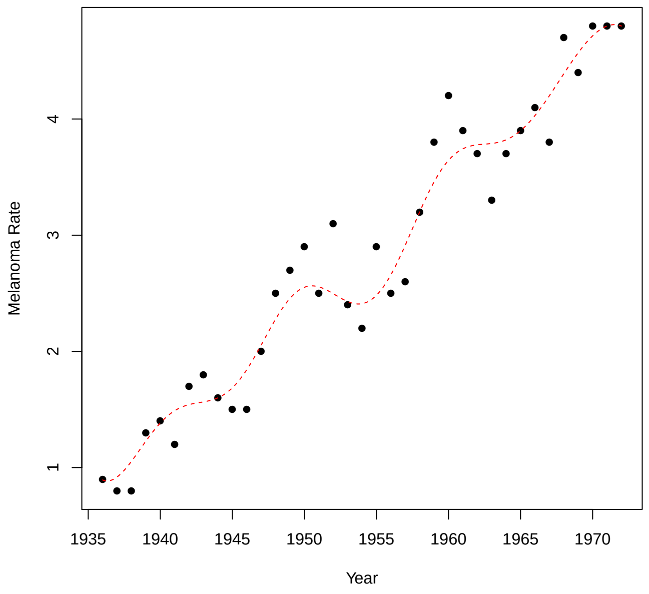


Figure 4.5: Fitting an L_1 Parameter Cascade to the Melanoma Data

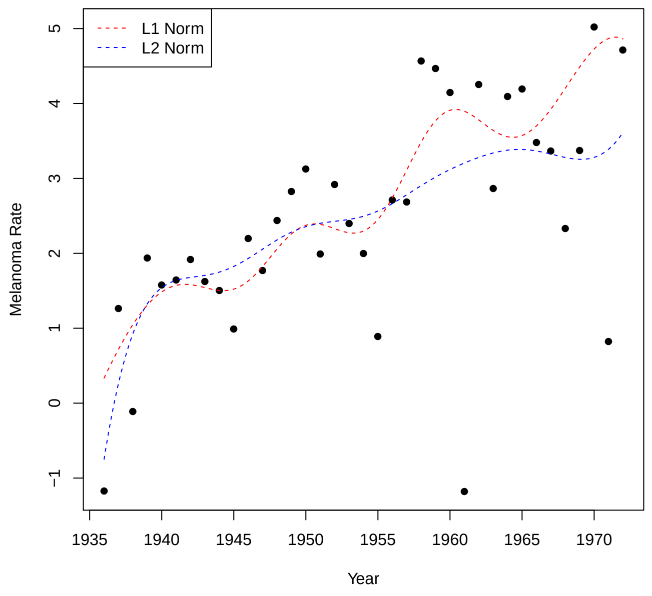
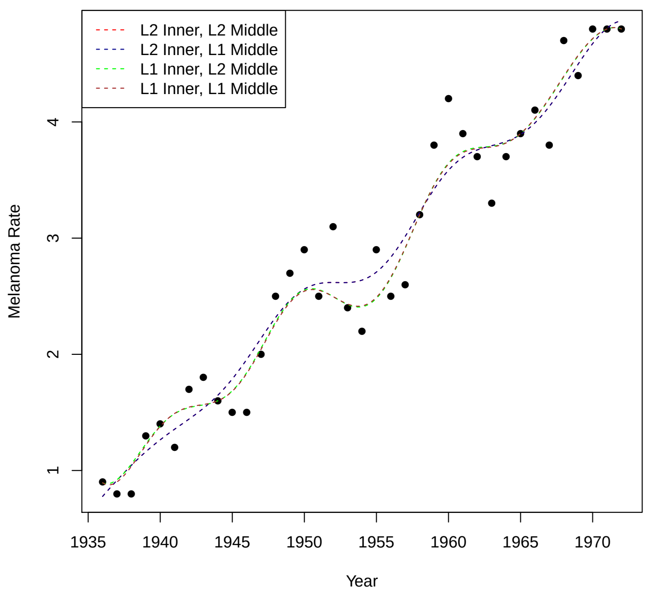


Figure 4.6: L_1 and L_2 Parameter Cascades with the same perturbed data as in Figure 4.1. Compare the L_1 curve in this plot with the one in Figure 4.5.



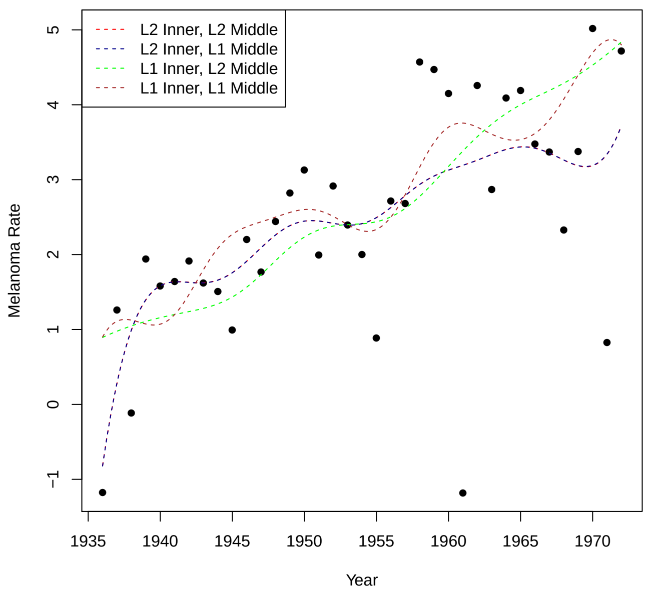


Figure 4.7: All possible combinations of L_1 and L_2 loss functions that can be used for the Parameter Cascade. The top plot applies them to the original melanoma data, the bottom to the same perturbed data as in Figures 4.1 and 4.6

4.3 Accelerating The Rate of Convergence

The MM Algorithm is very sluggish, and this is a well known weakness of both itself and the EM algorithm. However [?] suggests that this problem could be easily ameliorated in this particular case because of a special feature present. In practice, fitting the full model is not required and the computation of an associated summary statistic that determines how good a given choice of parameter is, is all that is needed. For our purposes, it will often be the case that only the value of $PENSSAE$ or GCV or SAE associated with a given choice of parameters is required as inputs to an optimisation routine, and it is not desirable to iterate until all the parameters for the full model converge if this effort can be avoided. In all the situtations in Chapter 3 where the Melanoma data was fitted for example, we generally didn't care about the coefficient vector computed at the lower level of the optimisation as such, and were only interested in it insofar as it could be employed to compute the loss function for a given parameter choice.

MacLanan and Krishnan ([?]) discuss the situation where one only wants to compare the likelihoods between a restricted model and a full model. They suggest the use of sequence acceleration methods to rapidly extract the likelihoods instead of running the EM algorithm to completion since the full models aren't needed. Wu ([?]) claims that acceleration methods for the EM algorithm translate quite easily to the MM case. On this basis, we explored whether this approach might be applied here to speed up the proposed estimation procedure.

The approach employed is known as Aitken Acceleration [? ?]. Suppose that there is a sequence $x_0, x_1, x_3, \dots$ converging to a limit x^*. Aitken's method makes the ansatz that $x_{n+1} - x^* \approx C(x_n - x^*)$ for some constant C. Many iterative algorithms in statistics exhibit this pattern as discussed in Section 4.1.3. This suggests the following equation:

$$\frac{x_{n+1} - x^*}{x_n - x^*} \approx \frac{x_n - x^*}{x_{n-1} - x^*}$$

Solving for x^* gives the accelerated sequence. There is an equivalent definition that is easier to generalise as given in [?]. Consider a sequence defined by functional iteration so that $x_{n+1} = F(x_n)$ for some function $F(\cdot)$. Define the error sequence by $e_n = x_{n+1} - x_n = F(x_n) - x_n$. The function $g(x) = F(x) - x$ returns the error associated with any value, and the limit of the sequence satisfies $g(x^*) = 0$. Suppose that the inverse of $g(x)$ in known and denoted by $h(e)$. Then x^* could be found by evaluating $f(0)$. The next best thing would be to use the values of the sequence to approximate $h(e)$, and then evaluate this approximate function at zero instead. The Aitken method approximates $h(e)$ by linear interpolation between (e_n, x_n) and (e_{n-1}, x_{n-1}), and then evaluates this approximation at $e = 0$.

4.3.1 Example: Using Imputation to Fit an ANOVA Model With Missing Data

For illustrative purposes, we will make use of an example from Chapter 2 of [?]. The authors discuss fitting an ANOVA model to a factorial experiment where some of the values are missing. They proceed by using the fitted model to estimate the missing values, fitting the model again with the new imputed values, and using the new fitted values in turn to update the estimates of missing values. The process is repeated until convergence. In the text, the authors do not work with likelihood or any probabilistic models and treat the question as a regression problem. This is similar to our L_1 fitting problem. The authors' example was implemented in R. For each iteration, the SSE statistic was computed. This defines an associated sequence $\{SSE_1, SSE_2, \dots, SSE_n, \dots\}$. Applying Aitken's method to this sequence produces a new sequence $\{ASSE_n\}$. As can be seen in Figure 4.8 and Table 4.1, the accelerated sequence converges far more quickly to the limit of the $\{SSE_i\}$ sequence than the original sequence.

4.3.2 Generalisations

Exploring more powerful methods than Aitken's method can be justified in two circumstances. The first is that if one is running the algorithm over and over again such that an increase in speed over many iterations means the effort invested is worth it. This might be the case for example if one wanted to use the bootstrap to model the distribution of a likelihood ratio statistic computed using the EM algorithm as previously described. The second is if the sequence is difficult to accelerate. In the context of accelerating the MM fitting algorithm introduced in Section 4.1, it shall be seen that both conditions apply.

As a field of study, sequence acceleration is closely related to time series analysis. A generic first order autoregressive model is given by

$$x_{n+1} = f(x_n, n) + \epsilon_n.$$

Consider the case where there are both no random errors so that ϵ_n is always zero, and the sequence converges to a limit. Here, the problem of determining the long term value of the sequence from a set of observations is equivalent to that of accelerating the sequence. If the specific form of $f(x_n, n)$ is known, there can often be a specific acceleration method that can exactly extract the limit. For illustration, suppose there were a sequence of the following form, but the parameters β_0 and β_1 were unknown:

$$x_n = \beta_0 + \frac{\beta_1}{n}. \tag{4.3}$$

As n goes to infinity, x_n converges to β_0. It is not difficult to show that the limit β_0 can be found by applying the following sequence transformation:

$$\begin{cases} \hat{\beta}_{1,n} = \dfrac{x_n - x_{n+1}}{\left(\frac{1}{n} - \frac{1}{n+1}\right)} \\ \tilde{x}_n = x_n - \dfrac{\hat{\beta}_{1,n}}{n}. \end{cases} \tag{4.4}$$

If the transformation in Equation (4.4) is applied to a sequence of values $x_1, x_2, \ldots, x_n, \ldots$ that is of form presented in Equation (4.3), then the transformed sequence $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n, \ldots$ will have the property that $\tilde{x}_n = \beta_0$ for all n. Likewise, the Aitken method is exact for sequences of the form $x_{n+1} = \beta_0 + \beta_1 x_n$, and so can be thought of as the deterministic analogue of an $AR(1)$ model.

The process of acceleration isn't quite as neat in practice because sequences don't adhere perfectly to these simple forms. Instead, the best that can be hoped for is that the transformed sequence converges to the same limit as the original one, but the rate of convergence is higher. For example, if transformation (4.4) is applied to a sequence of the form $y_n = \beta_0 + \frac{\beta_1}{n} + \frac{\beta_2}{n^2}$, then the transformed sequence is now of the form $\tilde{y}_n = \beta_0 + \mathcal{O}(\frac{1}{n^2})$, which converges to β_0 more quickly than the original sequence.[3] Suppose a convergent sequence is of the form $x_{n+1} = f(x_n)$ with $f(\cdot)$ differentiable and x^* is the limit. Using a first order Taylor expansion, it can be seen that for sufficiently large n, $x_{n+1} \approx x^* + f'(x^*)(x_n - x^*)$. In this case, Aitken acceleration has a decent chance of accelerating the sequence so long as it has 'burned in' sufficiently.

One generalisation, proposed in [?] is to use higher order polynomials to model the inverse error function $h(e)$. In this case $h(e)$ would be approximated by a quadratic through $(e_n, x_n), (e_{n-1}, x_{n-1})$ and (e_{n-2}, x_{n-2}). Making e the independent variable here instead of x means the estimated limit can simply be found by evaluating the approximating quadratic at $e = 0$ instead of finding the correct root of a quadratic to compute each element of the accelerated sequence. Another approach is to simply apply Aitken Acceleration to the sequence twice. Both of these generalised approaches were attempted for the missing data model described above, and the results can be seen in Table 4.1 and Figure 4.9. It can be seen that both methods improve convergence, though double Aitken acceleration is more effective (and easier to implement). One can take the process further. For the missing values linear model, these higher-order methods converge very rapidly and are prone to numerical instability thereafter due to the error terms being so small. If the Aitken method is applied three times to the original sequence, the first entry yields the limit immediately and there is no need to go any further. Applying the quadratic method twice in a row produces a new sequence for which the first entry is within 10^{-12} of the limit.

[3]Doing the algebra, it can be seen that it is now the case that $\hat{\beta}_{1,n} = \beta_1 + \beta_2 \left[\frac{2n+1}{n(n+1)}\right]$, and so $\tilde{y}_n = \beta_0 + \beta_2 \left[-\frac{2n+1}{n^2(n+1)} + \frac{1}{n^2}\right] = \beta_0 + \beta_2 \left[\frac{-n^3-n^2+n}{n^4(n+1)}\right]$.

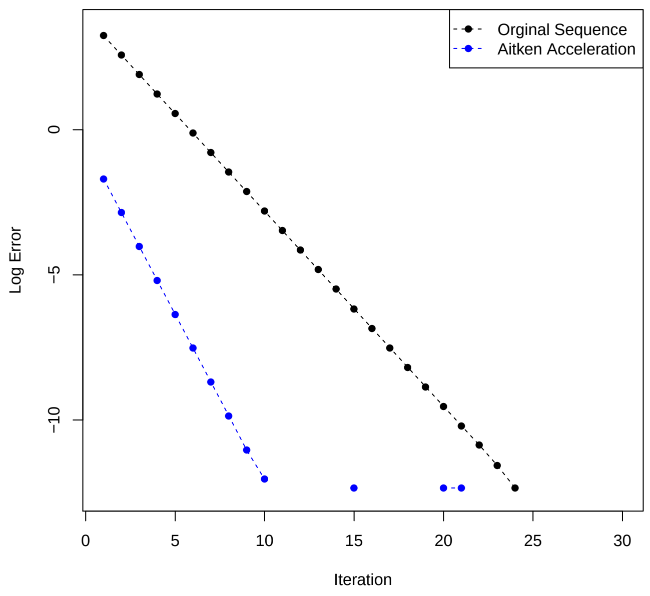


Figure 4.8: Log Errors for original sequence of SSE values and the accelerated one

Other Approaches: There are alternative approaches besides those described here. For example, the EM and MM algorithms generate a sequence of coefficient vectors $\{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n, \ldots\}$ with $\mathbf{c}_{n+1} = \mathbf{F}(\mathbf{c}_n)$ for some function $\mathbf{F}(\cdot)$. In our particular situation, the function $\mathbf{F}(\cdot)$ denotes the operator that takes a coefficient vector and returns the coefficient vector that minimises the associated $WPENSSE$ problem (4.1). The limit of this sequence - should it exist - is a solution to the equation $\mathbf{c} = \mathbf{F}(\mathbf{c})$. It is proposed in the literature to use Newton or Quasi-Newton methods as described in Chapter 3 and Appendix A to numerically solve this fixed point equation [**?** **?**]. The idea is that such methods will find the fixed point more rapidly than simply iterating $\mathbf{F}(\cdot)$ until one gets sufficiently close to the limit. These methods have the disadvantage of being more complex and time consuming to implement than the univariate acceleration methods.

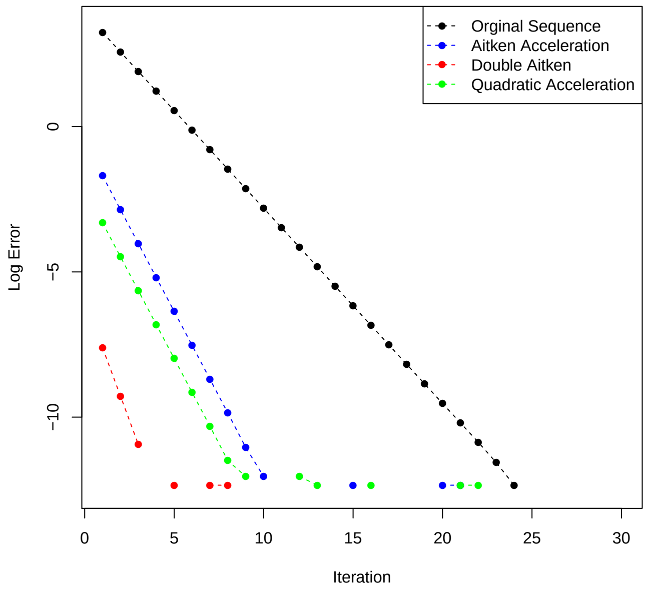


Figure 4.9: More sophisticated acceleration methods can provide a further boost to convergence. There are gaps in the plot because the more accelerated iterations have no valid log error since R cannot numerically distinguish them from the final limit.

Table 4.1: Iterations of the original sequence SSE_n, the accelerated sequence $ASSE_n$, the quadratically accelerated sequence $QASSE_n$, and the doubly accelerated sequence $DASSE_n$.

	SSE_n	$ASSE_n$	$QASSE_n$	$DASSE_n$
1	4949.6944444444	3203.8032711619	3203.7834325738	3203.7829457122
2	3575.1658950617	3203.7843303225	3203.7829788622	3203.7829457359
3	3282.7945625667	3203.7830400346	3203.7829479917	3203.7829457364
4	3220.5935028609	3203.7829521582	3203.7829458900	3203.7829457364
5	3207.3596279977	3203.7829461738	3203.7829457469	3203.7829457364
6	3204.5439391293	3203.7829457662	3203.7829457371	3203.7829457364
7	3203.9448588925	3203.7829457385	3203.7829457365	3203.7829457364
8	3203.8173952920	3203.7829457366	3203.7829457364	3203.7829457364
9	3203.7902754193	3203.7829457364	3203.7829457364	3203.7829457364
10	3203.7845052414	3203.7829457364	3203.7829457364	3203.7829457364
11	3203.7832775456	3203.7829457364	3203.7829457364	3203.7829457364
12	3203.7830163340	3203.7829457364	3203.7829457364	3203.7829457364
13	3203.7829607572	3203.7829457364	3203.7829457364	3203.7829457364
14	3203.7829489323	3203.7829457364	3203.7829457364	3203.7829457364
15	3203.7829464164	3203.7829457364	3203.7829457364	3203.7829457364
16	3203.7829458811	3203.7829457364	3203.7829457364	3203.7829457364
17	3203.7829457672	3203.7829457364	3203.7829457364	3203.7829457364
18	3203.7829457430	3203.7829457364	3203.7829457364	3203.7829457364
19	3203.7829457378	3203.7829457364	3203.7829457364	3203.7829457364
20	3203.7829457367	3203.7829457364	3203.7829457364	3203.7829457364
21	3203.7829457365	3203.7829457364	3203.7829457364	3203.7829457364
22	3203.7829457364	3203.7829457364	3203.7829457364	3203.7829457364
∞	3203.7829457364	3203.7829457364	3203.7829457364	3203.7829457364

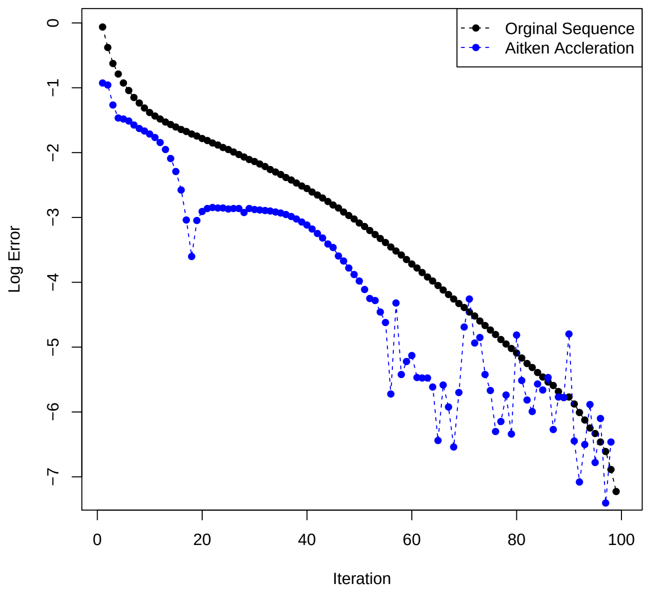


Figure 4.10: Accelerating the SAE sequence generating by the L_1 fitting algorithm using Aitken's Method. The improvement in covergence is mediocre.

4.3.3 Accelerating the L_1 Fitting Algorithm

The L_1 fitting algorithm devised in Section 4.1 is much more difficult to accelerate as can be seen in Figures 4.10 and 4.11. Figure 4.10 shows the result of applying Aitken acceleration to the SAE sequence generated, and Figure 4.11 shows the results of attempting numerous other acceleration methods, including approaches specifically designed for accelerating slowly converging sequences that Aitken acceleration cannot accelerate, e.g. Epsilon Algorithm and Lubkin's W transform [**?** **?** **?**]. No method performs substantially better than Aitken acceleration. The SAE sequence appears to be either numerically ill-behaved or of a very unusual form. To conclude, it is possible to save some time by acceleration, but the scope for doing so is limited and the process would have to be monitored carefully to ensure that numerical instability isn't an issue.

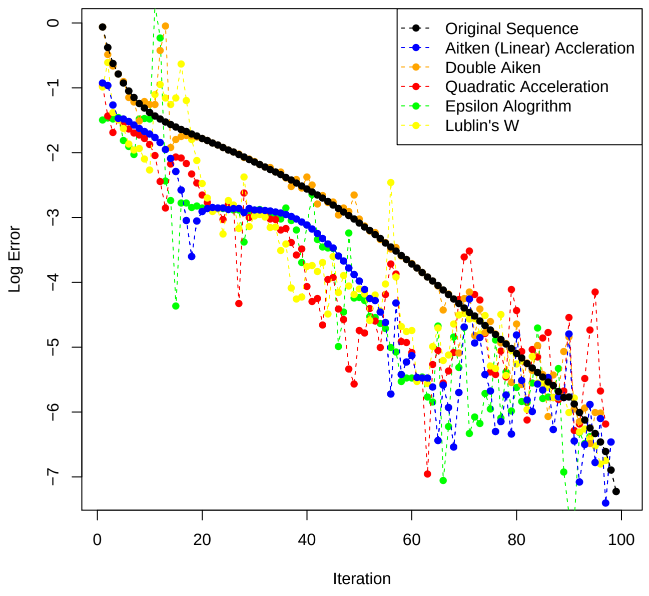


Figure 4.11: Accelerating the SAE sequence using multiple methods. Aitken's method performs the best, despite it's lack of sophistication.

Chapter 5

Profiling Non-Linear Regression Models With the Parameter Cascade

In previous chapters, the ODE models considered were linear. The work in this chapter generalises the results to non-linear ODEs and PDEs. We will discuss regression models that can be separated into an *inner problem* where one attempts to find the best fit given a choice of parameters, and a *middle problem* where one attempts to find the optimal choice of parameters. First a simple nonlinear regression model will be discussed as an example to introduce the ideas used in this chapter. Then, a Parameter Cascade method for fitting linear ODEs is devised. Finally, the Parameter Cascade approach will be used to fit PDEs. The approach used in this chapter here is related to an ODE fitting methodology developed by Ramsay et al [**? ?**]. The authors attempt to fit a parameterised ODE $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), t|\theta)$ using a penalised regression model of the form:

$$PENSSE(\mathbf{x}(t); \theta, \lambda) = \sum [\mathbf{y}_i - \mathbf{x}(t_i)]^2 + \lambda \int_0^T [\mathbf{x}'(t) - \mathbf{f}(\mathbf{x}(t), t|\theta)]^2 dt$$

The first term measures fit to the data, and the second measures fidelity to the ODE. This in turn defines a middle problem for θ:

$$H(\theta) = \sum [\mathbf{y} - \hat{\mathbf{x}}(t_i|\theta, \lambda)]^2,$$

where $\hat{\mathbf{x}}(t_i|\theta, \lambda)$ is the minimiser of $PENSSE(\mathbf{x}(t); \theta, \lambda)$ for fixed values of θ and λ. It is assumed that $\mathbf{x}(t)$ can be expanded as a linear combination of basis functions, and the authors proceed to conduct a traditional Parameter Cascade.[1] A sophisticated R package called `CollocInfer` has been developed to implement the method in a similar

[1]Not *entirely* traditional. To perform a fit, derivatives of $\mathbf{f}(\cdot)$ with respect to θ and $\mathbf{x}$ are required. `CollocInfer` uses finite differencing to estimate these derivatives if no information is provided by the user.

fashion as `Data2LD`.

The approach used in this chapter is more in the spirit of symmetry methods used to solve differential equations as described in [**?**]. The aim is to exploit structure in the specific dynamical system being modelled so that the problem of fitting the system is turned into a transformation model of some kind, which can then be fitted using a hierarchical approach. Our propsed approach can fit first-order linear PDEs with ease for example, because it exploits the fact that the solutions to these PDEs are constant along certain curves whose shape depend on the PDE at hand and the parameters.

5.1 Fitting a Non-Linear Regression Model With the Parameter Cascade

Consider the following non-linear regression model where observed values y_i are values observed at times t_i :

$$y_i = \alpha + \beta e^{\gamma t_i} + \epsilon_i. \tag{5.1}$$

This would be a straightforward linear regression problem if not for the unknown $e^{\gamma t}$ term. If γ were known, α and β could be found through simple linear regression with the $e^{\gamma t_i}$ term acting as an independent variable predicting the y_i. This suggests the following regression strategy. Define a function $H(\gamma)$ to be the sum of squared errors from performing simple linear regression on the y_i against $e^{\gamma t_i}$. That is:

$$H(\gamma) = \min_{\alpha,\beta} \sum [y_i - \alpha - \beta e^{\gamma t_i}]^2$$

This defines a middle problem, with the inner problem being that of minimising the simple linear regression problem given γ. The non-linear model can be estimated by using Brent's method to fit the middle problem. This approach was applied to simulated data with $\alpha = 100, \beta = 4$, and $\gamma = 1$, and the results can be seen in Figure 5.1. Things proceed as in the previous chapters, the parameter γ has been estimated with high precision, and the fit is of good quality.

5.2 Fitting Linear Homogeneous ODEs Using the Parameter Cascade

Recall a linear homogeneous ODE of order n is given by:

$$\frac{d^n y}{dt^n} = \sum_{k=0}^{n-1} a_k(t;\theta)\frac{d^k y}{dt^k}.$$

Under some mild technical conditions, the set of solutions to such an ODE is an n dimensional vector space and has a unique solution for each set of initial conditions. It is often more convenient to work with ODEs in matrix form. Any homogeneous linear ODE can be represented in matrix form:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}(t;\theta)\mathbf{y}.$$

For example, the ODE $y'' = -\omega^2 y$ with the initial conditions $y(0) = y_0$ and $y'(0) = v_0$ can be represented as:

$$\frac{d}{dt}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

with the initial condition $\mathbf{y}(0) = (y_0, v_0)'$. A basis for the solution set of any linear ODE can be formed from the set of solutions associated with the initial conditions $\mathbf{y}(0) = \mathbf{e}_i$, where $\mathbf{e}_i$ denotes the ith basis vector. This suggests the following cascade algorithm: given a set of parameters, find the set of solutions $\{y_1(t|\theta), \dots, y_n(t|\theta)\}$, where $y_k(t|\theta)$ denotes the solution with the kth basis vector as an initial condition. Then perform regression to fit the $\{y_i(t|\theta)\}$ to the observed data. The inner problem then consists of fitting a weighted sum of the $\{y_i(t|\theta)\}$ to the observed data and reporting the associated error given a choice of parameters, while the middle problem consists of finding the set of parameters that minimises this associated error.

For a problem where the ODE can be solved explicitly, things proceed as in Section 5.1. Consider again the ODE $y'' - \omega^2 y = 0$. The solutions generated by the initial conditions $(1,0)$ and $(0,1)$ is given by $\{A\cos\omega t + B\sin\omega t | A, B \in \mathbb{R}\}$. So the middle (least squares) criterion is given by:

$$H(\omega) = \min_{a,b} \sum [y_i - a\cos\omega t_i - b\sin\omega t_i]^2$$

Finding the optimal a and b given ω is an inner problem that can be solved using least squares regression as before. In fact, such a problem has already been encountered: the nonlinear model given in Equation 5.1 is associated with the ODE $y'' - \gamma y' = 0$. For ODE problems that cannot be explicitly solved, the trajectories $y_n(t|\theta)$ must instead be found by a numerical solver for each choice of θ. The inner problem then consists of linearly regressing the computed solutions against the observed data. To illustrate the method, it was applied to the following ODE with $\alpha = -0.3$ and $\beta = -1.0$:

$$y''(t) = \alpha\sqrt{t}y(t) + \beta\sin(2t)y'(t) \tag{5.2}$$

To minimise the middle problem, the Nelder-Mead method was used as Brent's method was was unsuitable because of the awkward topography. The results can be seen in Figure 5.2. The objective function has a complex topography, the method can fail to converge to the correct minimum if the sample size is too small or the variance is too high. In this case the parameters were sucessfully estimated and a good fit was producd.

The advantage of the Parameter Cascade here over a generic optimisation routine is that it is noticeably faster than trying to optimise everything in one go. The linear regression steps mean that the ODE needs to solved numerically fewer times, so that the algorithm runs approximately 30% faster for the ODE in Equation 5.2. However, the `nls` command is faster than the Parameter Cascade for Equation 5.1 even when no derivatives are provided. Compared to `Data2LD` and `CollocInfer`, this approach numerically estimates a basis for the solution set of the linear ODE and then projects the observed data onto the approximate solution set using the `lm` command. On the other hand, `Data2LD` and `CollocInfer` add penalties to the least squares criterion to ensure sufficient adherence to the ODE *post hoc.* This approach has the advantage of being much easier to implement, but it requires rigid adherence to the ODE, and cannot tackle non-linear problems like `CollocInfer`.

5.3 Estimation for First Order Linear PDEs

As discussed in Chapter 2, a similar framework can be used to perform estimation for PDEs in some cases. A complication is that for a PDE, the initial condition is a function rather than a constant. PDE problems cannot be tackled by either `Data2LD` or `CollocInfer`, while our approach exploits special structure to fit first order linear PDEs of the form:

$$\frac{\partial u}{\partial t} + c(x)\frac{\partial u}{\partial x} = 0$$

5.3.1 The Transport Equation

In Section 2.5 the Transport Equation was introduced and a fitting strategy was sketched out. Recall that the Transport Equation is defined by:

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} + \beta x \frac{\partial u(x,t)}{\partial t} & = 0 \\ u(x,0) & = f(x) \end{cases} \tag{5.3}$$

In Chapter 2, a middle objective function $H(\beta)$ to estimate the parameter β was defined, but no effort was made to fit the model. The objective function $H(\beta)$ was defined as the sum of squares:

$$H(\beta) = \sum [y_i - \hat{f}(x_i - \beta t_i)]^2.$$

And so:

$$\frac{\partial H}{\partial \beta} = -\sum 2 t_i \hat{f}'(x_i - \beta t_i)[y_i - \hat{f}(x_i - \beta t_i)]$$

To compute the gradient of $H(\beta)$, the estimates of the functions $f(x)$ and $f'(x)$ associated with a given choice of β are needed. This understates the difficulty however. The command `smooth.spline` will only return the GCV score, not the sum of squared errors. Therefore we are forced to use a more complicated objective function than least squares unless a routine to compute them is available. Fortunately, Brent's Method can be used to minimise $H(\beta)$ instead. As can be seen in Figure 5.3 the objective function is irregular, and care must be taken that one is close to the optimal value already. The estimated value of β is very close to the original value of $\beta = 1.5$.
Estimating $f(x)$ is harder than estimating β as can be seen in Figure 5.4. While 200 samples are sufficient to estimate β to a high degree of precision as shown in 5.3, 2000 samples are needed to estimate $f(x)$ to a similar degree of accuracy.

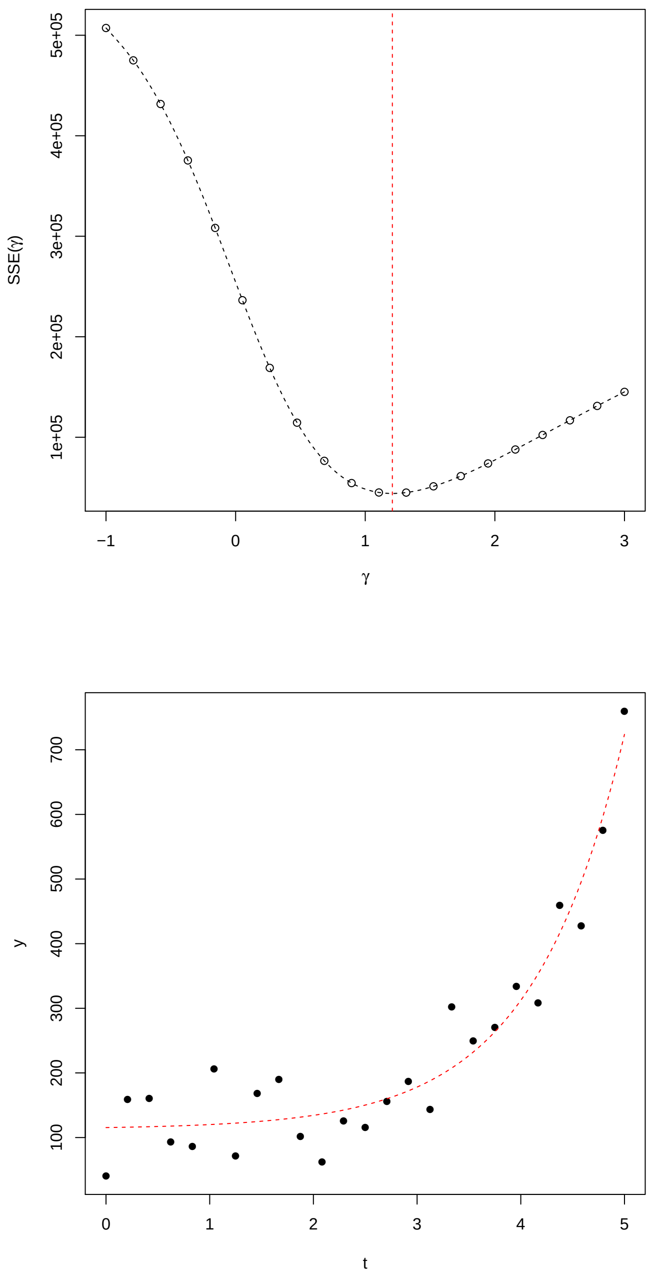
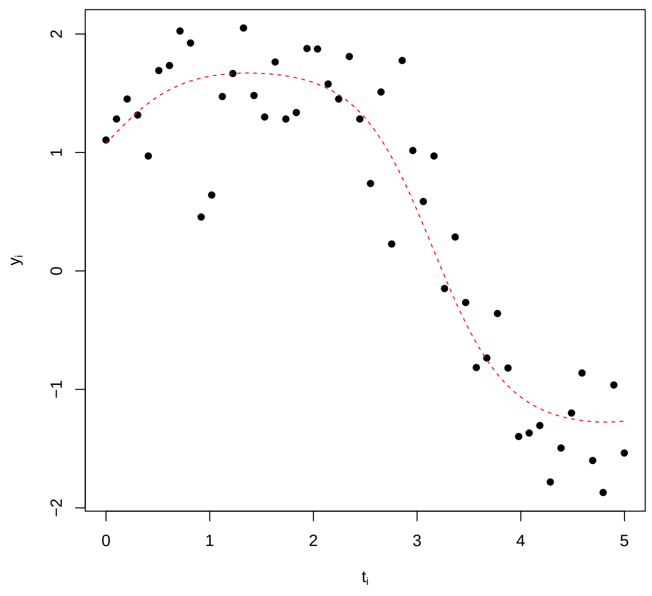


Figure 5.1: Profile Plot and Fitted Curve



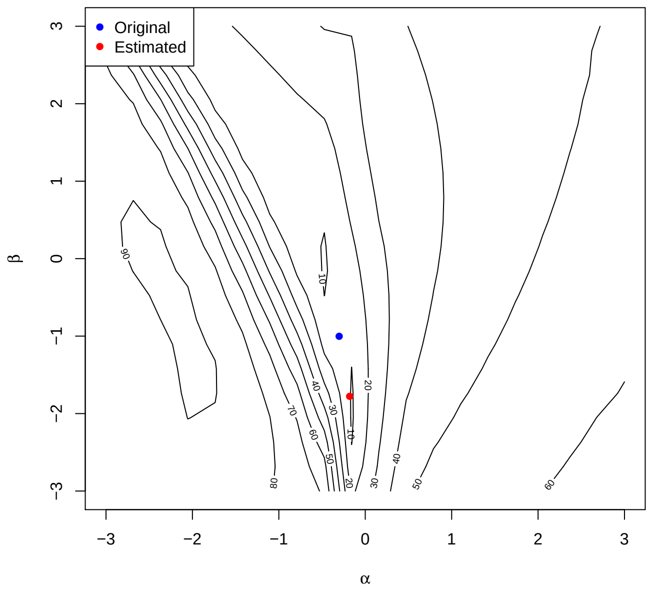


Figure 5.2: Plot of fit to simulated data, and contour plot of SSE against α and β. Blue dot is true parameter values, red is estimated parameter values.

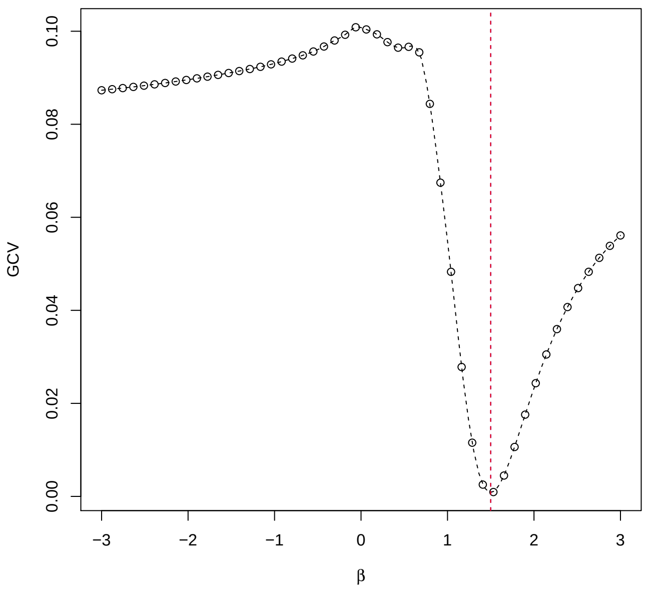
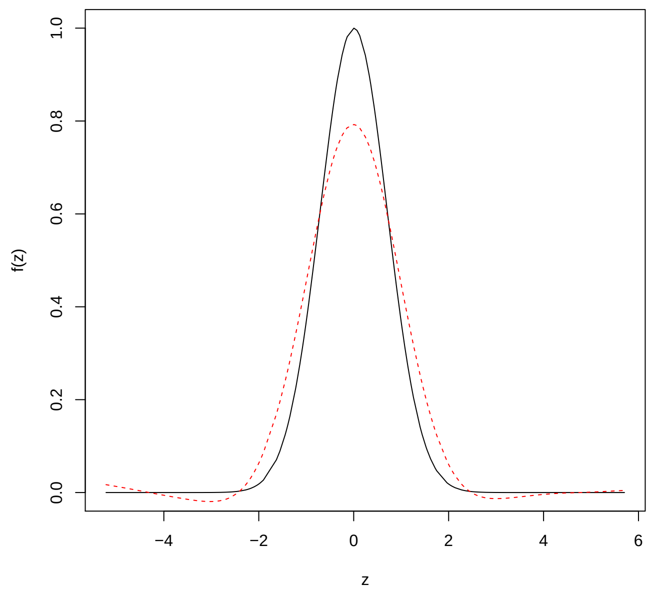
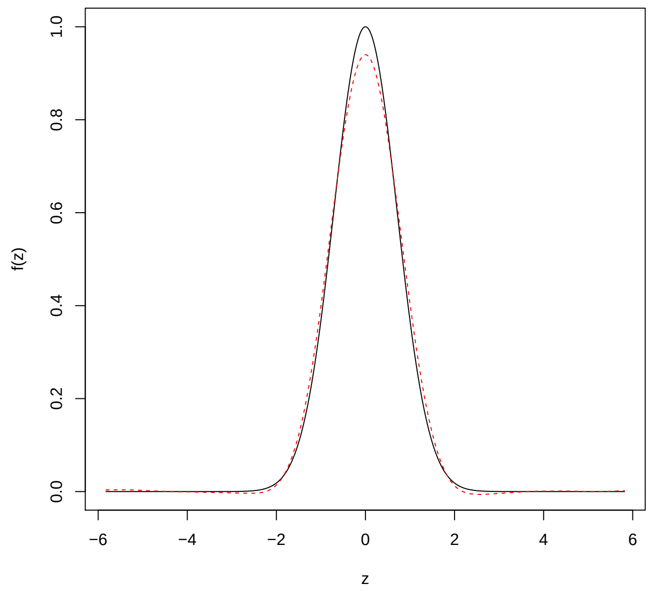


Figure 5.3: Plot of middle optimisation for the transport equation. Blue line denotes original parameter, red line denotes fitted estimate



(a)



(b)

Figure 5.4: The estimates of $f(x)$ computed for various sample sizes. The plot (a) gives the result with 200 sample points, the plot (b) gives the result for 2000 sample points. The variable $z = x - \beta t, z$ was used as the independent variable to avoid confusion between x and $x - \beta t$. Furthermore, the points used to fit $f(x)$ aren't displayed to reduce clutter and make it easier to compare the fitted curves with the original.

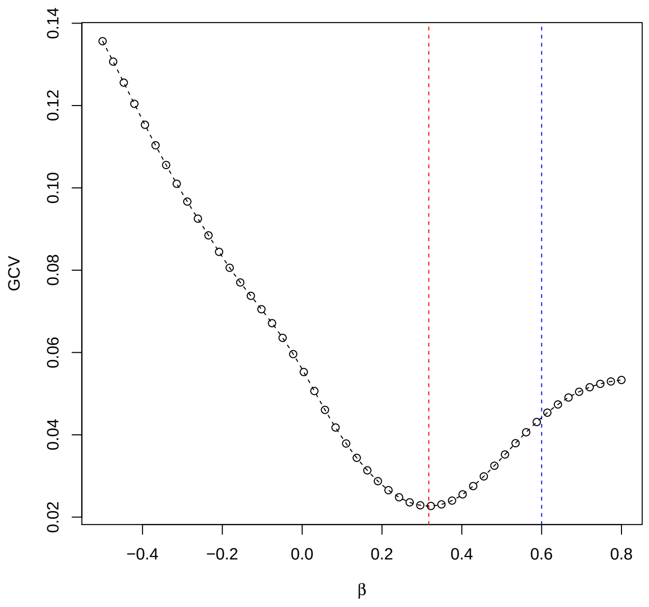


Figure 5.5: Plot of outer optimisation for the modified transport equation with only 30 sample points. Blue line denotes original parameter, red line denotes fitted estimate

5.3.2 The Transport Equation with Space-varying Velocity

The methodology will now be applied to a more complex PDE than the Transport Equation. Instead of having a constant velocity, the velocity will be allowed to vary with position by having the velocity equal to βx instead of β. This produces the following modified Transport Equation:

$$\begin{cases} \frac{\partial u(x,t)}{\partial t} + \beta x \frac{\partial u(x,t)}{\partial t} & = 0 \\ u(x,0) & = f(x). \end{cases} \tag{5.4}$$

The problem of estimating β for this PDE is ill-conditioned in the sense that `smooth.spline` will crash for some meshes. A more sophisticated non-parameteric fitting methodlogy is needed if our approach is to be consistently applied to first-order linear PDEs besides the Transport Equation.

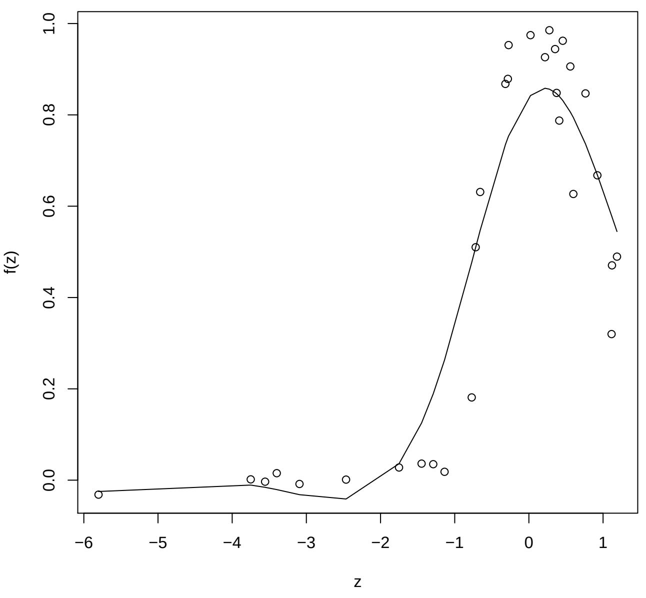


Figure 5.6: Plotted estimate of $f(x)$ for the modified transport equation with 30 sample points.

Chapter 6

Conclusion and Further Research

6.1 Derivative-Free versus Derivative-Based Methods

We have shown that use of derivative-free methods is highly effective for fitting data using non-smooth loss functions such as the sum of absolute deviations. A huge advantage is that it allows one to utilise existing code that only computes a goodness-of-fit statistic for a given choice of parameters. The derivative-free approach further allows problems that are not mainstream within FDA such as parametrically fitting PDEs without having to implement a complex solver like `Data2LD`. Additionally, the derivative-free approach allows for a considerable reduction in time spent on writing code to fit FDA problems. Brent's Method is much simpler than `Data2LD.opt`.

However, the methodology suffers from some weaknesses. It is undeniably slower than derivative-based methods. The MM algorithm for L_1 fitting suffers particularly badly from this weakness because it must solve many weighted $PENSSE$ problems in the course of its execution and it is difficult to accelerate. This must be balanced against the MM algorithm's ease of implementation.

6.2 Further Research

An obvious topic for further investigation is combining Brent's method with `Data2LD` so that the latter package can handle the parameter estimation, while Brent's method is used to find the optimal value of λ. Another is whether the MM algorithm for L_1 fitting can be combined with `Data2LD` to perform L_1 fitting where the parameters of the ODE are time varying functions. Finally, while the two level parameter cascade with L_1 errors was implemented, no effort was made to implement the three level parameter cascade. It has not been investigated by us whether the GCV is appropriate as a fitting criterion for λ in the L_1 case.

6.3 Quasi-linear Differential Problems

Throughout this thesis, it has been possible to use techniques from applied mathematics to construct solution strategies on a case by case basis, or use statistical methods to find a semi-parametric fit. As differential equations become more complex, both approaches begin to rapidly become non-viable. In this section, quasi-linear differential equation models will be briefly discussed to illustrate how small changes can greatly increase the difficulty of fitting.

The difference between a quasi-linear and a linear differential equation is that the coefficients in a quasi-linear equation are allowed to depend on the unknown function. Instead of an ODE such as $y' = \beta(t)y$, one would have an ODE such as $y' = \beta(y,t)y$. Though quasi-linear problems tend to be reminiscent of linear ones, they are nonetheless substantially more complicated, and require more technical knowledge to tackle. For a quasi-linear variation of a linear ODE, consider the Van Der Pol Equation:

$$y''(t) + \beta(1 - y(t))^2 y'(t) + y(t).$$

This ODE has no obvious solution. Even if a solution exists, an estimation strategy might be difficult to derive. For example, consider the inviscid Burger's Equation:

$$\frac{\partial u(x,t)}{\partial t} + \beta u(x,t)\frac{\partial u(x,t)}{\partial x} = 0$$

This equation is identical to the Transport Equation except that the rate term is equal to $\beta u(x,t)$. The solution is given by:

$$u(x,t) = f(s).$$

Here $f(\cdot)$ is an arbitrary function as before, and s is implicitly defined as the solution of the equation $x = \beta f(s)t + s$. Since $s = x - \beta ut$, this can be written as

$$u(x,t) = f(x - \beta ut).$$

Fitting this model is substantially trickier than the Transport Equation. There is no clear separation between the problem of estimating $f(\cdot)$ and β since $u(x,t)$ appears on the righthand side and scales β. A further complication is that $u(x,t)$ might only define a *relation*, instead of a function. There might be multiple values of u associated with a given (x,t) that satisfy the solution equation. Physically speaking, multiple values correspond to shock waves.

6.3.1 Discussion

We see that the level of knowledge required to devise fitting strategies can increase substantially even with seemingly modest increases in the complexity of the associated differential equation. The overall result is that as the complexity of the differential equation increases, more and more time will be needed to model it correctly.

Appendices

Appendix A

Overview of Optimisation Methods

This appendix provides an overview of the ideas in numerical optimisation used throughout this thesis. [? ?] are accessible texts for those who require more information. [?] is recommended since it covers line search methods in more detail than [?].

A.1 Gradient Descent and the Newton-Raphson method

The simplest derivative-based optimisation algorithm is known as *gradient descent*:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \mathbf{g}_n.$$

The fixed parameter $\alpha > 0$ controls how big a step the method will take on each iteration. Gradient descent has the property that the directions it generates will always point 'downhill' so that a small step will decrease the objective function:

$$\begin{aligned} f(\mathbf{x}_{n+1}) &= f(\mathbf{x}_n - \alpha \mathbf{g}_n) \\ &\approx f(\mathbf{x}_n) - \alpha \mathbf{g}_n^\top \mathbf{g}_n \\ &= f(\mathbf{x}_n) - \alpha \|\mathbf{g}_n\|^2. \end{aligned}$$

This means that $f(\mathbf{x}_{n+1}) < f(\mathbf{x}_n)$ so long as α isn't too big. Gradient descent is simple but only converges linearly under ideal conditions [? ? ?]. If the objective function isn't sufficiently ideal, gradient descent might only converge sublinearly [?].

Consider the question of minimising the function $f(x) = x^4$ using gradient descent, starting at $x = 0.5$ with $\alpha = 0.2$. The minimum of $f(x)$ is at $x = 0$, so the absolute value of the iterates x_n is a measure of the error. Figure A.1 plots the log errors for the first 20,000 iterations. It is apparent that the algorithm is converging sublinearly and that the rate of convergence is poor. It takes around 250 iterations before the error falls below 10^{-3}, approximately 1900 iterations before the error falls below 10^{-4}, and 13800 iterations before the error falls below 10^{-5}. The data suggests that the number of iterations needed to reduce the error to 10^{-n-1} is approximately 7.4 times the number of iterations needed to achieve an error of 10^{-n}. It could thus take over 100,000 iterations to get an error of less than 10^{-6}.

Gradient descent can be thought of as the naive choice if one only has access to $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$. Now suppose the second derivatives are also available. What would the 'obvious' choice be in this case? Perform a second-order Taylor expansion of the objective function around $\mathbf{x}_n$:

$$f(\mathbf{x}) \approx f_n + \mathbf{g}_n^\top(\mathbf{x} - \mathbf{x}_n) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_n)^\top \mathbf{H}_n(\mathbf{x} - \mathbf{x}_n).$$

The expression on the righthand side is minimised by $\mathbf{x} = \mathbf{x}_n - \mathbf{H}_n^{-1}\mathbf{g}_n$. Given the iterate $\mathbf{x}_n$, the *Newton-Raphson* method defines the next iterate by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}_n^{-1}\mathbf{g}_n.$$

The Newton-Raphson method converges quadratically, subject to technical conditions. The first condition is that one is already near the optimal point, while a second is that the Hessian at the optimal point is not pathological [**?** **?** **?** **?** **?**]. If these assumptions do not hold, undesirable behaviour can occur. For example, the Newton-Raphson method only converges linearly with rate 0.666 when applied to the function $f(x) = x^4$ as can be seen in Figure A.2. This is because the second derivative is zero at the optimal point [1][**?**]. While the Newton-Raphson method is a huge improvement over gradient descent in this case, it nonetheless fails to achieve the usual quadratic convergence.

The biggest weaknesses of the Newton-Raphson method are: (1) the cost of repeatedly computing the Hessians, (2) the possibility that $-\mathbf{H}_n^{-1}\mathbf{g}_n$ fails to be a descent direction, and (3) the possibility of poor performance or even divergence if one is far from the optimal value.

[1]For the case of univariate optimisation, the specific rate of convergence is given by $m/(m+1)$, where m is the number of consecutive higher order derivatives starting from the second derivative that are zero at the optimal point.[**?**] In the case where $f(x) = x^4$, we have that $f''(0) = 0$ and $f'''(0) = 0$, but $f^{(4)}(0) = 24 \neq 0$, and so $m = 2$.

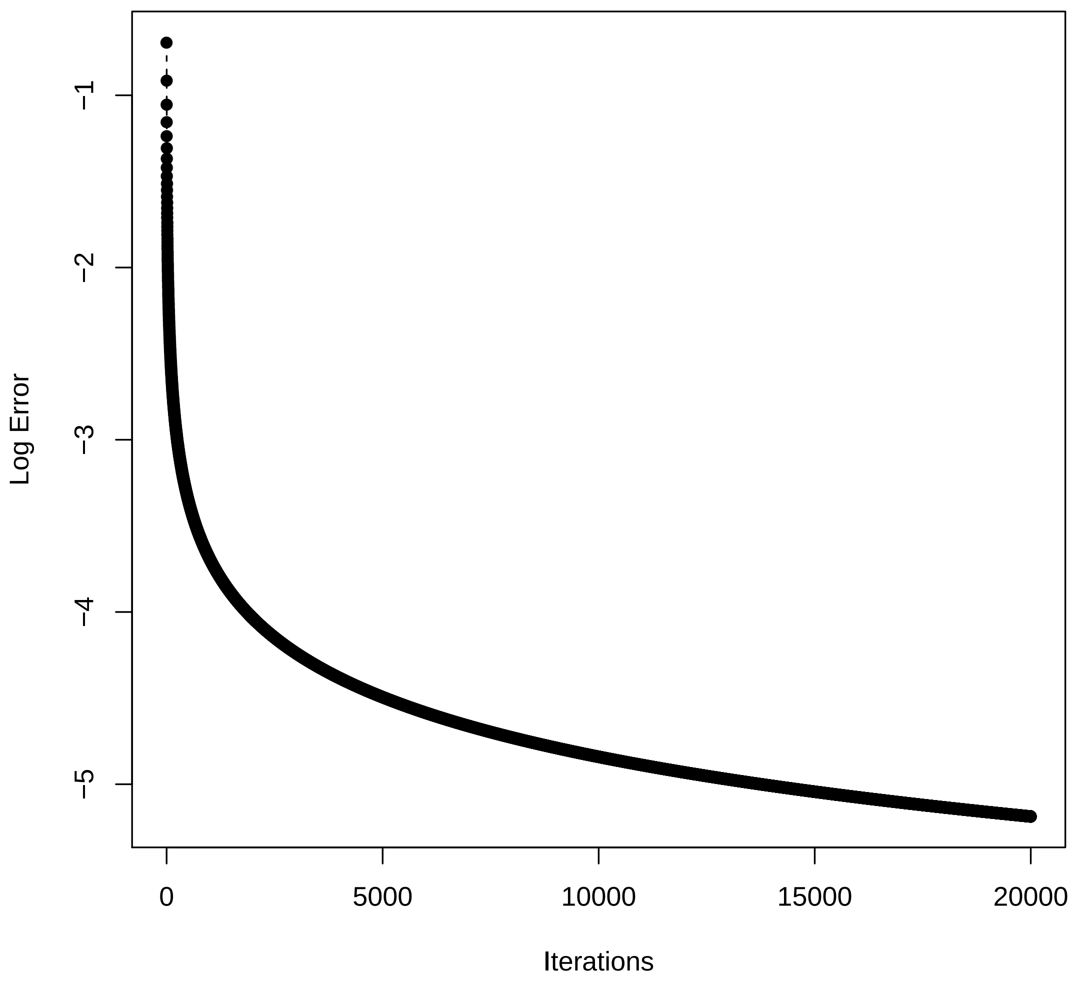


Figure A.1: Plot of the log errors from applying Gradient Descent to the function $f(x) = x^4$.

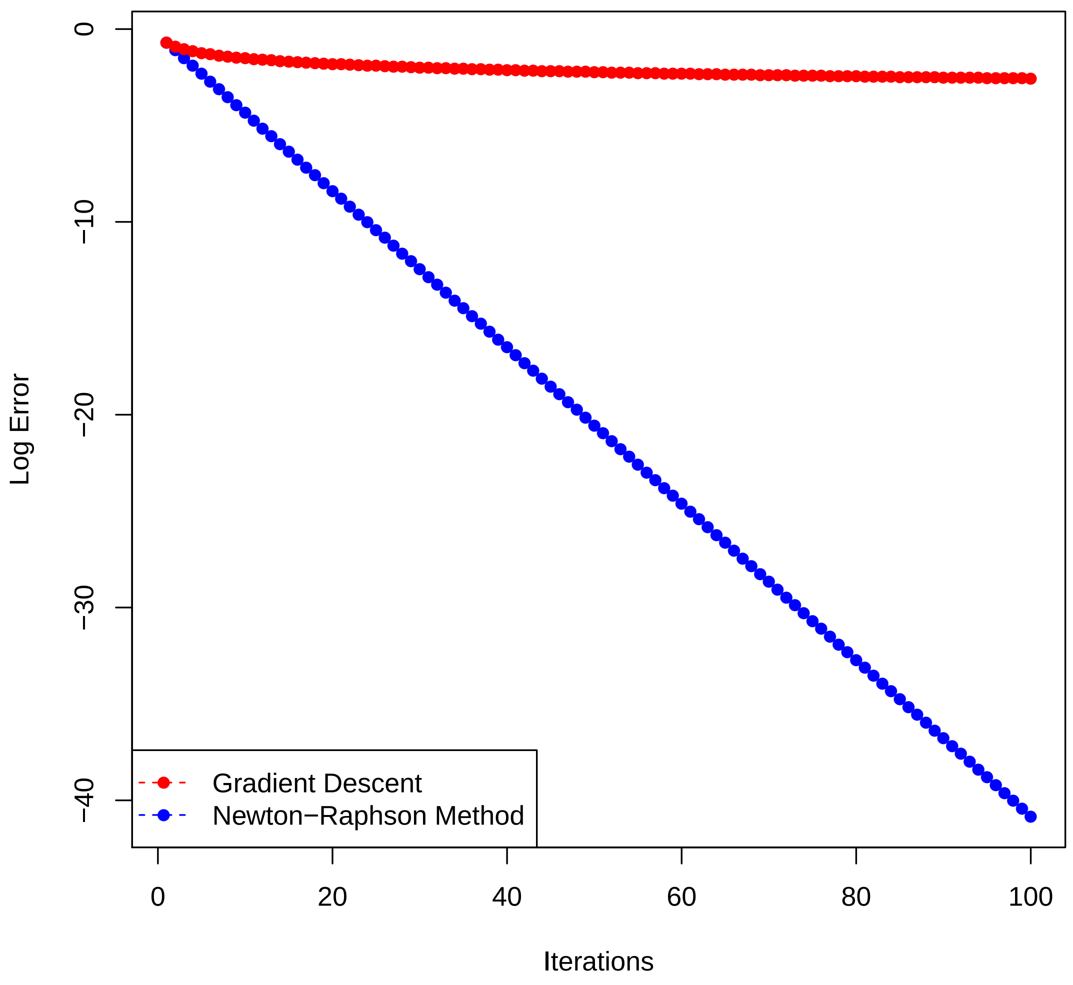


Figure A.2: Plot of the log errors from applying Gradient Descent and the Newton-Raphson Method to the function $f(x) = x^4$.

A.2 The Chord Method

Chord methods attempt to approximate the Hessian matrix by using a constant matrix $\mathbf{Q}$ and each iterate is defined by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{Q}\mathbf{g}_n.$$

The case $\mathbf{Q} = -\alpha\mathbf{I}$ with $\alpha > 0$ corresponds to gradient descent. The Chord Method is very useful. In some cases, for example, it is faster to compute the Hessian once for use throughout the entire algorithm, rather than repeatedly recomputing the Hessian so that $\mathbf{Q} = \mathbf{H}_0$ [?]. The Chord method is especially likely to be faster than the Newton-Raphson method if evaluating the gradient is cheap compared to the cost of computing the Hessian.
It is important that the matrix $-\mathbf{Q}$ be positive definite. The derivative of $f(x_n)$ in the direction $\mathbf{Q}\mathbf{g}_n$ is proportional to $\mathbf{g_n}^\top \mathbf{Q}\mathbf{g_n}$ since:

$$f(\mathbf{x}_n + \mathbf{Q}\mathbf{g}_n) \approx f(\mathbf{x}_n) + \mathbf{g}_n^\top \mathbf{Q}\mathbf{g}_n.$$

Ensuring that $\mathbf{x}^\top \mathbf{Q}\mathbf{x} < 0, \forall \mathbf{x} \neq \mathbf{0}$ means that $\mathbf{Q}\mathbf{g}_n$ will always be a descent direction so that f can be reduced by taking a step in its direction.

It can be shown that the Chord method converges linearly, [?]. An informal argument will be provided here which closely follows the presentation in [?]. Let $\mathbf{g}(\mathbf{x})$ denote the mapping $\mathbf{g}(\mathbf{x}) = \mathbf{x} + \mathbf{Q}\nabla f(\mathbf{x})$ so that $\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n)$. Suppose the sequence $\mathbf{x}_n$ converges. To determine how quickly the converge occurs, perform a Taylor expansion about the limit $\mathbf{x}^*$:

$$\begin{aligned}
\mathbf{x}_{n+1} - \mathbf{x}^* &= \mathbf{g}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
&= (\mathbf{I} + \mathbf{Q}\mathbf{H})(\mathbf{x}_n - \mathbf{x}^*) \\
&= \mathbf{K}(\mathbf{x}_n - \mathbf{x}^*).
\end{aligned}$$

Let $\mathbf{H}$ denote the Hessian of f at $\mathbf{x}^*$. The convergence of the Chord method around $\mathbf{x}^*$ is governed by the matrix $\mathbf{K} = \mathbf{I} + \mathbf{Q}\mathbf{H}$. If $\mathbf{K} = 0$, then $\mathbf{Q} = \mathbf{H}^{-1}$ and the method converges superlinearly. However it is very rarely the case that the Hessian at the limit point is available. Usually the matrix $\mathbf{Q}$ is only an approximation to $\mathbf{H}^{-1}$. The better the approximation, the smaller the matrix $\mathbf{K}$ will be, and the faster the rate of convergence.

A.3 More Sophisticated Hessian Approximations

Instead of using a fixed matrix on each iteration as with the Chord method, more advanced methods allow the matrix $\mathbf{Q}$ to vary on each iteration

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{Q}_n\mathbf{g}_n.$$

Using the derivation of the Newton-Raphson method presented in Section A.1, suppose that the objective function $f(\mathbf{x})$ is locally approximated by a quadratic function as follows:

$$f(\mathbf{x}) \approx f_n + \mathbf{g}_n^\top(\mathbf{x} - \mathbf{x}_n) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_n)^\top \mathbf{A}_n(\mathbf{x} - \mathbf{x}_n)$$

for some matrix $\mathbf{A}_n$. This approximation is minimised by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{A}_n^{-1}\mathbf{g}_n$$

A condition generally imposed is that the approximate Hessians $\mathbf{A}_n$ used must be positive-definite. This ensures that the approximate quadratics don't have any saddle points or surfaces on which the second derivative is zero so that a well defined search direction is guaranteed [**?** **?** **?**]. The choice $\mathbf{Q}_n = \mathbf{H}_n^{-1}$ corresponds to the Newton-Raphson method. The discussion in Section A.2 suggests that to ensure faster than linear convergence, it is necessary to ensure that $\mathbf{I} + \mathbf{Q}_n\mathbf{H}$ goes to $\mathbf{0}$ as n goes to infinity.[2]

A.4 Linearly Convergent Methods

Not every method that changes $\mathbf{Q}_n$ on each iteration converges superlinearly. A straightforward example of linear convergence is found by setting $\mathbf{A}_n$ to the diagonal elements of the Hessian, so that

$$\mathbf{A}_n = \begin{bmatrix} \frac{\partial f^2}{\partial x_1^2} & & \\ & \ddots & \\ & & \frac{\partial f^2}{\partial x_n^2}. \end{bmatrix}$$

Another method that only converges linearly is Fisher's method of scoring for maximum likelihood estimation. This uses the expected information matrix $\mathcal{I}(\theta)$ to approximate the observed information $\mathbf{I}(\theta)$. It is not the case that $\mathcal{I}(\hat{\theta}) = \mathbf{I}(\hat{\theta})$, so one should not expect $\mathbf{I} + \mathbf{Q}_n\mathbf{H} \to \mathbf{0}$ as the algorithm converges to the MLE $\hat{\theta}$. As a result, Fisher's method of scoring will only converge linearly, so long as it is a reasonable approximation to the observed Fisher Information.[3]

[2]As noted in [**?**], it is actually only required that $\mathbf{Q}_n$ approximates $\mathbf{H}_n$ along the directions which the algorithm is searching.

[3]As the sample size grows larger, the expected Fisher Information gets increasingly good at approximating $\mathbf{I}(\hat{\theta})^{-1}$, so that Fisher's Method of Scoring tends to converge faster and faster as the sample size gets bigger. But that does not mean that Fisher's Method of Scoring achieves superlinear convergence when applied to one specific set of data.

A.5 Quasi-Newton Methods

Quasi-Newton methods use the computed gradients to construct approximations to the true Hessians as the algorithm proceeds [?]. These methods produce a sequence of pseudo-Hessians $\mathbf{B}_n$ that satisfy the *secant condition:*

$$\mathbf{B}_n(\mathbf{x}_n - \mathbf{x}_{n-1}) = \mathbf{g}_{n+1} - \mathbf{g}_n.$$

In one dimension, finding a B_n that satisfies the secant condition is equivalent to computing a finite difference approximation to the second derivative:

$$\begin{aligned} B_n(x_n - x_{n-1}) &= f'(x_n) - f'(x_{n-1}) \\ B_n &= \frac{f'(x_n) - f'(x_{n-1})}{x_n - x_{n-1}}. \end{aligned}$$

When the sequence x_n converges, the denominator in the finite difference approximation also converges to zero, so the that the rate of convergence is faster than for a finite difference approximation which uses a fixed denominator h [?].

For multivariate problems, the second derivative is in the form of a matrix, so there is not enough information to construct a full approximation afresh on each iteration. Rather, the approximate Hessian is partially updated using one of several approaches. R's `optim` routine uses the BFGS method to compute the next approximate Hessian [?]. BFGS finds the symmetric matrix $\mathbf{B}_{n+1}$ satisfying the secant condition such that the inverse $\mathbf{B}_{n+1}^{-1}$ minimises a weighted Frobenius distance between itself and the previous inverse $\mathbf{B}_n^{-1}$. A low memory variant of BFGS known as L-BFGS is also available in R's standard libary [? ?].

Quasi-Newton Methods converge superlinearly, but are not quadratically convergent like Newton's Method [?]. They have the advantage however of avoiding the cost of computing Hessian matrices, so they can prove faster than Newton's method in practice despite more iterations being needed to achieve a given degree of accuracy.

A.6 Finite Differences

In several places in this thesis finite difference methods are used to approximate derivatives.[4] For example, one might wish to use finite difference methods to approximate the elements of $\mathbf{H}_n$. There are many different ways of computing finite difference approximations. For example, a straightforward scheme to approximated $\mathbf{H}_n$ that only requires two additional function evaluations is given by:

[4]Refer to [? ? ?] for more detail.

$$[\mathbf{H}_n]_{ij} \approx \frac{f(\mathbf{x}_n + h\mathbf{e}_i) + f(\mathbf{x}_n + h\mathbf{e}_j) - f(\mathbf{x}_n - h\mathbf{e}_i) - f(\mathbf{x}_n - h\mathbf{e}_j)}{4h^2}.$$

For very small values of h, any numerical error in evaluating the objective function will start to dominate the finite difference estimates. Suppose that that a univariate function $g(x)$ were computed with errors $\epsilon(x)$ and $|\epsilon(x)| \leq \bar{\epsilon}$ for all x. Furthermore, suppose a forward difference $\nabla_h g(x)$ approximation were used to estimate $g'(x)$. Taking into account the error in evaluating $g(x)$, the approximation proceeds as follows:

$$\begin{aligned}
\nabla_h g(x) &= \frac{g(x+h) + \epsilon(x+h) - f(x) - \epsilon(x)}{h} \\
&= \frac{g(x) + hg'(x) - g(x) + \epsilon(x+h) - \epsilon(x) + \mathcal{O}(h^2)}{h} \\
&= \frac{hg'(x) + \mathcal{O}(\bar{\epsilon}) + \mathcal{O}(h^2)}{h} \\
&= g'(x) + \mathcal{O}\left(\frac{\bar{\epsilon}}{h}\right) + \mathcal{O}(h) \\
&= g'(x) + \mathcal{O}\left(\frac{\bar{\epsilon}}{h} + h\right).
\end{aligned}$$

The approximation error will first shrink and then get larger and larger as $h \to 0$. The optimal choice of h is given by $h = \mathcal{O}(\sqrt{\bar{\epsilon}})$, because it simultaneously keeps the order of magnitudes of both $\bar{\epsilon}/h$ and h small.

The convergence rate for the finite difference method is more complex to define than for other methods. Define the errors by $\mathbf{e_n} = \mathbf{x_n} - \mathbf{x}^*$ where $\mathbf{x}^*$ is the optimal point. Subject to technical conditions[5] the errors satisfy the following recurrence relation [**?**]:[6]

$$\|\mathbf{e}_{n+1}\| \leq K[\bar{\epsilon} + (\|\mathbf{e}_n\| + h)\|\mathbf{e}_n\|]. \tag{A.1}$$

If the objective function is evaluated to a degree of accuracy close to machine precision, it is the case that $\bar{\epsilon}$ is on the order of 10^{-16} and $h \approx 10^{-8}$. On the other hand, if the objective function is approximated using a complex or slow simulation method, it might only be the case that $\bar{\epsilon} \approx 10^{-4}$ and $h \approx 10^{-2}$. To get a sense of the implications of this result, consider the following three cases:

- If $\|\mathbf{e}_n\|$ is very large compared to h and $\bar{\epsilon}$ so that $h \ll \|\mathbf{e}_n\|$ and $\bar{\epsilon} \ll \|\mathbf{e}_n\|$, then $\|\mathbf{e}_{n+1}\| \leq K\|\mathbf{e}_{n+1}\|^2$ holds approximately and the convergence appears quadratic.

[5]Including that $h > M\sqrt{\bar{\epsilon}}$, for some constant M that depends on the function being approximated.

[6]In the mulitvariate case here, we require that $\|\epsilon(\mathbf{x})\| \leq \bar{\epsilon}$ where $\epsilon(\mathbf{x})$ is the evaluation error as before.

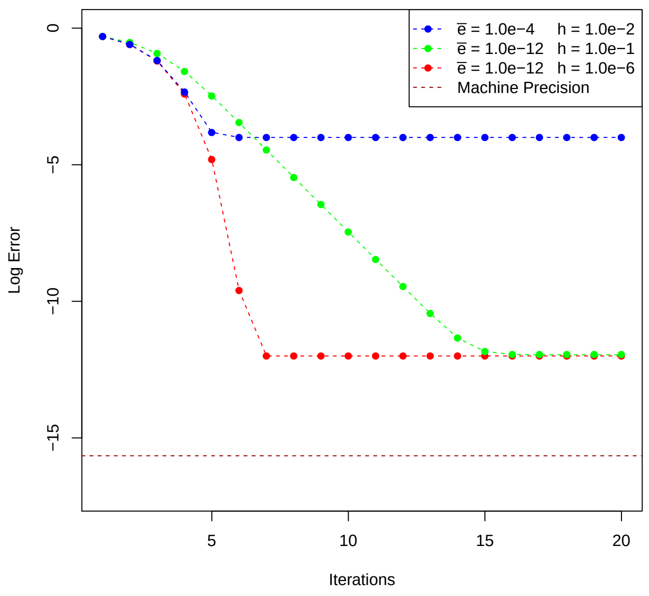


Figure A.3: Plot of the log iterates produced from the recurrence relation $e_{n+1} = \bar{\epsilon} + (e_n + h)e_n$ with $e_0 = 0.5$, for various choices of $\bar{\epsilon}$ and h.

- If $\|\mathbf{e}_n\|$ is of similar order of magnitude to h so that $\|\mathbf{e}_n\| \approx Ch$ for $1 < C < 10$; and both terms are large compared to $\bar{\epsilon}$, so that $h \gg \bar{\epsilon}$, and $\|\mathbf{e}_n\| \gg \bar{\epsilon}$; then it is approximately the case that $\|\mathbf{e}_{n+1}\| \leq K(C+1)h\|\mathbf{e}_n\|$, so that the rate of convergence appears to be linear.

- Finally, if $\|\mathbf{e}_n\|$ is of the same order of magnitude as $\bar{\epsilon}$ or smaller, we have that $\|\mathbf{e}_{n+1}\| \leq \mathcal{O}(K[\bar{\epsilon} + (\bar{\epsilon} + h)\bar{\epsilon}])$. A reduction in the error is no longer guaranteed.

In summary, the finite difference method converges quite rapidly at first, but begins to slow down and then stagnate entirely as it proceeds. The degree of accuracy that can be ultimately achieved is limited by $\bar{\epsilon}$. Figure A.3 illustrates the recurrence relation in Equation A.1. It is apparent that instead of converging towards zero, the various sequences converge to a limiting value depending the specific value of $\bar{\epsilon}$, and h governs the rate of convergence.

A.7 Line Search Methods

So far, every method has entailed computing a search direction $\mathbf{p}_n$ and letting $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha\mathbf{p}_n$, where it is often the case that $\alpha = 1$. The generated search directions $\mathbf{p}_n$ have been required to be a descent directions so that:

$$\frac{d}{d\alpha} f(\mathbf{x}_n + \alpha \mathbf{p}_n)\Big|_{\alpha=0} < 0.$$

This means that if α is small enough, a step in direction $\mathbf{p}_n$ will reduce the value of the objective function. At most, the optimisation method only knows the values of $f_n, \mathbf{g}_n$ and $\mathbf{H}_n$, so it can construct a quadratic or linear approximation to the objective function $\mathbf{x}_n$ and use that to find the next point. As illustrated in Figure A.4, the approximation can fail if one takes too big a step. For complex estimation problems, the objective function often has multiple peaks and troughs, so one must be careful that one has not wandered out of the range of validity of the locally constructed approximation. The naive solution is to set α to some very small value. The first problem with this approach is that α might be unnecessarily small so that convergence is needlessly slow. The second problem is that the quality of a choice of α is governed by the higher order derivatives of the objective function, especially the higher order derivatives at the optimal point. It will often be the case that such information is not available. Third, a globally valid choice of α might be far too low in some places by definition. More flexibility would allow α to be bigger where viable to gain a faster rate of convergence. The more sophisticated approach is to allow α to vary on each iteration:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n.$$

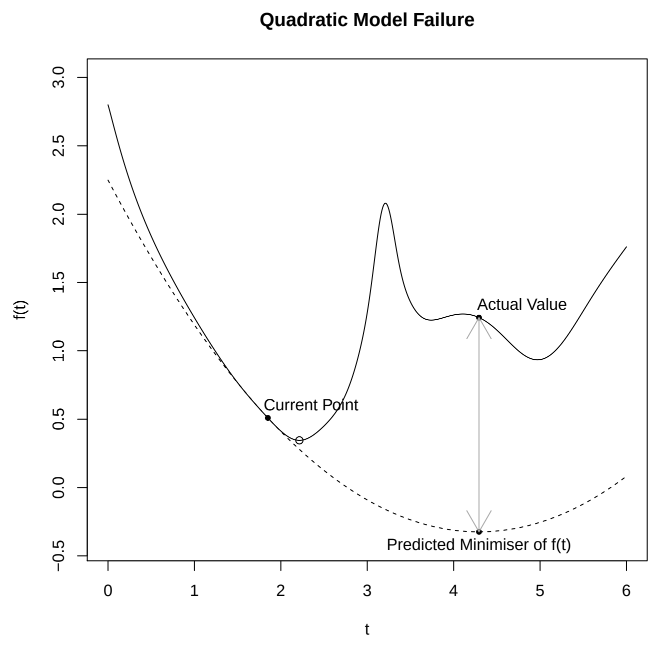


Figure A.4: Extrapolating too far out can lead to disaster!

Alongside the problem of choosing the search direction $\mathbf{p}_n$, it is now necessary to decide on each iteration how far in this direction to go. This entails probing the objective function along the ray $\{\mathbf{x}_n + \alpha\mathbf{p}_n | \alpha \geq 0\}$ to find the next point. For convenience, let the function $\phi(\alpha) = f(\mathbf{x}_n + \alpha\mathbf{p}_n)$ denote the restriction of $f(\cdot)$ to the ray from $\mathbf{x_n}$ in the direction $\mathbf{p}_n$. Note that $\phi(0) = f(\mathbf{x}_n)$ and $\phi'(\alpha) = \mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha\mathbf{p}_n)$. The goal of the line search is to find a point along the ray generated by $\mathbf{x}_n$ and $\mathbf{p}_n$ that is satisfactory. In principle one could attempt to find the value α^* that minimises $\phi(\alpha)$, but this is generally regarded as excessive. Usually one only iterates until sufficient improvement has been found and uses that point for the next iteration [**?**].

Besides simply setting $\alpha_n = \alpha$ for all n, the simplest line search method is known as a *Backtracking Line Search*. In this case, α is set to an initial value α_0 and repeatedly shrunk down to $\rho\alpha$, where $0 < \rho < 1$ until a satisfactory point is found.

Algorithm 1 Sketch of Backtracking Line Search

```
Require: That $0 < \rho < 1$; that $\alpha_0 > 0$; and that $\mathbf{p}$ is a descent direction at $\mathbf{x}$.
  function BacktrackingLineSearch($\mathbf{x}, \mathbf{p}$)
      $\alpha \leftarrow \alpha_0$
      while $f(\mathbf{x} + \alpha\mathbf{p})$ is unsatisfactory do
          $\alpha \leftarrow \rho\alpha$
      end while
      return $(\mathbf{x} + \alpha\mathbf{p})$
  end function
```

A.7.1 Wolfe Conditions

In order to ensure the line search method converges, the steps taken are required to satisfy the Wolfe conditions:

$$f(\mathbf{x}_n + \alpha_n\mathbf{p}_n) \leq f(\mathbf{x}_n) + c_1\alpha_n\mathbf{p}_n^\top \nabla f(\mathbf{x}_n) \qquad \textbf{(W1)}$$

$$|\mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha_n\mathbf{p}_n)| \leq c_2|\mathbf{p}_n^\top \nabla f(\mathbf{x}_n)|. \qquad \textbf{(W2)}$$

Restated in terms of $\phi(\alpha)$, the Wolfe conditions are:

$$\phi(\alpha_n) \leq \phi(0) + c_1\alpha_n\phi'(0) \qquad \textbf{(W1')}$$

$$|\phi'(\alpha_n)| \leq c_2|\phi'(0)| \qquad \textbf{(W2')}$$

The constants c_1 and c_2 are required to satisfy $0 < c_1 < c_2 < 1$ [**?**]. The first condition ensures sufficient decrease in the objective function, the second ensures a sufficient decrease in the gradient between steps. Intuitively, the bigger $|\phi'(0)|$ is, the bigger the decrease in the objective function demanded and the smaller the decrease in slope demanded, and vice versa.

A.7.2 More sophisticated Line Search Algorithms

More sophisticated line search algorithms try to make better use of the values of $\phi(\alpha)$ and $\phi'(\alpha)$ computed in the course of the search. The first approach is to construct a quadratic interpolant through the points $\phi(0), \phi'(0)$, and $\phi(\bar{\alpha})$, where $\bar{\alpha}$ is the current value of α. The constructed quadratic $m(\alpha)$ has the properties that $m(0) = \phi(0), m'(0) = \phi'(0)$ and $m(\bar{\alpha}) = \phi(\bar{\alpha})$ The next value of α is taken to be the minimiser of $m(\alpha)$. If the line search has been running for longer, so that values $\phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$ are available, then cubic interpolation is possible through the values $\phi(0), \phi'(0), \phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$. The next value of α is taken as the minimum of the interpolating cubic, which entails solving a quadratic equation. While cubic methods can converge more rapidly than quadratic methods[7] they are also sometimes unstable and can have multiple critical points.

More exotic choices proposed in the literature include the possibility of using rational functions to approximate $\phi(\alpha)$ instead of polynomials, as discussed in [**?**]. An example of such an approximation is:

$$\phi(\alpha) \approx \frac{p\alpha^2 + q\alpha + r}{s\alpha + u}. \tag{A.2}$$

Here, the values p, q, r, s and u are the parameters of the rational function to be determined.

[7]It is noted in the literature that line searaches using cubic interpolants constructed using only values of $\phi(\alpha)$ and no derivatives do not converge quadratically [**?**].

Appendix B

Implicit Filtering[1]

The weakness of the finite difference approach discussed in Section A.3 is that errors in evaluation eventually cripple the method. Suppose that the errors were related to the stepsize h so that the error bound is given by $\bar{\epsilon}(h)$ instead of a constant $\bar{\epsilon}$. In this case the error in the forward difference approximation would be given by:

$$\mathcal{O}\left(\frac{\bar{\epsilon}(h)}{h} + h\right).$$

So long as $\bar{\epsilon}(h)/h \to 0$ as $h \to 0$, the error in approximating the derivative goes to zero as well. The method might converge properly in this case. The Implicit Filtering methodology is designed for optimising problems where the exact value of the objective function $f(\cdot)$ is unavailable. Instead, it is assumed that there is a parameter h which controls the degree of accuracy. It is further assumed that the lower h, the lower the error. It is usually the case that getting a higher degree of accuracy implies a higher computational running time. For example, if the objective function is of the form $\mathbb{E}[f(\theta)]$ where the expectation is being approximated using a Monte Carlo method, it would be reasonable to set $h = 1/\sqrt{N}$ where N is the number of samples used so that the standard deviation is proportional to h. On the other hand if the expectation were being approximated by numerical integration, it would be reasonable to set h to the step size used.

B.1 Description of the Algorithm

Let $f(\mathbf{x}; h)$ denote the result of approximately evaluating $f(\cdot)$ at the point $\mathbf{x}$ with precision level h and let $\epsilon(\mathbf{x}; h)$ be the associated error. To generate a search direction, Implicit Filtering uses an approximation $\nabla_h f(\mathbf{x})$ to the gradient $\nabla f(\mathbf{x})$ that depends on h. The simplest such approximation employs forward differencing to approximate the gradient:

[1]Interested readers are pointed towards [**?** **?** **?**].

$$[\nabla_h f(\mathbf{x})]_i = \frac{f(\mathbf{x} + h\mathbf{e}_i; h) - f(\mathbf{x}; h)}{h} \tag{B.1}$$

Here $[\nabla_h f(\mathbf{x})]_i$ denotes the ith component of $\nabla_h f(\mathbf{x})$ and $\mathbf{e}_i$ is the ith basis vector. This approximate gradient is then used to define a search direction. The algorithm proceeds to conduct a line search along this direction until a point that achieves a sufficient reduction is found, which then becomes the latest iterate, and a new search direction is computed. In the event of any of the following occurring, it is deemed that more precision is needed:

- A point achieving sufficient reduction cannot be found after a pre-specified maximum number of iterations.
- A clear descent direction cannot be identified.
- The approximate gradient is of a similar order of magnitude to h, so that one can't be confident that true gradient isn't in fact zero.

If any of these conditions hold, the value of h is shrunk so that $h \leftarrow \delta h$ with $0 < \delta < 1$. The algorithm then proceeds again with this higher level of precision. The algorithm terminates when the change in the value of the objective function produced by reducing the value of h and running again is within a chosen tolerance. The sequence of approximate values returned by Implicit Filtering is monotone decreasing so that if $m < n$ then $f(\mathbf{x}_m; h_1) \geq f(\mathbf{x}_n; h_2)$, where h_1 is the precision used with $\mathbf{x}_m$ and h_2 is the precision used with $\mathbf{x}_n$. However since Implicit Filtering only ever approximately evaluates the objective function, it is not necessarily the case that $m < n$ implies that $f(\mathbf{x}_m) \leq f(\mathbf{x}_n)$.

An variation of the Gauss-Newton algorithm based on the Implicit Filtering has been developed, and was used to fit parameters to an harmonic oscillator of the form $y'' + cy' + ku = 0$, where the initial conditions are known and c and k must be estimated [**?**].

For Implicit Filtering to converge, the error needs to decrease sufficiently rapidly relative to the sequence of precision parameters h_n. The specific conditions depend on the specific variation of Implicit Filtering used [**? ? ? ?**].The convergence criteria are similar to:

$$\lim_{n \to \infty} \left[\frac{\epsilon(\mathbf{x}; h_n)}{h_n} + h_n \right] = 0. \tag{B.2}$$

In a statistical context, failing to treat the convergence criteria with respect risks a failure of convergence. Consider again the question of minimising $\mathbb{E}[f(\theta)]$. For the Monte Carlo method that sets $h_n = 1/\sqrt{n}$, the associated error is approximately equal to $\sigma/\sqrt{n}$ for some $\sigma > 0$, Substituting this into Equation (B.2) gives:

$$\begin{aligned}\lim_{n\to\infty}\left[\frac{\epsilon(\mathbf{x};h_n)}{h_n}+h_n\right] &= \lim_{n\to\infty}\left[\frac{\left(\frac{\sigma}{\sqrt{n}}\right)}{\left(\frac{1}{\sqrt{n}}\right)}+\frac{1}{\sqrt{n}}\right] \\ &= \sigma.\end{aligned}$$

In this case, there is no guarantee that the method will converge. An obvious rectification would be to set the sample size associated with h_n to $(1/h_n)^4$ instead of $(1/h_n)^2$ so that $\epsilon(x;h_n)/h_n = \sigma/\sqrt{n}$. This illustrates a potential weakness of Implicit Filtering, that the cost of approximating the objective function grows very rapidly as h decreases.

Brief experimentation found that there were many disadvantages associated with Implicit Filtering. These include:

- Implicit Filtering is complex to code, and is thus difficult to maintain and debug. The R code used to fit the ODE in Equation (B.3) by Implicit Filtering came out at a little over 300 lines long. The code in the Data2LD package that performs optimisation is over 600 lines long. Code that uses Brent's Method tends to be much shorter.
- Implicit Filtering proved to be very slow when applied to the test problem in Section B.2.
- The results of the fitting are sensitive to the value of the shrink factor δ chosen.
- It can be necessary to add a penalty term to the objective function to ensure convergence.

B.2 Using Implicit Filtering to Fit an ODE to the Melanoma Data

To test Implicit Filtering, the following quasi-linear fourth order ODE was fitted to the melanoma data:[2]

$$y^{(4)} = \mu^2[1-\sin(\pi y'')^2]y''' - \omega^2 y''. \tag{B.3}$$

The objective function used was a penalised sum of squared errors of the form:

$$PENSSE(f(t),\omega,\mu) = \rho\sum[y_i - f(t_i)]^2 + (1-\rho)\int |f''(t)|^2 dt.$$

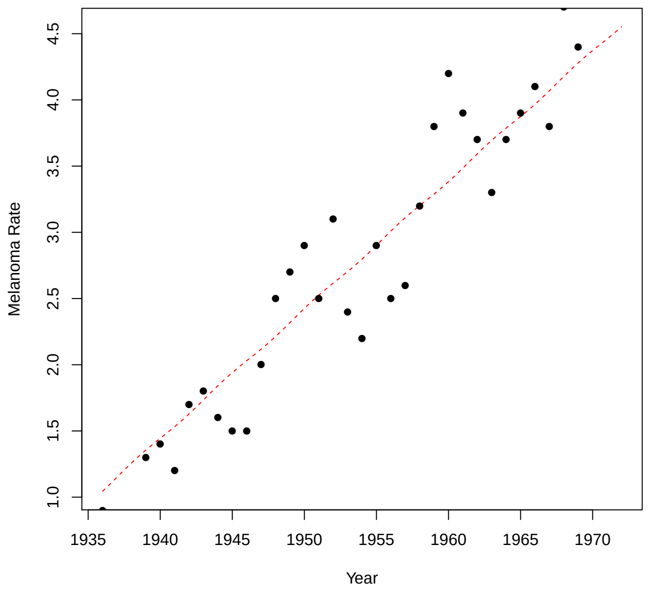
[2]The version of Implicit Filtering used is actually a modified version of that described above. A Quasi-Newton algorithm was used instead of naive gradient descent to compute search directions, and central differences were used to estimate the gradient instead of forward differences as in (B.1).

The value of $PENSSE$ is influenced by ω and μ because $f(t)$ is required to be a solution of Equation (B.3) with given values of ω and μ. The Implicit Filtering algorithm will not converge correctly without the penalty term as illustrated in Figure B.2. To compute $PENSSE$, the package `deSolve` was used to numerically solve Equation (B.3) with appropriate values of ω and μ. The precision factor h determined the step-size used. The $\int |f''(t)|^2 dt$ term was approximated by taking the vector of computed values of $f''(t)$ returned by `deSolve`, and then finding the sum of squared values. As $h \to 0$, this approximation becomes more and more accurate.

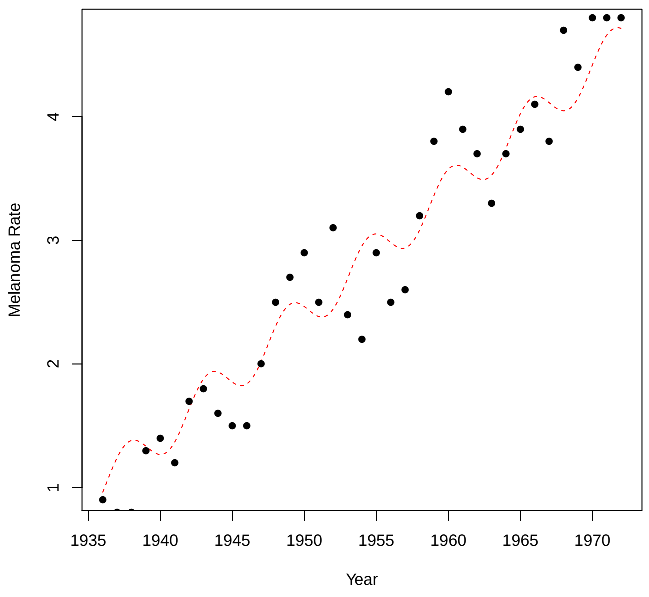
As can be seen in Table B.1, the algorithm takes a long time to run. It can be seen in both the table and Figure B.1 that changing the value of δ can introduce qualitative changes in behaviour. The algorithm is much quicker for $\delta = 0.9$, presumably because the algorithm is converging to a different fit than for the other cases. For the fastest case where $\delta = 0.9$, 200 values of the $PENSSE$ sequence are generated before the sequence converges to within a tolerance of 10^{-4}. This statistic substantially underestimates the actual amount of work done since Implicit Filtering rejects many evaluations as being inadequate in the course of its execution and further evaluations are needed to compute the approximate gradients $\nabla_h f(\cdot)$. For the case where $\delta = 0.9, PENSSE$ was computed over 3700 times with various values of h used.

δ	Running Time (Seconds)	Running Time (Minutes)
0.7	1717.807	28.63
0.8	1611.459	26.85
0.9	1013.165	16.88

Table B.1: Time taken for Implicit Filtering to fit (B.3) to the melanoma data for various values of δ.



(a)



(b)

Figure B.1: Fitting the ODE (B.3) to the Melanoma data. The exact value of the shrink value δ affects the fit the Implicit Filtering algorithm converges to. For $\delta = 0.7$ the computed fit in (a) resembles a straight line, but $\delta = 0.9$ results in a sinusoidal plus linear trend as can be seen in (b).

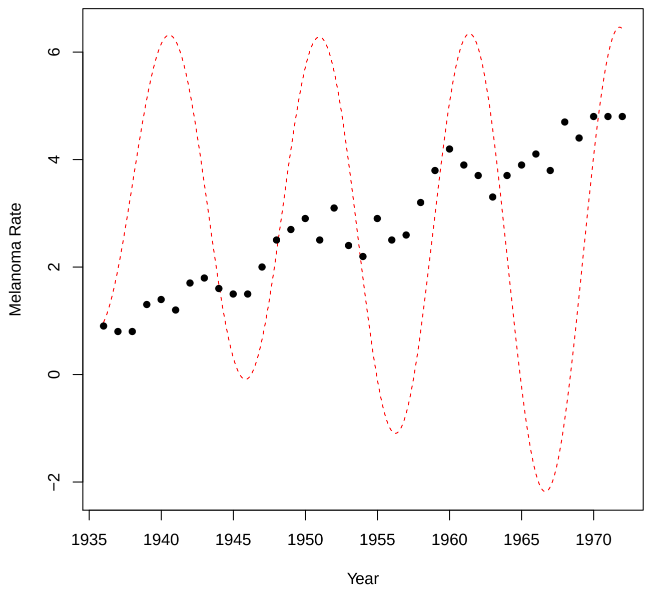


Figure B.2: Without a penalty term, Implicit Filtering entirely fails to fit the ODE (B.3) to the melanoma data.