# 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.[30]

The approach taken here is to assume that an unknown differential equation can adquately - though not necessarily exactly - describe the process producing the data.

#### Specification of Function Spaces

The functions in question are generally all 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 $\{a_1, a_2, \dots\}$ such that:

$$f(t) = a_1\phi_1(t) + a_2\phi_2(t) + \dots$$

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 generally 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 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 in question are continous mappings.

In practice one attempts to resolve this difficulty by finding or otherwise constructing a discrete problem that resembles the functional problem, and then solve this approximate problem.

It might be that case that the approximate problem can itself only be solved approximately using numerical methods.

Statistical problems that involve differential equations are particularly difficult. More naive approaches force the practitioner to solve the ordinary differential equation (ODE) numerically everytime it is desired to evaluate the goodness of fit. For these situations, it is necesary by definition to use numerical analytic techniques to construct a proxy problem that resembles the original problem sufficiently well and that is sufficiently easy to work with.

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 $\theta$ to be estimated, $\sigma$ is a volatility parameter, and $W(t)$ is a standard Brownian motion.

This SDE is equivalent to asserting for any time $t$ and increment $h$ that

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

Suppose there are observations $X_1, X_2, \ldots, X_N$ 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

$$X_{k+1} \approx X_k + f(X_k;\theta) + \sigma h Z_{k+1}.$$

The $Z_k$ here are i.i.d standard Normal random variables. This is known as the *Euler-Maruyama Approximation.*

Instead of attempting to estimate parameters for the SDE, we can fit 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.

In FDA, the assumption is usually made that all the functions can be represented as a linear combination from some choosen *finite* set of basis functions. Rather than discretise the differential operator as in the above example, the space of functions is discretised instead.

A differential equation (or a similar problem) over some finite dimesional space of functions with $n$ dimensions can be represented as a problem over the Euclidean space $R^n$, this is a discrete problem.

The modelling process for functional data as described in Figure 1.1 can be more complex than standard statistical problems.

**Formulate a Model:** As is the case for any statistical problem, the first step is to formulate a model . Here one must only be certain that the model at used is sufficiently broad or well-specified to be able to actually capture the phenomona at hand.

**Construct a Discretised Model that Approximates the Original Model:** Unless the statistical model is trivial, the next step is to construct a proxy model. This generally requires ideas from Numerical Analysis.

**Conduct Statistical Analysis Using the Discretised Model:** While the discretised model tends to be simpler than the original model, this task is not necesarily trivial as shall be seen.

**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 substantially biased as a result of the approximation error introduced, even if the original model were perfectly sound. If the orginal model is biased, then the approximate one might be even more so.

Therefore, one should consider conducting post hoc checks. For example, running the analysis again with a more accurate approximate model and comparing the results with the original model. If both agree, it is evidence the approximate models are both reasonably accurate. Constructing a more accurate approximation is generally a straightforward and intuitive process, with the exact approach depending on the situation at hand.

In the context of FDA, this generally entails increasing the number of basis functions so that the associated approximation error is smaller.

For example, suppose that one were attempting to estimate the parameters of an ODE by means of least squares, and one was using a finite difference solver to compute the fitted values, and hence to determine the goodness-of-fit.

Once the fitting alogrithm had converged, one might run the solver again with a smaller stepsize and the same parameters and check if this has made a substantial change in the the sum of squared residuals.

If there has been a substantial change as a result of the stepsize reduction, then one would have to consider running the entire fitting procedure again starting from the previously computed parameter estimate, except with the smaller stepsize, taking the new parameter estimates, and then checking again if decreasing the stepsize yet again produces substantial change in the goodness-of-fit statistic.

This procedure can even be automated. The Implicit Filtering alogrithm computes an approximate gradient using finite differnences and uses this to perform optimisation. If the alogorithm 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.

If the fitting method used is slow however, then these 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, Functional Data Analysis does not always require the recomputation 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 dimesional vector space. As shall be seen, the objective function is generally a mapping from some vector space $R^n$ to $R$ that can often be evaluated reasonably easily, or at least more easily than having to run an ODE solver.

**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 nonlinear regression models. These constructed nonlinear regression models should be checked for statistical validity.

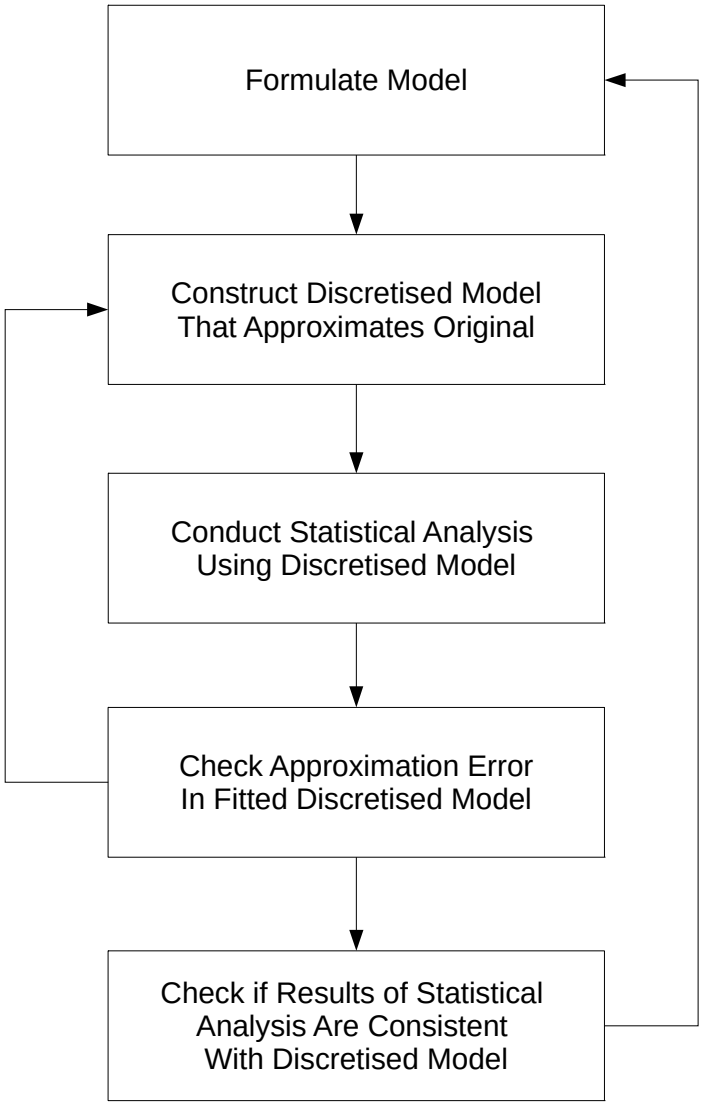



Figure 1.1: Statistcal Modelling Process For Functions

## 1.2 Penalised Regression

Suppose we have $N$ noisy observations $y_i$ at times $t_i$ from some function $f(t)$, and we wish to estimate $f(t)$, from the data. A naive approach would be to estimate $f(t)$ by minimising a least squares criterion:

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

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 $g(t_i) = y_i$ is will minimise $SSE(\cdot)$. There are an infinite number of degrees of freedom, but only a finite number of observations.

To ensure uniqueness, it is necessary to impose a further condition to discriminate between different candidates, a way to choose between different functions that interpolate a given set of points.

### 1.2.1 Smoothing Splines

One potential criterion is to introduce a second order penalty. 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 by using the integral of the second derivative over a chosen interval $[0, T]$. The upper limit $T$ should be choosen to allow for all observation times to be included.

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

For a given set of points, the smooth interpolating curve that minimises the energy 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 overfitting and poor predictive power. There is a trade-off between bias and variance.

In practice, a joint estimation strategy is pursued that attempts to find a good balance between fidelity to the observed data and reasonably regular behavior. This involves minimising the following penalised least squares criterion:

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

The $\lambda$ term dictates the trade-off between fidelity to the data and regularity.

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 $PENSSE$ 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 $\phi_i(t)$ that can represent any cubic spline with knots at the $t_i$. This implies that

$$f(t) = \sum_{i=1}^{K} c_i \phi_i(t).$$

Note that it is only required that the set of basis splines only possess enough resolution to represent the function that minimises $PENSSE$, it is not required that this set of splines is minimal.

Let the design matrix $\mathbf{\Phi}$ be defined by $\mathbf{\Phi}_{ij} = \phi_i(t_j)$, and let the weight matrix $\mathbf{R}$ be defined by $\mathbf{R}_{ij} = \int_0^T \phi_i''(t)\phi_j''(t)$. Then $PENSSE$ can be written in terms of the vector of coefficients $\mathbf{c}$ and observations $\mathbf{y}$ as:

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

The problem has been discretised into one on $R^K$.

The optimal value of $\mathbf{c}$ is given by

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

This is an exact solution to the original problem because the span of the $\{\phi_i(t)\}$ 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

Consider a more difficult penalised regression problem:

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

The penalty against $f''(t)$ has been replaced with a penalty against $f''(t) - f(t)$. $PENSSE$ can be minimised in this case taking by a piecewise function consisting of linear combinations of $\sin(t)$ and $\cos(t)$ over each interval, and matching them together.

Note that a function of the form

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

can be written as a polynomial in $e^{it}$ and $e^{-it}$. For this reason, such a piecewise trignometric function can also be referred to as a piecewise trignometric polynomial or a piecewise trignometric spline.[34]

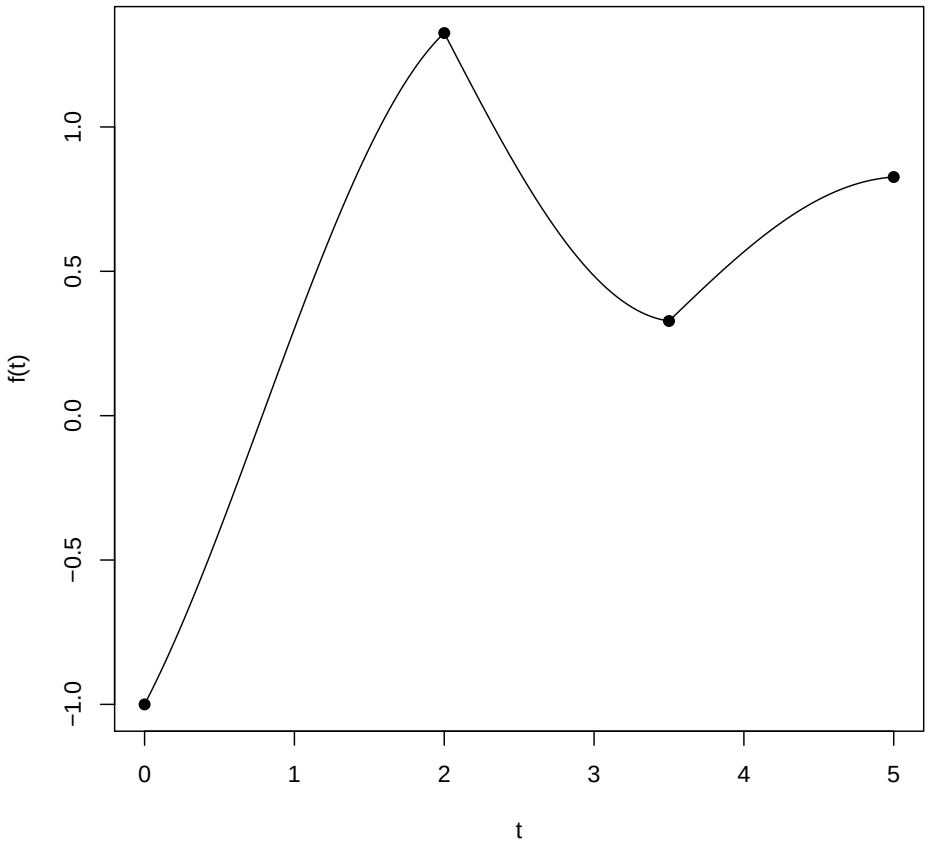



Figure 1.2: Plot of a Piecewise Trigonometic Curve. Note the kinks between segments.

As can be seen in Figure 1.2, a piecewise trignometric 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 desireable to impose the additional constraint that $f(t)$ must be everywhere differentiable. This cannot be achieved for a pieceswise basis formed from the functions $\{\sin(t), \cos(t)\}$ because there are only two free parameters on each segement and they are needed to ensure continuity.

## 1.3 Finte Dimensionalisation: the General Case

To find an exact solution to the two problems in Section 1.2, it was necessary to construct a finite dimensional function space that contained the minimal function. However it is not guaranteed that this is always possible. In practice, one would hope that the optimal function can be approximated sufficiently well by taking a linear combination from some choosen set of functions. Spline bases tend to be a reliable workhorse that are effectively the default choice. They provide a good balance between being well behaved as objects for regression and having good approximating power.

For comparison, Chebyshev Polynomials can often provide better approximation power for a given number of basis functions.[2] Unfortunately, it was found that they can be poorly behaved statistically because they consist of high order polynomials that are difficult to fit to data.

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

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

2. Assume that $f(t)$ can be written as a finite combination of chosen basis functions. In practice, this is only approximately true, so it is important to ensure that our basis can actually approximate the optimal $f(t)$ sufficently well. The function $f(t)$ can thus be written:

$$\begin{aligned} f(t) &= \sum_{i=1}^{K} c_i \phi_i(t) \\ &= [c_1, \ldots, c_K]'[\phi_1(t), \ldots, \phi_K(t)] \\ &= \mathbf{c}'\phi(t) \end{aligned}$$

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

3. Formulate the model in terms of the coefficient vector $\mathbf{c}$. A statistical problem over some given functional space has been transformed into a statistical problem over $R^K$.

For a given choice of $\mathbf{c}$, one gets a goodness-of-fit statistic of some kind back. It's important to note that the problem of fitting the coefficients $\mathbf{c}$ is a problem in nonlinear regression as a result of the finite dimesionalisation. Besides formulating an FDA model, one needs to consider the questions of constructing a finite dimensional approximation and then solving the associated nonlinear regression. The situation is sketched in Figure 1.4.

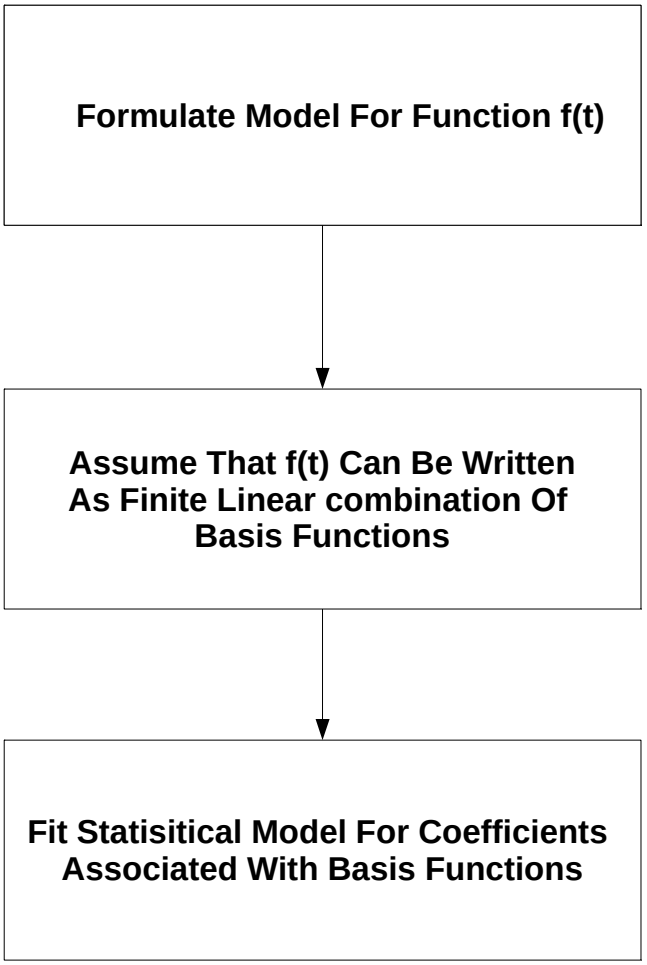



Figure 1.3: Statistical Modelling Process For Functional Data Analysis

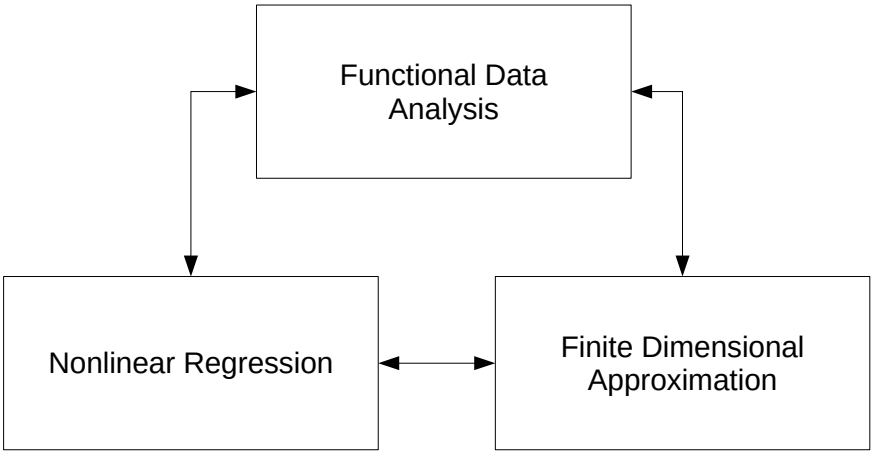



Figure 1.4: Elements of Functional Data Analysis

### 1.3.1 FDA With A Quadratic Basis

As is done in some texts[1], we will immediately provide an example with a very small basis to illustrate these steps. Consider the following penalised regression problem:

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

The differential equation associated with the penalty term is known as an Euler's Equation, and can be regarded here as a toy equation representative of equations such as Bessel'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 - a linear combination of the basis functions $\{1, t, t^2\}$:

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

Then:

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

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:

$$PENSSE(a,b,c;\lambda) = \sum_{i=1}^{N}(y_i - at_i^2 - bt_i - c)^2 + \lambda[a \ b \ c]'\mathbf{K}[a \ b \ c] \tag{1.1}$$

[1] [2] for example.

We have now gone from a problem specified in terms of functions, to a penalised least squares problem in the three coefficents $a, b$ and $c$. The quality of this approximate model as $\lambda$ 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 method was fitted to simulated data. A solution to the ODE $t^2 f'' - f = 0$ was generated over the interval $[0, 1]$, samples were taken at various points before being corrupted by Gaussian noise. The quadratic that minimised (1.1) 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

It's already been noted that the quality of the model depends partially on how well $f(t)$ can ever be approximated by a quadratic over $[0, 1]$ in the first place. Therefore, the quadratic $q(t)$ that minimises $\int_0^1 |f(t) - q(t)|dt$ was found numerically and also plotted in Figure 1.5.

Figure 1.5 suggests that $f(t)$ can be approximated reasonably well by quadratics for so long as one stays away from the point $t = 0$. This is consistent with theory. The ODE $t^2 f'' - f = 0$ behaves degenerately at the origin. When $t = 0$, the ODE has what is known as 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 execptionally tricky around this point.[13, 38]

Comparing the two fits in Figure 1.5, it is fair to argue that 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. The penalised fit is being heavily influenced by the singularity at $t = 0$ and probably would have performed better if a more robust loss function than least squares were used.

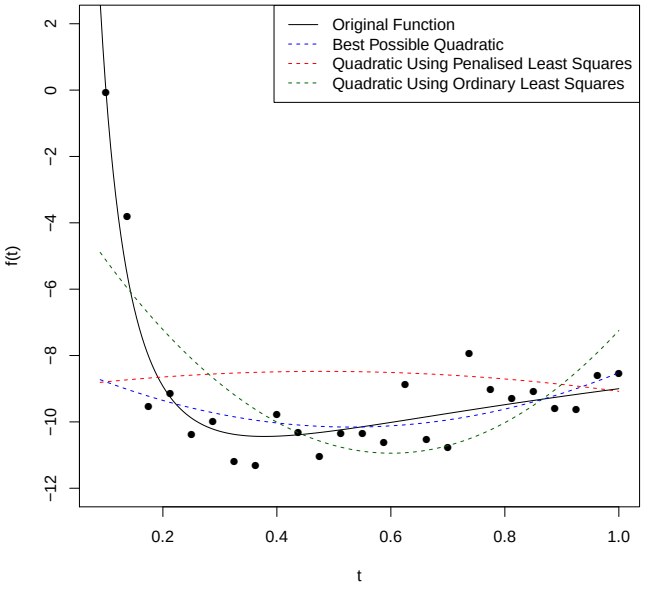



Figure 1.5: Performing FDA with the differential operator $Lf = t^2 f'' - 0.5f$ and the basis set $\{1, t, t^2\}$.

## 1.4 The FDA Package

Instead of having to develop develop FDA algorithms from scratch as done in Section 1.3, the `FDA` package was developed[29, 32] to tackle penalised problems of the form:

$$PENSSE(f) = \sum_{i=1}^{N} [y_i - f(t_i)]^2 + \lambda \int |Lf(t)|^2 dt \tag{1.2}$$

Here $Lf$ is a parameterised linear differential operator of the form $\sum_{j=0}^{n} \beta_j D^j$ where the $\beta_j$ are constants. The result of fitting the differential operator $Lf = f - \omega^2 f^{(4)}$ with $\omega = 0.65$ is shown in Figure 1.6.

The `FDA` packages is not as powerful as the `Data2LD` package, which will be introudced later on. It has the advantage of simplicity and ease of use though, 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 guidence on the best choice of the parameters $\beta_i$ nor the smoothing parameter $\lambda$.[2]

[2]The `FDA` package has a command called `lambda2gcv` whose documentation claims it '[finds] the smoothing parameter that minimizes GCV' [29]. Inspection of the code for this function shows that it only performs a fit based on the value of $\lambda$ passed and then reports the GCV. Incorrect or unclear documentation is unfortunately not an uncommon problem with FDA codes.

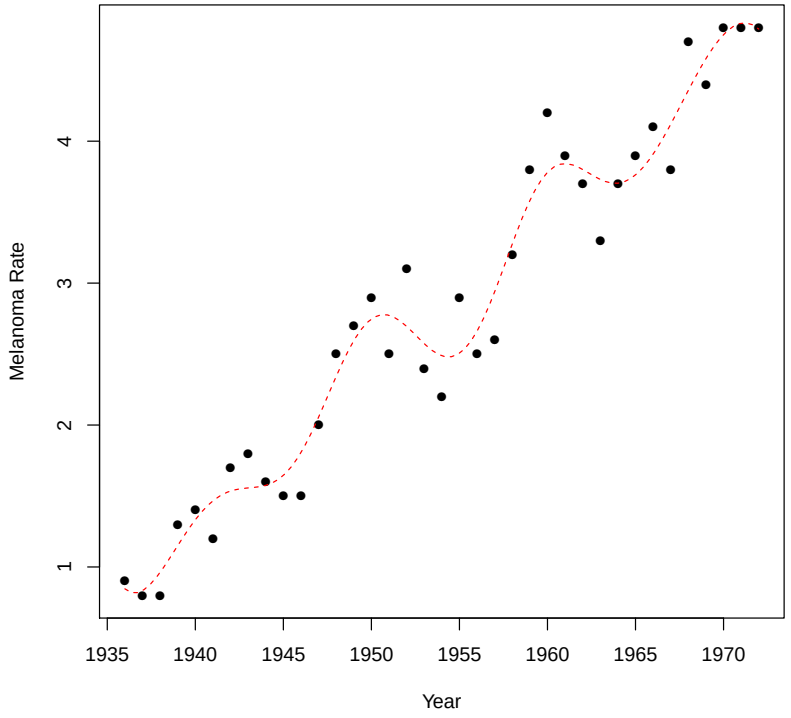



Figure 1.6: Using the `FDA` package to smooth the melanoma data with the differential operator $Lf = f - \omega^2 f^{(4)}$.

## 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:

$$\sum \beta_i(t) D^i f(t) = u(t) \tag{1.3}$$

The $\beta_i(t)$ are paremeter functions for the linear differential opertor on the lefthand side, and $u(t)$ is a forcing function.

More generally, Data2LD can model a system of inhomogenous linear differntial equations:

$$\mathbf{y}(t)' + \mathbf{B}(t)\mathbf{y} = \mathbf{u}(t) \tag{1.4}$$

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 $i$th equation.

A further advantage of Data2LD over the FDA package is that not only can it smooth ODEs with functional parameters, but it 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.[3]

## 1.6 Modelling the Reflux Data: A Parametric Approach vs Data2LD

The Reflux data, plotted in Figure 1.7, describes the output of an oil refining system. A given fraction of oil is being 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 [30]. The Reflux data was taken from the Data2LD package used for FDA, which will be discussed in more detail later. The authors of the Data2LD package model the data using the following ODE:

$$\begin{cases} y'(t) = -\beta y(t) & t \leq t_0 \\ y'(t) = -\beta y(t) + u_0 & t \geq t_0 \\ y(0) = 0 \end{cases} \tag{1.5}$$

Up until 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 arbitray constant, then the solution is given by

[3]For Data2LD, the smoothing parameter is written in terms of $\rho = \lambda/(1+\lambda)$.

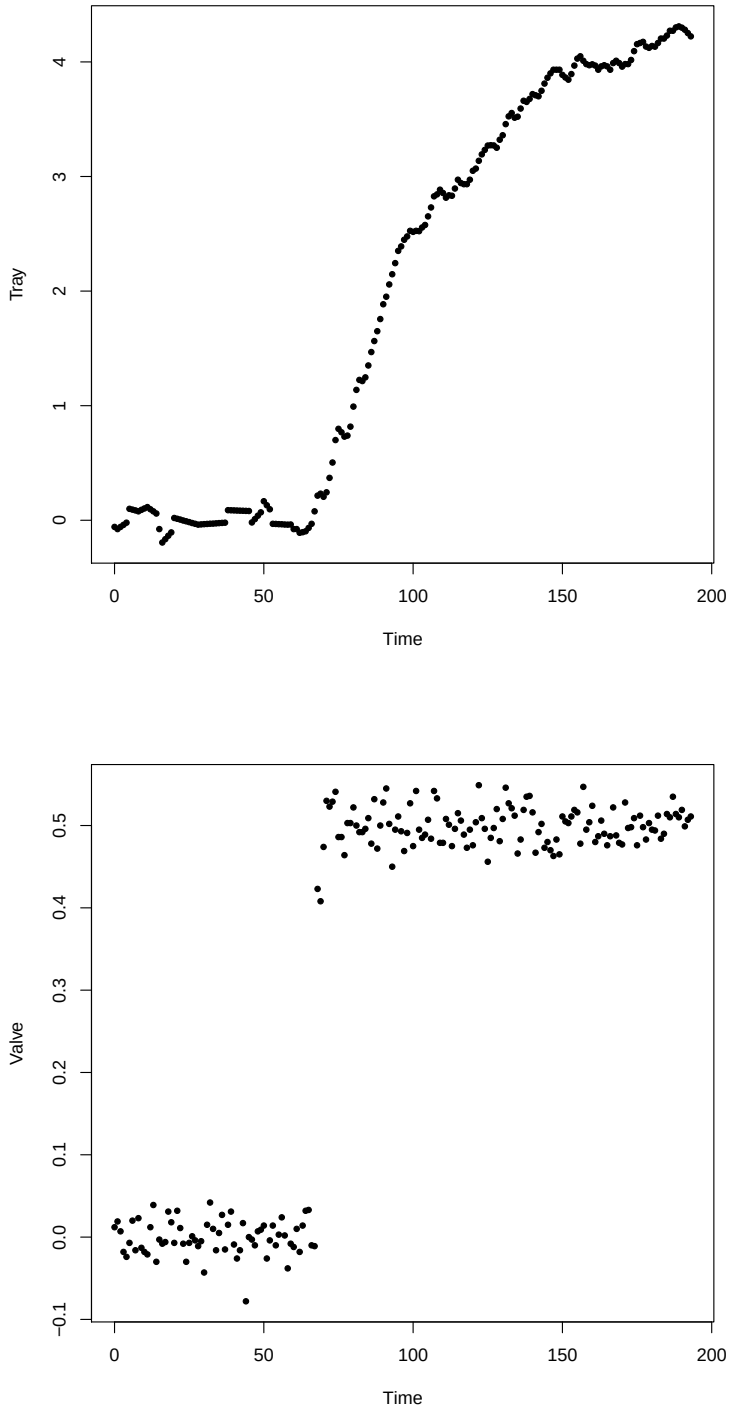



Figure 1.7: Reflux Data

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

Without loss of generality the exponential term $Ce^{-\beta(t-t_0)}$ can be replaced with one of the is of the form $Ce^{-\beta t}$. This is the case because $Ce^{-\beta(t-t_0)} = Ce^{-\beta t}e^{-\beta t_0} = [Ce^{-\beta t_0}e^{-\beta t}]$, the $e^{-\beta t_0}$ term is thus absorbed into the constant term.

In order to ensure that $y(t)$ is continous at $t_0$ and monotone increasing, we require that $\gamma + C = 0$ and that $\beta > 0$

### 1.6.1 Parametric Approach

It turns out that the constraint $C = -\gamma$ is unsuitable from the point of view of numerical parameter estimation. 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.

Assume that $y(t)$ is instead given by:

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

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 $t_0'$ defined by:

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

The function $\tilde{y}(t)$ is a combination of simpler functions, joined together using the maximum operator instead of the addition operator, see Figure 1.8.

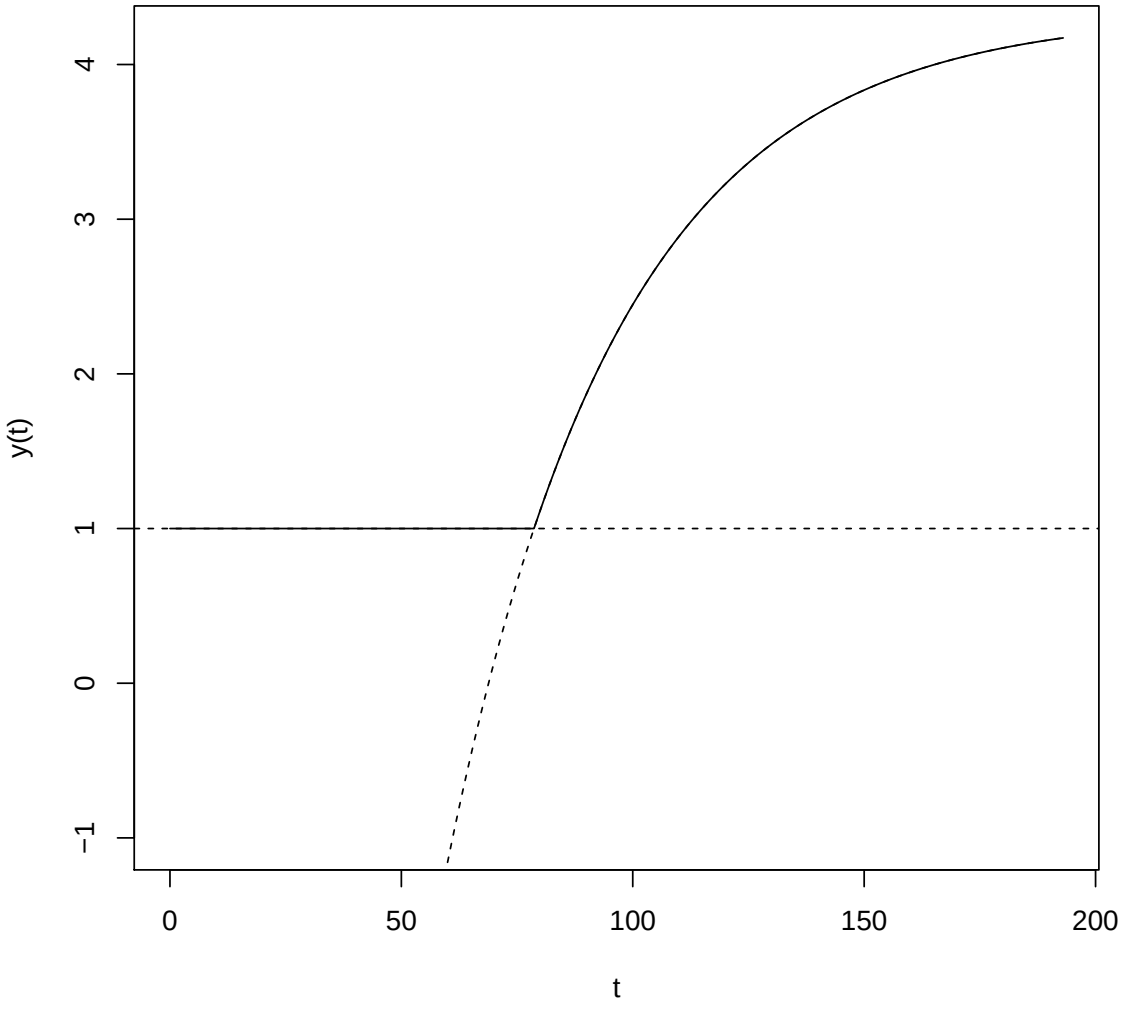



Figure 1.8: Plot of $\tilde{y}(t)$ and its constituent functions

**Parametric Fitting**

Instead of approximately solving an associated problem as discussed in Section 1.3, a purely parametric approach to fitting the ODE (1.5) will be employed. The question of modelling the Reflux data using FDA will be discussed in a later chapter.

We assume that the breakpoint $t_0$ is known in advance. Then our model for $y(t)$

$$y(t) = \begin{cases} 0 & t \leq t_0 \\ \beta_0 + \beta_1 e^{\beta_2 t} & t \geq t_0 \end{cases} \tag{1.6}$$

Note that this function might not be well defined at $t_0$, we will adress the question of matching later on. We must estimate the three unknown coefficients $\beta_0, \beta_1, \beta_2$.

**Estimating $\beta_0$ 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

So an initial estimate for $\beta_0$ is given by $\hat{\beta}_0 = \max(y_i)$

**Estimating $\beta_1$ and $\beta_2$ from $\beta_0$ and the data:** For $t \geq t_0$, the model in Equation 1.6 can be rearranged so that:

$$\log(\beta_0 - y(t)) = \log|\beta_1| + \beta_2 t$$

This equation is only valid so long as the left hand side is well defined however. It is necessary to exclude the largest observed value of $y$.

The values of $\log|\beta_1|$ and $\beta_2$ can be estimated by performing linear regression 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 $\log|\beta_1|$.

**Simulataneous Estimation of Parameters:** Now that we have reasonable estimates for $\beta_0, \beta_1$, and $\beta_2$, 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 produce 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 continous function that entirely satisfies the orginal 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 $\beta_0, \beta_1$, and $\beta_2$.

**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 tied to the ODE model to compute estimates. While this has produced an effective fit, there are obvious concerns about generalising this approach to other ODEs. Futhermore, since the fitting model was devised by peeking at the data, it is not obvious that one can find a valid p-value for the fit without getting an entirely new set of data.

This issue is difficult to resolve using purely 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 employed in Section 1.6.1 requires a considerable amount of domain-specific knowledge, the functional model 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.

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 $\beta$ 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.

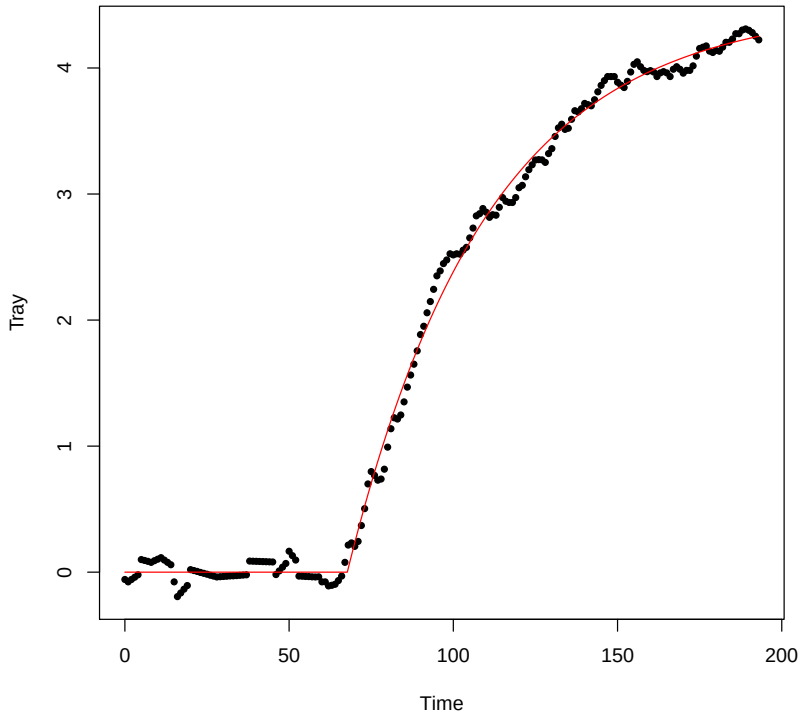



Figure 1.9: Fitting the Reflux data to the ODE model parametrically.

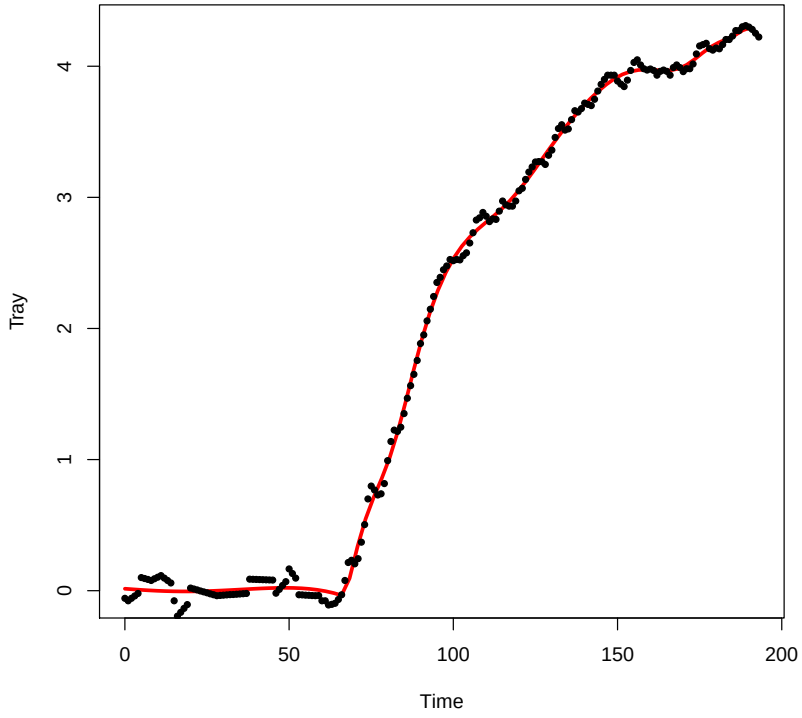



Figure 1.10: Modelling the Reflux data using `Data2LD`.

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 discontinous.

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[5]

Throughout this thesis, it will sometimes be desireable 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 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ that converges to a value $\mathbf{x}^*$.

[5]There are slightly different definitions of convegrgence rates from text to text, but all capture the same basic meaning. Refer to [7, 25].

**Linear Convergence:** A convergent sequence is said to *converge linearly*[6] to $\mathbf{x}^*$ (with convergence rate $\mu$) if there is a $0 < \mu < 1$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = \mu \tag{1.7}$$

If a sequence $\mathbf{x}_n$ converges linearly with constant $\mu$, 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 $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ If plotted on a log scale, the error terms $\|\mathbf{x}_{n+1} - \mathbf{x}^*\|$ will tend to lie on a straight line. As suggested by the nomenclature, 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: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$.

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

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

It is often the case that one can be more specific. A sequence is said to converge superlinearly with order $p$ if there exist positive constants $p > 1$ and $\mu > 0$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|^p} = \mu \tag{1.8}$$

If $p = 2$, the sequence is said to converge quadratically.

Note that there is no requirement that $\mu < 1$ in this case. If $p = 2$, the sequence is said to converge quadratically.Taking logs yields that $\log(\|\mathbf{x}_{n+1} - \mathbf{x}^*\|) \approx -p\log(\|\mathbf{x}_n - \mathbf{x}^*\|)$. 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

[6] In [25], 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.

have error on the order of $10^{-2}$, the next again will have error on the order of $10^{-4}$, and so on. An example of superlinear convergence is given by the sequence $x_n = 2^{-2^n}$.

**An Extended Definition of Convergence Rates:** The above appraoch to defining the rate 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 (1.8). To cover these situations, a sequence is also said to converge linearly/sublinearly/superlinearly if there is an associated auxillary sequence $\epsilon_n$ such that $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq \epsilon_n$ for all $n \geq 0$, and the sequence $\epsilon_n$ converges linearly/sublinearly/superlinearly to zero.[7]

**Linear Convergence and Iterated Mappings:** Nearly all estimation algorithms used in statistics start with an initial estimate $\theta_0$ and generate a sequence of estimates by $\theta_{n+1} = \mathbf{M}(\theta_n)$ for some mapping $\mathbf{M}(\cdot)$. The alogrithm is stopped when the generated sequence has converged within a tolerance of the limit $\theta^*$. 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 motiviated fitting alogorithm 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:

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

[7]The simple definition presented here is known as *Q-Convergence*, and the extended definition is known as *R-Convergence* [25].

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

[8]Consider for example the multivariate sequence defined by $(x_{n+1}, y_{n+1}) = (x_n^2, y_n/2)$. This 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.

# Chapter 2

# Hierarchial Estimation and the Parameter Cascade

First, a hierarchail parameter estimation method for PDEs is introduced. The Parameter Cascade is then introduced. Finally, a discussion on quasi-linear problems.

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 Hierarchial fitting of a Partial Differential Equation

A linear PDE that would be analogous to the linear ODE used to model the Reflux data would be 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 $\beta$.

The ODE $y'(t) + \beta y(t) = 0$ can be thought of as a simplifcation 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 - $\beta$ - 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 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 $\beta$ were unknown? The above discussion suggests a hierarchial approach to estimation: for a given choice of $\beta$, 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 $\beta$ with its sum of squared error:[1]

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

The problem of minimising $H(\beta)$ is a non-linear least squares problem that is also a two level hierachial estimation problem. The inner level consists of non-parametrically fitting a function to the set of points $\{(y_i, x_i - \beta_i)\}$ given $\beta$. 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.

## 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 $T_\theta$ is some differential operator, that is parameterised by an unknown $\theta$ that is to be estimated.

$T_\theta$ can be an ordinary differntial operator or a partial differential operator; linear, quasi-linear, or nonlinear.

There are two statistical objects to be estimated here: the parameter $\theta$, and the function $f(t)$.

[1] In case the left hand side might be slightly unclear - for the $i$th observation, the associated function $f(\cdot|\beta)$ is evaluated at $x_i - \beta t_i$.

Ramsay and Cao propose the following hierarchial approach to estimation[5]: Given a fixed value of $\theta$, let $f(t|\theta)$ denote the function that minimises $PENSSE(f, \theta)$ For a given value of $\theta$, it's associated mean square error is then defined by:

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

By making $f(t)$ dependent on $\theta$, the fitting problem has been reduced to a non-linear least squares problem.

This leaves the issue of estimating the optimal value of $\theta$ - Ramsay and Cao propose the use of gradient descent.

For a given value of $\theta$, $f(t|\theta)$ is found. These two values together are then used to compute $MSE(\theta)$ and $\nabla MSE(\theta)$. Finally, a new value of $\theta$ is computed by perturbing $\theta$ in the direction of the gradient.This scheme is sketched out in Figure 2.1.

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 $\theta$, and the *Middle Optimisation* entails finding the of value of $\theta$ that minimises $MSE(\theta)$.

There is thus a 'cascade' of estimation problems, where the results of the lower level estimation problems feed back in to the higher level ones.

Note that every time a new value of $\theta$ 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 $\theta$ is available.

Figure 2.1 is an idealised sketch of the Parameter Cascade as Ramsay and Cao would understand it. 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 be a complex process in its own right. This risks masking some of the comptuational work that is happening. A more realistic description is provided in Figure 2.2. In this thesis, Parameter Cascade problems that cannot be differentiated easily or at all are considered.

## 2.3 Three Stage Parameter Cascade

Up to this point, the structural parameter $\lambda$ has been treated as fixed. But it is possible to extend the Parameter Cascade to estimate $\lambda$.

It is necessary to introduce an *Outer Criterion* $F(\lambda)$ that determines how good a given choice of $\lambda$ is.

A common choice of outer criterion is Generalised Cross Validation[5, 31].

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

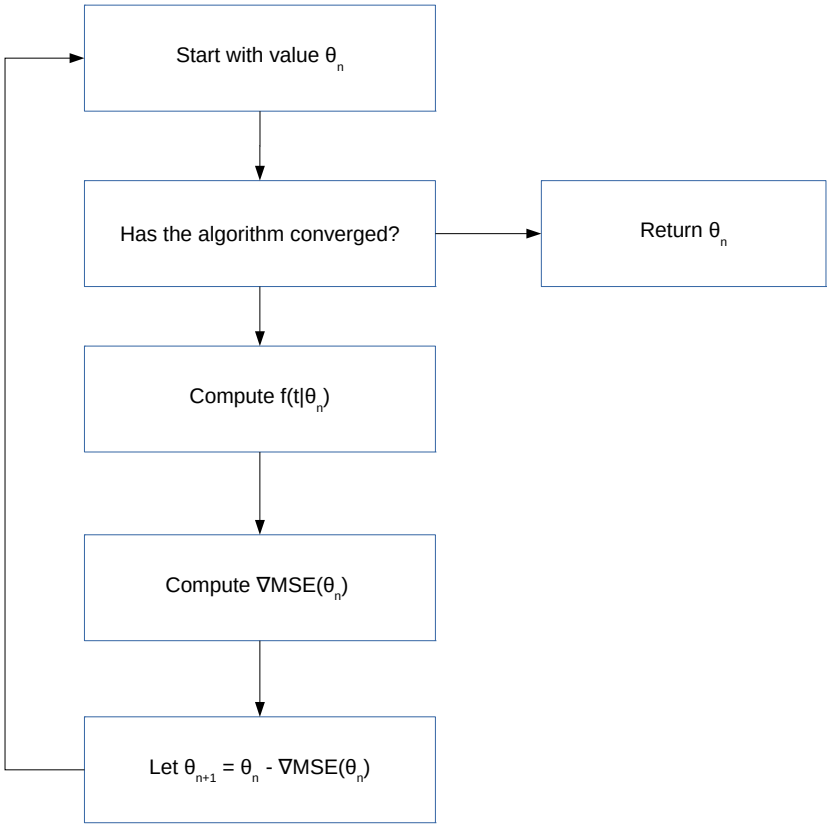



Figure 2.1: Two Stage Parameter Cascade (Simplified)

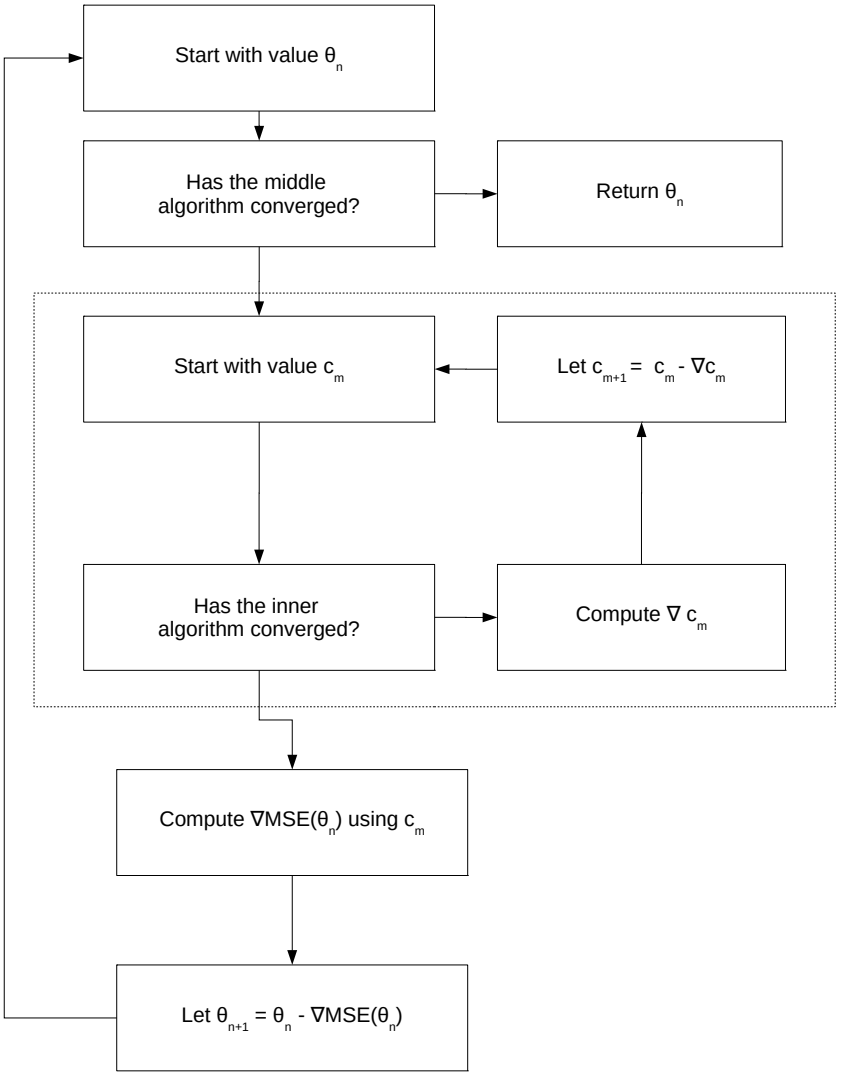



Figure 2.2: Schematic of the Two Stage Parameter Cascade With the Inner Optimisation Visible

to find the best parameter choice associated with a given value of $\lambda$, which in turn generates a series of subproblems to find the fitted function as the parameter space is explored.

As state in Chapter 1, neither the `FDA` nor `Data2LD` packages implement the three stage parameter cascade. They instead expect practioners to find the best choice of $\lambda$ by cycling through a set of predetermined values or even just employing manual adjustment.

## 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 is not implemented.

The `Data2LD` function is written in what might be called 'R Style' - upon calling the method, it returns a list with a of number computed quantites and statistics. 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 the 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. There is plenty of existing methods for tackling such optimisation problems.

### 2.4.1 How Data2LD Estimates Parameters

The code for Data2LD can be difficult to understand unfortunately. While software with such powerful features was probably inevitably going to be complex, the authors compound the issue by not heeding best practices recommended for making code easy to read and 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.[23]

The search directions used by the `Data2LD.opt` command are the gradient descent direction:

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

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

and the Newton Direction:

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

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

- First Wolfe Condition:

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

- Second Wolfe Condition:

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

- 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?

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

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 actually produce any improvement in the objective function. A failure in **T4** means the function has overshot a critical point.[4]

[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 pathalogical behaviour.

[4] If **T4** fails, this implies that $\mathbf{p}_n^\top \nabla f(\theta_n + \alpha_n \mathbf{p}_n)$ and $\mathbf{p}_n^\top \nabla f(\theta_n)$ are of opposite sign since $\mathbf{p}_n$ is choosen so that $\mathbf{p}_n^\top \mathbf{g}_n < 0$. The Intermediate Value Theorem means there is an $\bar{\alpha}$ between 0 and $\alpha_n$ such that $\mathbf{p}_n^\top \nabla f(\theta_n + \bar{\alpha} \mathbf{p}_n) = 0$, so that there is a critical point on the line segment between $\theta_n$ and $\theta_n + \alpha_n \mathbf{p}_n$.

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.4. `Data2LD.opt` is designed to be robust against the possibility that the objective function mightn't 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. Data2LD reduces the step size.
- If all four tests are failed, then the newest point is unsuitable entirely. Data2LD falls back on interpolation to try to find a critical point of $\phi(\alpha)$, falling back on quadratic interpolation methods if necessary.[5]

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.[6] One would think that the Newton-Raphson Method would be the first approach attempted to perform the line search before resorting to interpolation-based methods since it's both simpler to implement and faster to converge. The effort of computing $\phi''(\alpha)$ is mostly a sunk cost because of how the interface of `Data2LD` is defined.

The overall impression one gets is that the authors `Data2LD` might have been working under time pressure and added on extra tests as necessary to make the code work rather than starting from first principles. This would explain the failure to adhere strictly to coding guidelines and why the authors modfied the line search algorithm to consider the possibilty of the objective function becoming worse, but didn't contemporaneously modify the search alogrithm to make use of the $\phi''(\alpha)$ values instead of using interpolation.

This is perfectly understandable. `Data2LD` is a very sophisticated and complext piece of software.

[5]The line search code for the `Data2LD` is lightly commented and dense, all that one can be strictly certain of is that the method uses radicals to compute the next value of $\alpha$, falling back on solving a linear equation if necessary. Getting the root of a quadratic is equivalent to finding a critical point of a cubic, and solving a linear equation is equivalent to finding the critical points of a quadratic.

[6]Differentiating the expression $\phi'(\alpha) = \mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha \mathbf{p}_n)$ with respect to $\alpha$ 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 $\mathbf{x}_n + \alpha \mathbf{p}_n$.

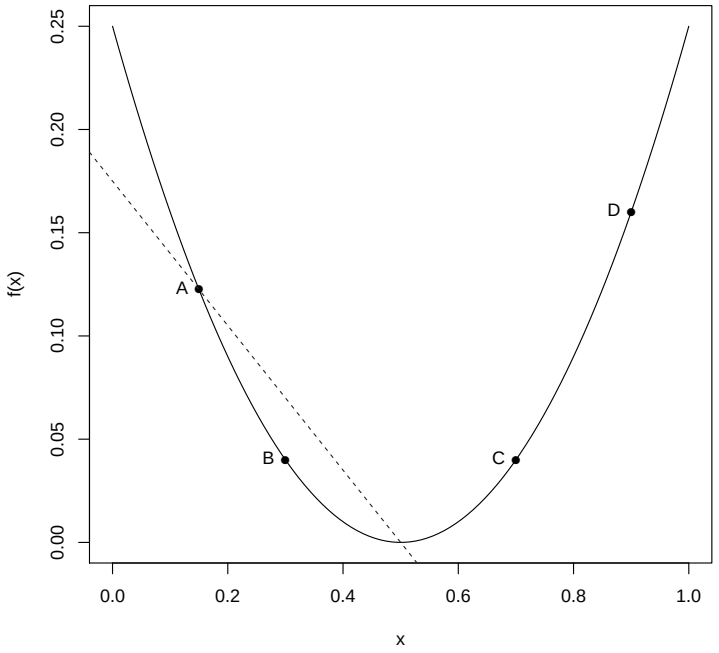



Figure 2.3: Point A is the initial point. Point B passes **T1** with $c_1 = 0.5$ and passes **T2** with $c_2 = 0.9$. Point C fails **T1** with $c_1 = 0.5$ and also fails **T3**, but passes **T4**, and passes **T2** with $c_2 = 0.9$. Point D fails all four tests.

### 2.4.2 Modifying `Data2LD`

**TODO: The text below is literally a copy and paste from an email sent in October 2018 titled "melanoma - tracking thetas"**

**Newton Method with Gradient Line Search**

```
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 Line Approximation To Slope** We experimented with using a secant approximation to estimate the slope instead of the true gradient. The output is presented below. The secant method's line searches tend to be slower than the one that uses the true derivative, and the algorithm doesn't reduce the gradient as much in each iteration. There is an advantage in that it's perhaps easier to see the line searches happening in the second case.

```
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.
```

# Chapter 3

# Derivative-Free Optimisation and the Paremeter Cascade

The Parameter Cascade requires the computation of derivatives for the various levels of the problem to perform optimisation. However, computing these can be a time-consuming and complex task. In some cases the derivative might not even exist. In this chapter, the use of derivative-free methods for optimisation. 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 objective function $f(\theta)$. Given an iterate $\theta_n$ and possibly some associated data, a quadratic approximation $m_n(\theta)$ to the objective function is constructed. The next iterate $\theta_{n+1}$ 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 minimised, producing a new iterate $\theta_{n+2}$. Ideally, the $\theta_n$ will approach the optimal point and the sequenece of quadratic models will become increasingly accurate approximations so that the process can be repeated until convergence. [25]

### 3.1.1 Newton's Method

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

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

It is not difficult to show that the critical point of $m_n(\theta)$ is given by $\theta_{n+1} = \theta_n - f'(\theta)/f''(\theta)$, which is the usual Newton formula [13, 19, 20, 25].

For a point close to $\theta_n$, 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[13]. This formula suggests that if $\theta_n$ is close to the optimal point $\theta^*$ 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 $\theta^*$. As a result, $\theta_{n+1}$ will be quite close to $\theta^*$. The next model $m_{n+1}(\theta)$ will thus be substantially better than $m_n(\theta)$ at approximating $f(\theta)$ around $\theta^*$, and so $\theta_{n+2}$ will be much closer to $\theta^*$ than $\theta_{n+1}$. Newton's method converges very rapidly so long as one is sufficiently close to $\theta^*$ to start with. 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 or the median 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 [13, 25][1]:

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

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 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 $\theta_{n-1}$ tends to be further from $\theta^*$ than $\theta_n$. 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 $\theta^*$, substituting in the $(\theta - \theta_{n-1})$ term inflates the error relative to Newton's Method and acts as a drag

[1] The Secant Method is denoted as the *Method of False Position* in [13]

on convergence. It can be shown that the Secant Method convergeces superlinearly with order 1.618, but 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[13]. 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 $(\theta_{n-2}, f(\theta_{n-2}))$[25, 39].

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

This model has a 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[25]. This is more acute for multivariate problems in particular. One approach is to provide enough points at the start and run the alogorithm 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 enough points to construct a parabola. If one is using a linear approximation, one must impose a limit 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 of approximating functions by polynomials. The only difference is 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. If one applies parabolic interpolation to a sufficiently smooth objective function, then one is in a sense automatically employing

Newton's Method even if one made no effort to investigate the differentiabilty of the objective function.

On the other hand, the methods all share the same fundamental handicap as well, 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 quadratric approximations are all something like $(\theta - \theta_n)^3$. 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. [13, 20, 25]

This is less academic than it might seem. Suppose one had a complicated likelihood function $L(\theta)$. Perhaps to evaluate the likelihood one must numerically integrate some kind of complex marginal distribution that depends on $\theta$. Instead of attempting to find explicit formulae for the score and information functions, if one could produce a crude estimate $\hat{\theta}$ and crude estimate of the error $\hat{\sigma}_\theta$, then one could use successive parabolic interpolation with $\{\hat{\theta}, \hat{\theta} - 2\hat{\sigma}_\theta, \hat{\theta} + 2\hat{\sigma}_\theta\}$ 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 $\theta$ that maximises $L(\theta)$ fairly quickly. It is necessary to provide plausible starting values for $\theta$ because the quadratic model is only certain to be valid if one is already near the optimal value.

## 3.2 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 that they are guaranteed to ensure consistent and steady progress towards the optimal point provided the function is continuous and does not have more than one local minima or maxima.

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 whole process is plotted in 3.1

By far, the most commonly bisection method 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 1.618.[20**?** ]

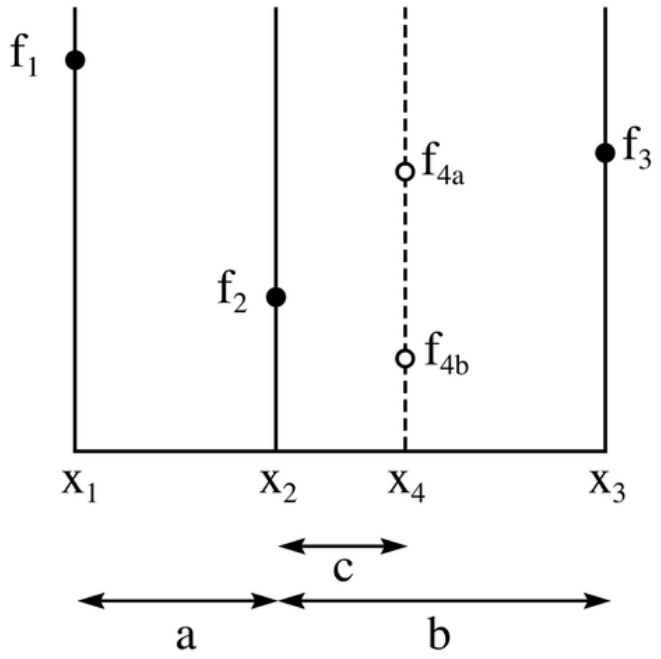



Figure 3.1: Diagram of golden-section search taken from Wikipedia. **TODO:** Find a better plot???

## 3.3 Brent's Method

Brent's Method is a hybrid of successive parabolic interpolation and golden-section search [4]. 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, while 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.

## 3.4 Estimation Of Parameters For A Standard Cauchy Distribution Using Brent's Method

To illustrate how Brent's metthod is employed in practice it will be used on a straight-forwards estimation problem first. Consider the quesion 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 by the usual method entails solving a fairly complex system of rational equations in $\mu$ and $\sigma$. 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 $\mu$ and $\sigma$ separately. The profile log-likelihood of $\sigma$ is computed:

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

R can evaluate $\ell(\sigma)$ straightfowardly by using Brent's method to optimise $L(\mu, \sigma)$ with respect to $\mu$ and holding $\sigma$ constant. The function $\ell(\sigma)$ can then be in turn optimised with respect to $\sigma$ to find the optimal value of $\sigma$. 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 $\mu$ and $\sigma$. To ensure that the algorithm was sufficiently close to the MLE, the median was used as an initial estimate of $\mu$, and half the interquartile range was used as an initial estimate for $\sigma$. 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 $\mu$ and scale parameter $\sigma$. 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 $\mu$ (rather than the actual true parameter), 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 derivitives, but 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 in principle to compute an approximate confidence interval for $\sigma$ and $\mu$[27]. Instead of analytic methods, one can use finite differences to approximately compute the necessary derivatives to the desired degree of accuracy[10, 22]. 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 $\sigma$ 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 $\sigma$, an approximate confidence interval for $\sigma$ can be computed. It is approximately the case that $\sigma$ 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 the case here as can be seen in the spike in the score statistic on the left caused by the Fisher information changing sign at approximately $\sigma = 2.35$. This indicates that the confidence interval might be wider than the range of for which a quadratic approximation around the MLE is valid, and should perhaps be treated with some scepticism.

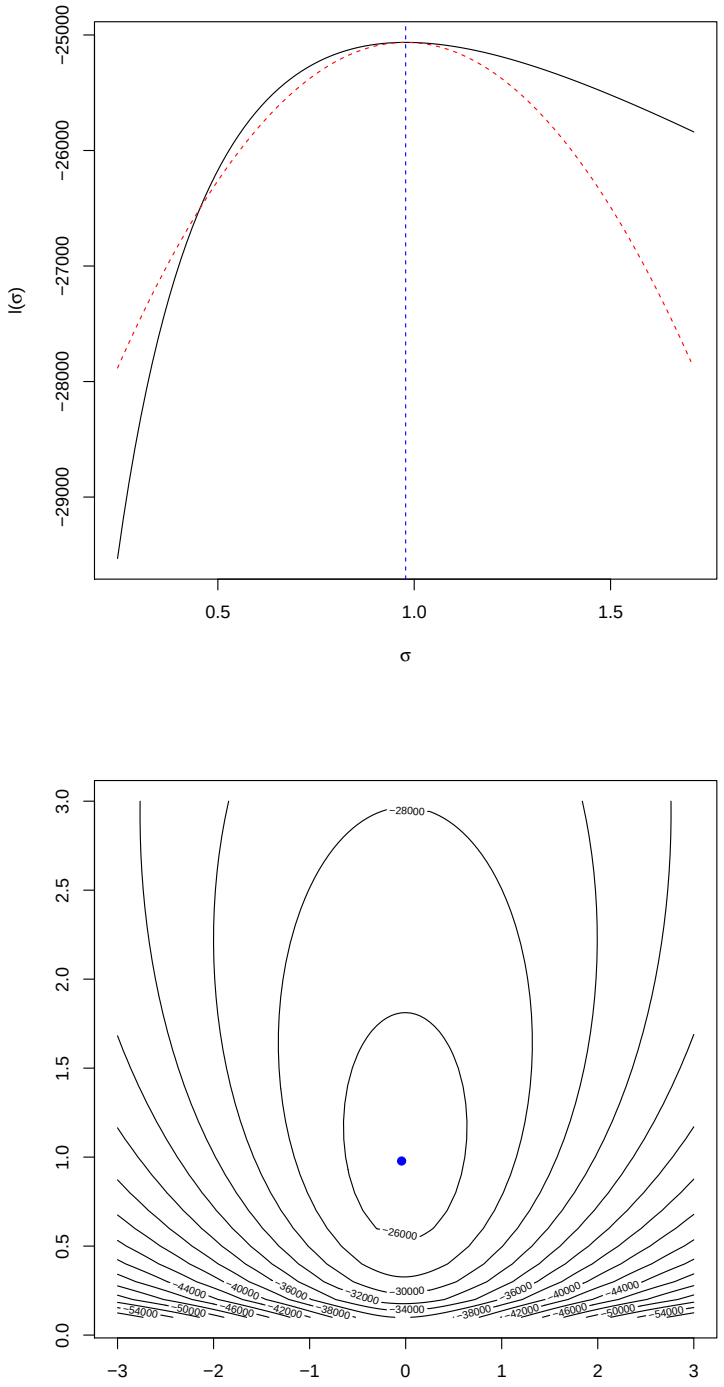



Figure 3.2: Profile log likelihood in $\sigma$, and contour plot of the joint log likelihood.

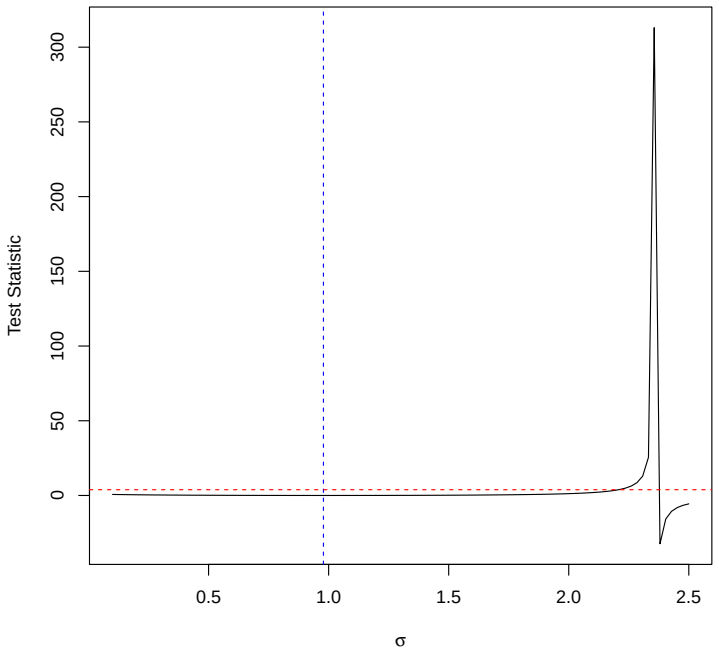



Figure 3.3: Plot of profile score statistic.

## 3.5 Robust ODE Parameter Estimation

If observations of values from an ODE are subject to heavy-tailed noise such as in the Cauchy case, least squares regression becomes unsuitable. An obvious candidate is L1 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 L1 regression is that median is naturally associated with this approach; the sample median of a set of numbers is the constant value that minimises the L1 error just as the sample mean is the constant value that minimises the least squares error[35][2]. L1 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 entirely unsuitable at worst, or not guaranteed to converge quickly at best.

Brent's Method can tackle such problems however, being robust against non differentiabilty. For nonlinear L1 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 it fairly quickly.

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

Consider for example, the following ODE with $\beta = -0.5$ :

$$\begin{cases} y'' - \beta(1-y^2)y' + y = 0 \\ y(0) = 1 \\ y'(0) = 0 \end{cases} \tag{3.2}$$

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, 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 $\beta$ from noisy observations.

The `desolve` package [36] was used to numerically find the values of $y(t)$ at choosen time points $\{t_1, \ldots, t_K\}$. The values of $y(t)$ at these points - corrupted by random Cauchy noise - were independently sampled $N$ times. This produced a set of $KN$ observations: $\{y_{11}, y_{12}, \ldots, y_{1N}, \ldots, y_{K1} \ldots, y_{KN}\}$. Because the data is heavy-tailed, least squares regression is inappropriate. Instead, the goodness of fit associated with a given choice of $\beta$ was measured by the sum of absolute errors associated with a given choice of $\beta$ :

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

Here $y(t; \beta)$ denotes the solution of Equation 3.2 for a given choice of $\beta$. To evaluate $SAE(\beta)$ at a given value of $\beta$, 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 $SAE(\beta)$. Figure 3.4 shows the original curve, the generated points, the realisation of $SAE(\beta)$, and the fitted curve generated by $\hat{\beta}$

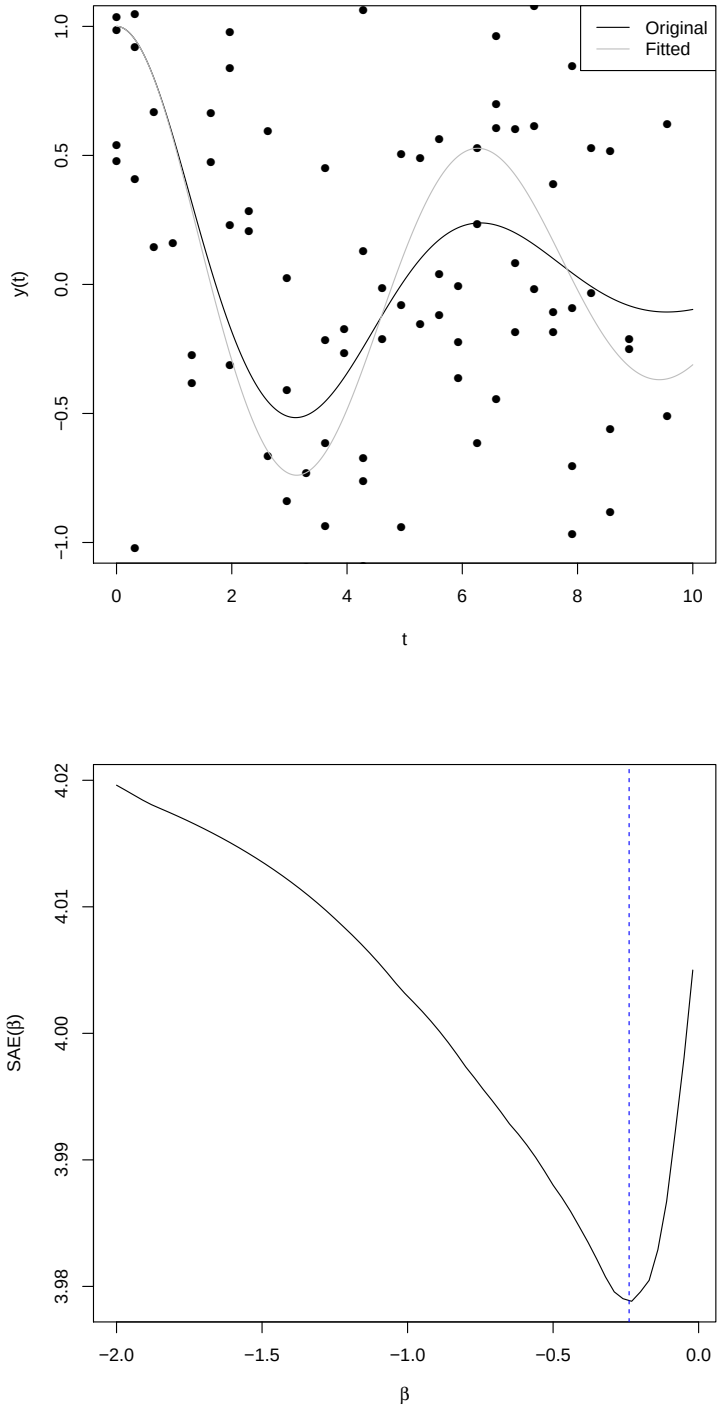



Figure 3.4: Original curve, fitted curve, and objective function.

## 3.6 The Parameter Cascade and Brent's Method

Recall that the Parameter Cascade has three levels.

First, the inner problem. There is a given functional $J(f;\theta,\lambda)$ that takes a function $f$ and associated parameters $\theta$ and $\lambda$ and returns a real number. Usually, the function $f$ is represented by a vector of coefficients with a given associated basis. The function $\hat{f}(t|\omega,\lambda)$ that optimises $J(\cdot;\theta,\lambda)$ is then found. Outside of toy cases, this problem cannot be solved analytically. The problem is nearly always solved numerically by restricting the space of functions to the span of some set of choosen 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 $\theta$ and $\lambda$ :

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

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

And finally, the outer problem attempts to determine the value of $\lambda$ that minimises the prediction error (generalisation error) by minimising another function $F(\lambda,\hat{\theta}(\lambda),\hat{f}(t|\omega,\lambda)) = F(\lambda)$. 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. This criterion is in turn optimised to find the optimal $\lambda$.

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 choosen 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 certain criterion for a given set of parameters. As previously discussed, the complexity of such problems can increase fairly rapidly and require a considerable degree of non-Statistical expert knowledge and often must be essentially 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 as the FDA package or Data2LD to avoid the effort of having to develop one's own. Ideally, it should be possible for one to plug in existing code that can compute $H(\cdot)$ and the optimal function as required.

There is thus a considerable degree of potential modularity present in the Parameter Cascade that is not fully investigated in Ramsay and Cao's paper [5], and research that inherits that framework. The Parameter Cascade can be adapted to heavy-tailed errors for example, by using appropriate loss functions for the various levels of the problem.

Not only is it good research practice to have mostly independent components that can be tackled and verified seperately before being combined, it is also good practice

from a software engineering perspective because the potential for complex interactions between different parts of code is reduced. This tends to save on debugging and testing requirements, which can be quite high when implenenting codes for FDA.

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 Brent's method however, there is more seperation, which makes it very easy to build optimisation routines on top of other 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 fitting the melanoma dataset with a parameterised linear differential operator:

$$L_\omega = D^2 - \omega^2 D^4. \tag{3.3}$$

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 routines that can do the numerical work of fitting the data with differential penalty given in (3.3) for given choices of $\lambda$ and $\omega$, and then report the associated mean square error.

Using Brent's method, the function $H(\omega; \mathbf{x}, \lambda)$ can be optimised with respect to $\omega$ for a given fixed $\lambda$. In turn, the outer objective function can be parameterised in terms of $\lambda$ and the associated optimal choice of $\omega$. This defines an objective function that can be again optimised to find the optimal choice of $\lambda$.

For $\omega$, Figure 3.5 shows tht the error is not particularly sensitive to small deviations from the optimal value even for fairly high values of $\lambda$. This suggests that the fitted curve will be adjusted to ensure no substantial increase in the error so long as $\omega$ isn't altered too much from the optimal value.

Heuristcally speaking, a flat objective function in the neighbourhood of the optimal point as can be seen in Figure 3.5 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.

If $\lambda$ is set too low, the optimal value of $\omega$ is numerically indistinguishable from zero. This is the case when $\omega$ is optimised for the value of $\lambda$ that minimises the GCV, Brent's method reports zero as the optimal value to within its default tolerance.

For $\lambda$, the curve has two critical points, with an asymptote as $\lambda$ tends to infinity.

A huge advantage of this approach compared to Data2LD's use of derivative-based methods is that it allows for the use of more robust loss functions since no use at all is made of derivatives.

Suppose one wanted to choose $\omega$ to minimise the Median Absolute Deviation - median($|y_i - \hat{f}(t_i|\omega, \lambda)|$) - instead of the least squares error. This loss function is choosen instead of the usual L1 error for the sake of demonstration because the L1 error might sometimes be tackled using a generalised version of gradient descent known as the subgradient method, while getting any kind of a derivative for MAD is difficult. It is quite simple, one just replaces the code that computes the least squares error with a few lines of R code that computes the MAD and run the optimisation routine again. It can be seen in Figures 3.6 and 3.7 that the MAD gives similar results to the usual least squares criterion, which suggests that both estimators are mutually consistent with each other.

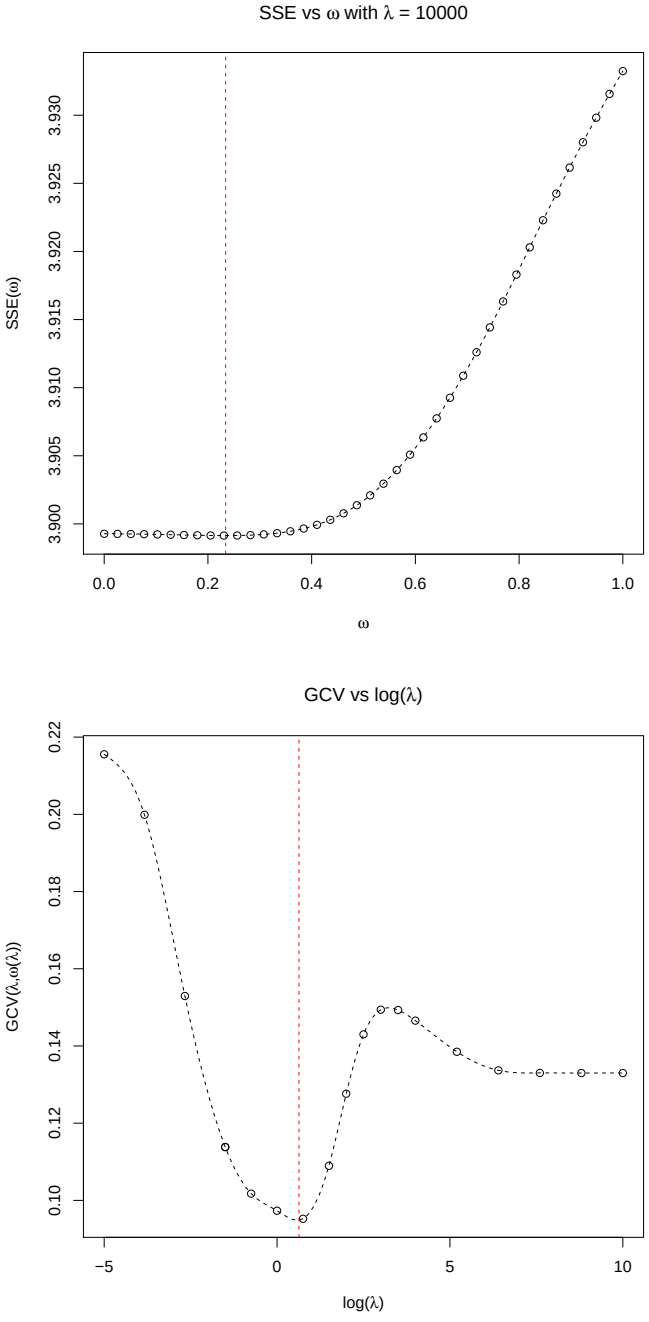



Figure 3.5: Plots of the middle and outer optimisation problems.

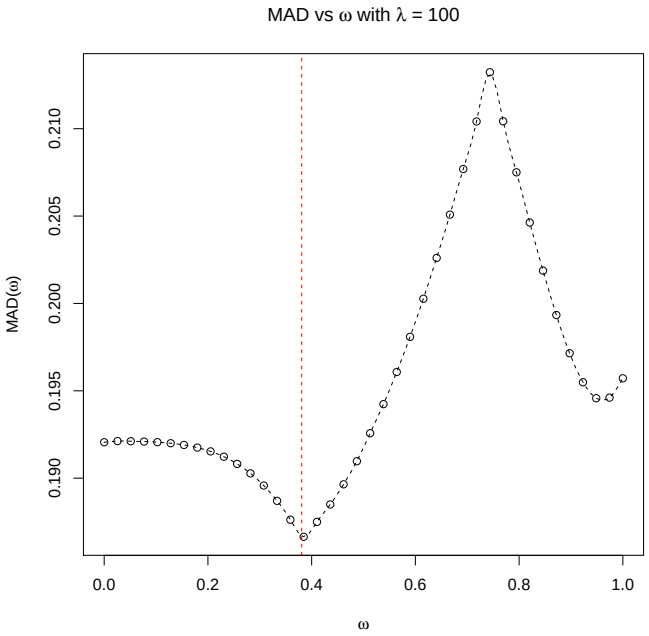



Figure 3.6: Plot of the middle optimisation problem with MAD used as a loss function

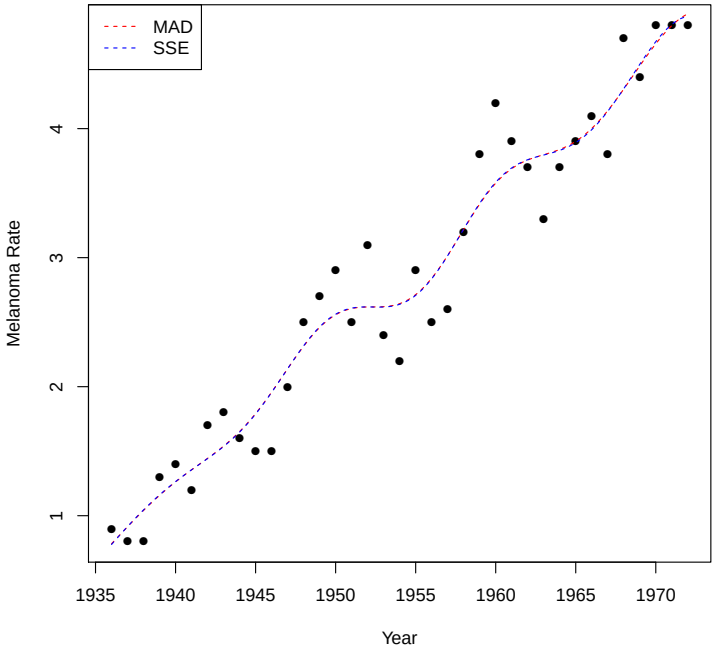



Figure 3.7: Comparison of fits for MAD and SSE criteria for middle problem

# Chapter 4

# A Two Level $L_1$ Parameter Cascade Using the MM Alogrithm.

In the previous chapter, Brent's method was introduced and was used to tackle Parameter Cascade problems where the middle level uses a loss function which is difficult or to differentiate or simply has no well-defined derivative everywhere. It was remarked that Brent's method ensures that the different elements of the Parameter Cascade tend to be loosely coupled from each other and this allows one to combine different fitting methodologies for different levels with straightforwardly.

Here these ideas are developed futher. 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 Paremeter 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 the previous it was extended to functions that take more than one real argument by optimising over each coordinate seperately when the Cauchy likelihood was optimised. However, there is no guarantee that this approach will perform well, and it can even 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 subproblem, which has the cumlative effect of increasing the running time of the alogrithm. Brent's method further requires the specification of a bounding box that contains the optimal point since it uses bisection, and that is harder and harder to do as the number of dimensions increases. All of these considerations mean that Brent's Method is highly unsuitable for peforming $L_1$ fitting over a space of functions which tend to have a large number of dimensions - by definition, there is one

[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 $f(11) = -10.99$.

dimension introduced for each basis function used. Likewise, the non-differentiability of the absloute value function means that other approaches that implicitly rely on differentiability such as parabolic interpolation are inadvisable.

Instead a different approach will be employed, a generalisation of the Iteratively Reweighted Least Squares algorithm for computing the $L_1$ median of a set of $N$ items $\{x_1, \ldots, x_N\}$ 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 MM alogrithm, and then proceed to straightforwardly extend this MM algorithm to produce a modified Penalised Sum of Squares problem that can be iteratively solved and reweighted 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 $\{x_1, \ldots, x_N\}$ , one wished to find the number $x$ that minimised the $L_1$ distance between them:

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

It is well known that $SAE(x)$ is minimised by the sample median of the numbers [35][2]. The usual approach to computing the sample median - sorting the numbers and taking the one in the middle - can't be generalised to FDA problems, so 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 work. Instead an approach known as Majorise-Minimise or the MM Alogrithm will be used[12, 19, 20]. For a given iterate $x_n$, a function $M(x|x_n)$ is required with the following properties:

$$M(x|x_n) \geq SAE(x)$$
$$M(x_n|x_n) = SAE(x_n)$$

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:

[2]This is asserted without proof in [35], probably because the proof tends to be simultaneously awkward but trivial to those familiar with it. The amount of work required to demonstrate that the sample median minimises $SAE(x)$ is greatly reduced if one notes that $SAE(x)$ is a convex function in $x$. Any local minimum of a convex function is also a global minimum[3], so one only needs to show that for any sufficiently small $\epsilon$ that $SAE(\bar{x}) \leq SAE(\bar{x} \pm \epsilon)$, where $\bar{x}$ is the sample median.

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

If such a function $M(x|y)$ could be determined such that $M(x|y)$ would be 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 Alogrithm. In fact, the EM algorithm is a special case of the MM algorithm[42] [3].

The most important question associated with the MM alogrithm is the construction of the majorising function because this tends to take up the bulk of the effort. Once the majoriser has been found, the algorithm is generally straightforward to implement, as will be seen shortly [12, 21]. Verifying a potential majoriser is usually straightforward, finding one in the first place is more difficult.The EM algorithm for example takes advantage of the probablistic structure of the problem and Jensen's inequality [4]. For an $L_1$ problem, the usual approach is to employ the Arithmetic Mean-Geometric Mean inequality [21]. Only the AM-GM inequality in its simplest form is required here, that the geometric mean of two numbers is less than or equal to their arithemtic mean:

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

It's worth noting that 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. One can represent the $L_1$ norm as a geometric mean, which then allows for the $L_1$ norm to majorised and seperated by a weighted sum of squares. Given an interate $x_n$, the AM-GM inequality implies that:

[3]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.

[4]The EM alogrithm is intended to maximise the log-likelihood and drives it upwards on each iteration, so it's an example of a Minorise-Maximise algorithm.

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

This in turn implies that:

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

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 either, 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 alogrithm thus 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 is defined by:

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

Here $T$ is used instead of $L$ to denote a differential operator that might not necessarily be linear.[5] As before, this can be majorised by a weighted sum of a residual-weighted penalised sum of squared errors, and a vestigal $\sum|x_i - f_n(t_i)|$ term that is only included for completeness and can be safely ignored in the course of the actual optimisation.

[5]In some situations $T$ could even be an integral operator. This could easily be the case for example if the observed values were the measured velocities of a vehicle, and the penalty was intended to impose constraints on quantities such as the distance travelled or fuel consumed

$$PENSAE(f) \leq \frac{1}{2} WPENSSE(f|f_n, \theta, 2\lambda) + \frac{1}{2}\left(\sum |x_i - f_n(t_i)|\right) \quad (4.1)$$

$$= \frac{1}{2}\left(\sum \frac{[x_i - f(t_i)]^2}{|x_i - f_n(t_i)|} + 2\lambda \int |Tf|^2 dt\right) + \frac{1}{2}\left(\sum |x_i - f_n(t_i)|\right) \quad (4.2)$$

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 montonically forced downwards.

### 4.1.3 Discussion

The sequence of penalised errors $PENSAE(f_n)$ is monontone decreasing but cannot be less than zero, so it is a bounded monotone sequence. The Monotone Convergence Theorem for sequences of real numbers[33] thus guarantees that a given generated sequence $PENSAE(f_n)$ will always converge to a limit. There are two cavaets. First, the sequence might converge to a different point depending on the starting values - 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. 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 [24], 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 the surrogate problems associated with EM alogrithms tend to be easy to solve using existing tools in the field. Likewise, 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 actual code for implementing penalised $L_1$ regression is brief.

The literature on the MM algorithm remarks that it is simple to implement and good at tackling high dimesional penalised regression, though convergence can be slow [42]. These claims are borne out when the convergence of the method is examined in Section 4.1.4 below.

#### Estimating the Worst Case Running Time

The arguments used to prove the Monotone Convergence Theorem can be extended a little to give a crude and weak estimate of how long the MM algorithm runs before the associated $PENSAE(f_n)$ sequence convergences to within a given tolerance. Suppose

there were a monotone decreasing sequence bounded between $a$ and $b$ so that $a \geq x_0 \geq x_1 \geq \cdots \geq x_n \geq \cdots \geq b$. By the Monotone Convergence Theorem, this sequence converges to some value $x^*$ such that $a \geq x^* \geq b$ and $x_n \geq x^*$ for all $n$.

Given $\epsilon > 0$, it is not hard to see that there exists an associated number $N(\epsilon)$ such that there exists an $n \leq N(\epsilon)$ with the property that $|x_{n+1} - x_n| \leq \epsilon$, where $N(\epsilon)$ is defined by $N(\epsilon) = \lceil (a-b)/\epsilon \rceil + 1$.[6] This result gives a crude and weak bound on how long it takes an before algorithm with a descent property achieves a tolerance threshold. If for example, the MM algorithm is programmed to terminate as soon as $|PENSAE(f_{n+1}) - PENSAE(f_n)| \leq 10^{-6}$, then one can expect the alogrithm to terminate within around $10^6$ iterations in the worst case.

Such woefully slow performance is unusual, but not entirely unprecedented. An example of a simple Poisson estimation problem where the EM Algorithm exhibits sublinear $\mathcal{O}(1/n)$ convergence[7] is given in [20]. The Implicit Filtering fitting algorithm previously discussed is another example of an incredibly slow method, requiring around 200 iterates before the sequence of values objective function converges to within $10^{-4}$ of each other [8].

This result is very pessimistic though, the EM and MM Alogrithms usuallly converge linearly [12, 24]. The MM alogrithm for $L_1$ fitting does not quite converge linearly though as shall be seen in Section 4.1.4 below.

[6] If $|x_{n+1} - x_n| > \epsilon, \forall n \leq N(\epsilon)$, it would be the case that $x_{N(\epsilon)} < b$, which contradicts the assumption that every element $x_n$ of the sequence is bounded below by $b$.

[7] A possibly vector-valued sequence $\{\mathbf{x}_n\}_{n=1}^{\infty}$ is said to exhibit $\mathcal{O}(1/n)$ convergence or converages at rate $\mathcal{O}(1/n)$ if there exists constants $c_1 > c_2 > 0$ and a limit value $\mathbf{x}^*$ such that $c_1/n \geq \|\mathbf{x}_n - \mathbf{x}^*\| \geq c_2/n$.

[8] The exact order of convergence of Implicit Filtering is difficult to determine. Experimentation suggests that the rate of convergence is sublinear.

### 4.1.4 Testing the Algorithm on the Melanoma Data

Since minimising $PENSAE$ is a problem over many dimesions, plotting the objective function to verify that the optimal function has been found isn't possible. Instead the MM alogrithm described in Section 4.1.2 will be tested by applying it to the melanoma data pertubred by random noise. Further, the covergence of the algorithm for the original melanoma dataset will be examined.

Figure 4.1 presents the $L_1$ and $L_2$ inner fits to the melanoma data corrupted by Cauchy distributed noise. The value of $\omega$ is held fixed at the reasonable value of 0.3, which was chosen as being roughly the average of the two different estimates of $\omega$ from computed in 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 least-squares fit tends to chase the heavy-tailed noise on the other hand. This is strong evidence that the curve that minimses $PENSAE$ has been found and that the method has been implemented correctly.

Figures 4.2 and 4.4 plot the convergence of $SAE$ and $PENSAE$ over the course of the alogrithm. Note that the PENSAE statistic doesn't quite actually converge monotonically as the theoretical analysis predicted. Instead, it fluctuates before settling down to the typical and expected pattern of languid monotone decline. Upon investigation, it was determined that over the first handful of 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 is equal to 1.48, and the highest was equal to $4.8 \times 10^6$. It seems that this rapid and large change produces qualititative changes in behaviour before the algorithm manges to 'burn in'. It is likely that observations with low weights are being effectively 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 4.3 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 $n \to \infty$.

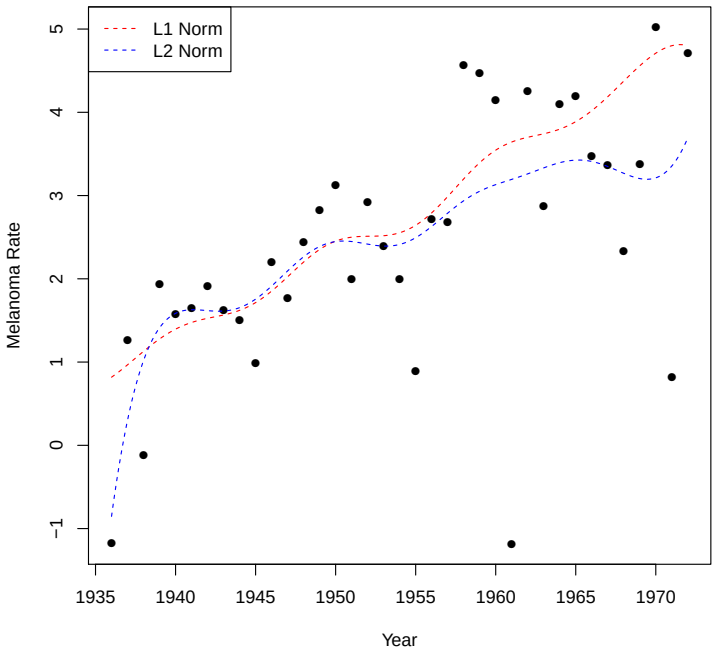



Figure 4.1: Comparison of $L_1$ and $L_2$ inner fits to Cauchy perturbed data with $\omega$ fixed at 0.3

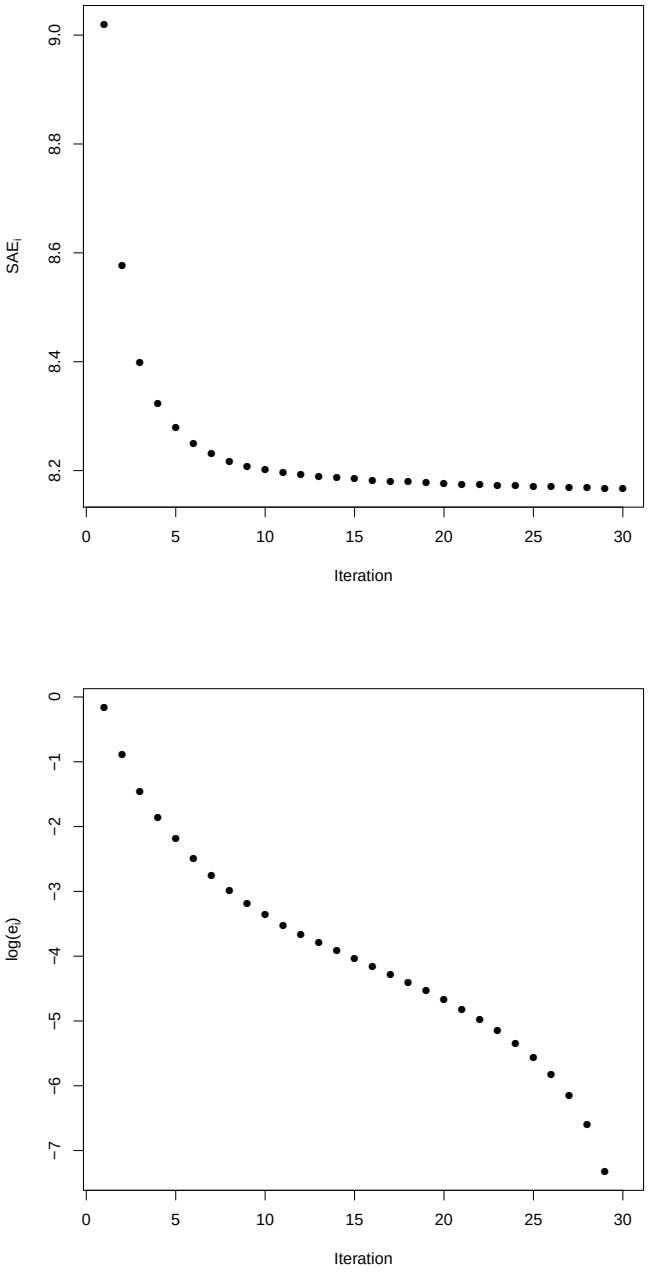



Figure 4.2: Plot of values and log differences for SAE Statistic

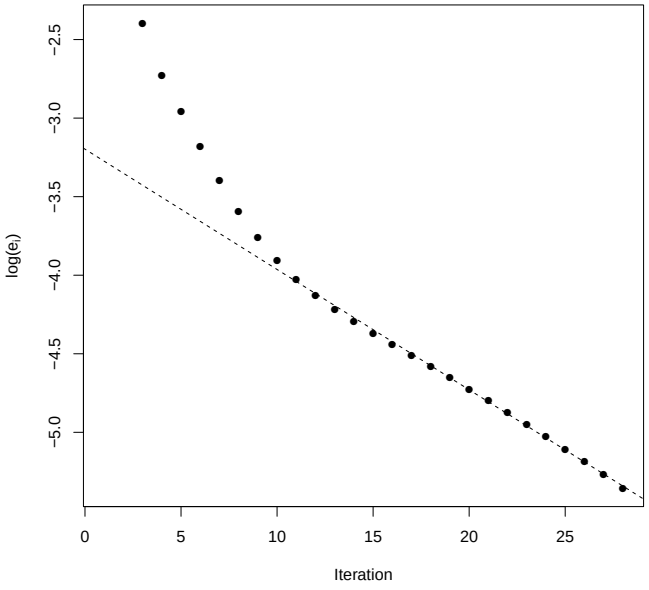



Figure 4.3: Plot of log norm differences for coefficients. Note that they tend to settle on a line.

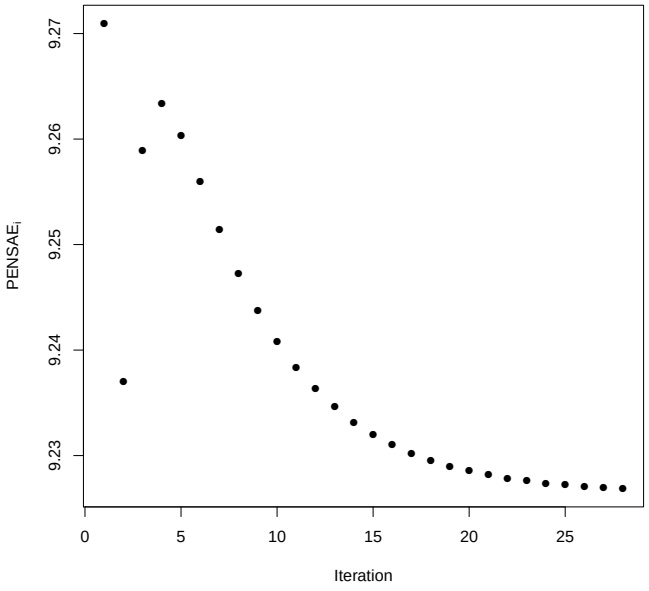



Figure 4.4: Plot of PENSAE statistic as the algorithm proceeds.

## 4.2 The Two Level Parameter Cascade with $L_1$ Norm

The inner problem of the parameter cascade is a semiparametric least squares regression model. The fitted function is modeled as a weighted sum of a solution to the differential equation (parametric), and a non-parametric residual. The lambda 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 probably 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 omega and lambda 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 lambda and omega parameters and the parameter cascade must use robust estimation at every level.

In the previous chapter, it was 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 the previous section, 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.

In Figure 4.5, the result of fitting a two level $L_1$ Parameter Cascade with L1 errors is plotted. It can be seen that the $SAE(\omega)$ function is irregularly shaped. In Figure 4.6 both the $L_1$ and $L_2$ fits to Cauchy-perturbed melanoma data are shown. Figure 4.7 plots the results of applying the $L_1$ and $L_2$ Parameter Cascades to the original and perturbed Melaonoma 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.

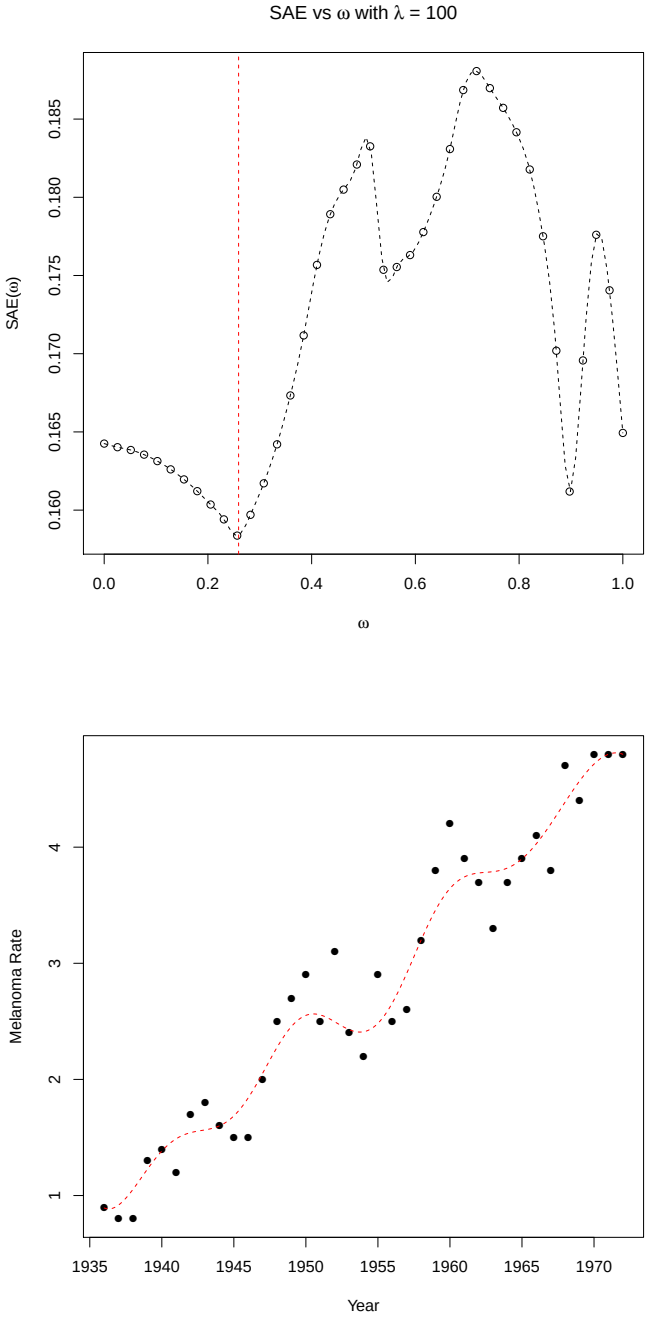



Figure 4.5: Fitting an $L_1$ Parameter Cascade to the Melanoma Data

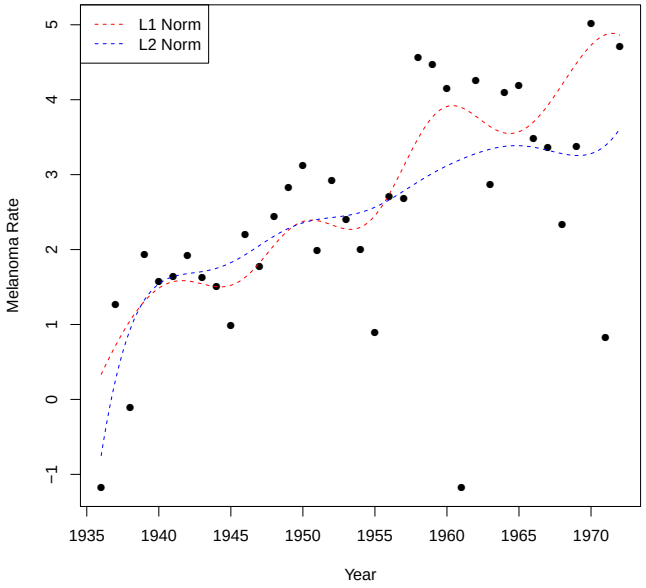



Figure 4.6: $L_1$ and $L_2$ Parameter Cascades with the same perturbed data as in Figure 4.1. Compare the $L_1$ curve in this plot with the one in Figure 4.5.

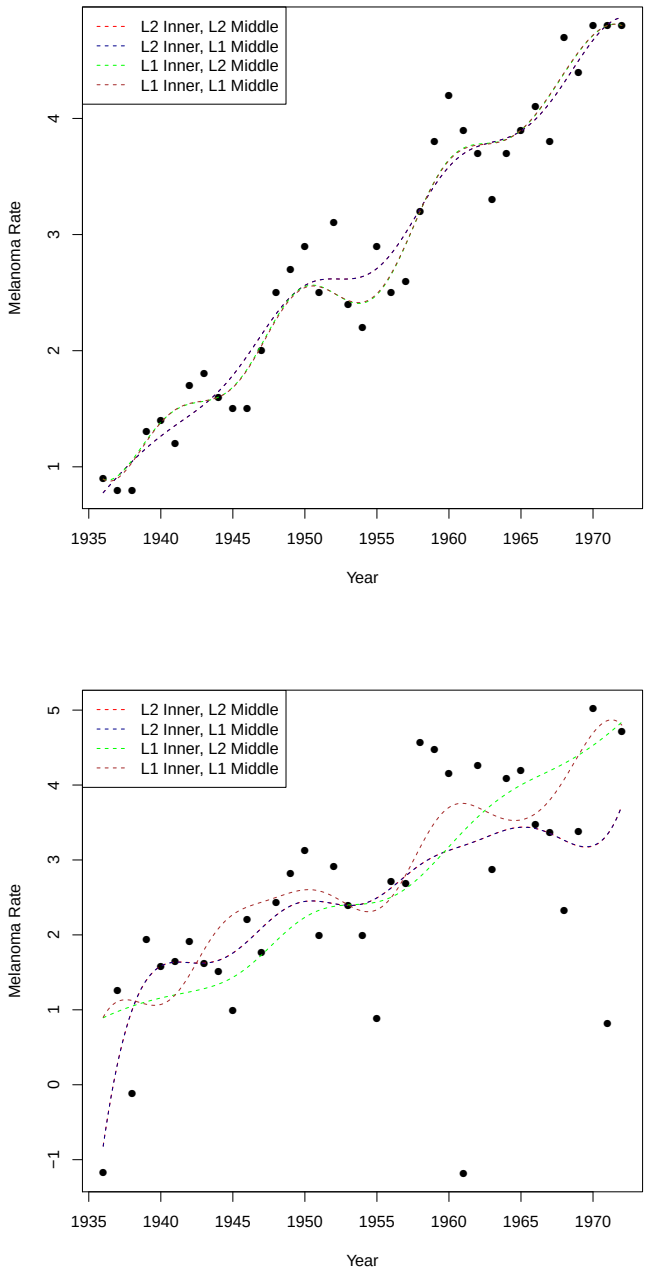



Figure 4.7: All possible combinations of $L_1$ and $L_2$ loss functions that can be used for the Parameter Cascade. The top plot applies them to the orginal melanoma data, the bottom to the same perturbed data as in Figures 4.1 and 4.6

## 4.3 Acclerating The Rate of Convergence

The MM Alogrithm is very sluggish, and this is a well known weakness of both itself and the EM algorithm. The literature however suggest that this problem could be easily ameliorated in this particular case because of a special feature present. In practice one doesn't want to fit the full model, but only wants to compute an associated summary statistic that determines how good a given choice of parameters is. 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 desireable to iterate until the full model converges if this effort can be avoided.

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 [24] instead of running the EM algorithm to completion since the full models aren't needed. The literature on the MM algorithm claims that acceleration methods for the EM algorithm translate quite easily to the MM case [42]. On this basis, we explored whether this approach might be applied here.

The textbook approach employed is known as Aitken Acceleration[8, 24]. Suppose that there is a sequence $x_0, x_1, x_3, \ldots$ 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 accelerted sequence.

There is an equivalent definition that is easier to generalise [13]. 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 one knew the inverse of $g(x)$, which will be 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 Illustrative 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 [24]. 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 again update the estimates of missing values. The process is repeated until convergence. In the text, the authors do not work here with likelihood or any probablistic models and treat the question as purely a regression problem. This is similar to our $L_1$ fitting problem.

The authors' example was implemented again in R.[9]. For each iteration, the $SSE$ statistic was computed. This defines an associated sequence $\{SSE_1, SSE_2, \ldots, SSE_n, \ldots\}$. 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

Outside of more specialised texts, Statistics is generally content with Aitken's method and multivariate generalisations. Exploring more powerful methods can 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 an likelihood ratio statistic computed using the EM Algorithm as previously described. The second is if the sequence is difficult to accelerate. In this situation, 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 $\epsilon_n$ is always zero, and the sequence converges to a limit. Here, the problem of deterimining 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 $\beta_0$ and $\beta_1$ were unknown:

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

[9]The $SSE$ statistic here is different from the $RSS$ statistic presented in the text this example was taken from. The new code converges to the same estimates as in the original, so the example has been re-implemented correctly. It was not possible to determine with what degrees of freedom $RSS$ was associated with.

As $n$ goes to infinity, $x_n$ converges to $\beta_0$. It is not difficult to show that the limit $\beta_0$ 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 (4.4) is applied to a sequence of values $x_1, x_2, \ldots, x_n, \ldots$ that is of form (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 so neat in practice because sequences don't adhere perfectly to these simple forms. Instead, the best that can be realistically hoped for is that the transformed sequence converges to the same limit as the original, 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 $\beta_0$ more quickly than the original sequence.[10]

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 [13] is to use higher order polynomials to model the inverse error function $h(e)$. So $h(e)$ 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 appproximating quadratic at $e = 0$ instead of having to find the correct root of a quadratic to compute each element of the accelerated sequnce.

Another approach is to simply apply Aitken Acceleration to the sequence twice.

Both these approaches were attempted for the missing data model, 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 numerically instability thereafter due to the error terms being so small, so plotting or tabluating them was not worth the additional clutter. 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.

[10] 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 $\tilde{y}_n = \beta_0 + \beta_2 \left[-\frac{2n+1}{n^2(n+1)} + \frac{1}{n^2}\right] = \beta_0 + \beta_2 \left[\frac{-n^3-n^2+n}{n^4(n+1)}\right]$.

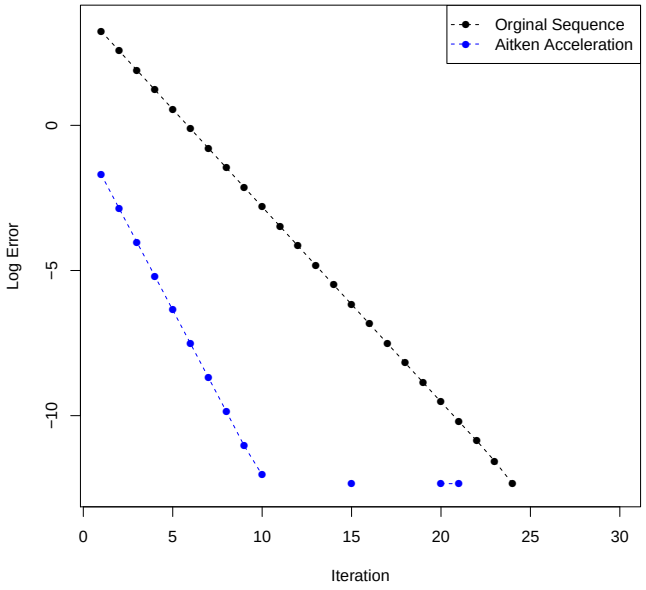



Figure 4.8: Log Errors for original sequnce of SSE values and the accelerated one

**Other Approaches:** There are alternative approaches besides those described here. For example, the EM and MM alogrithms generate a sequence of coeffcient 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 $\mathbf{F}(\cdot)$ would denote 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 such as those described in the chapter on Brent's method to numerically solve this fixed point equation [6, 8]. 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.

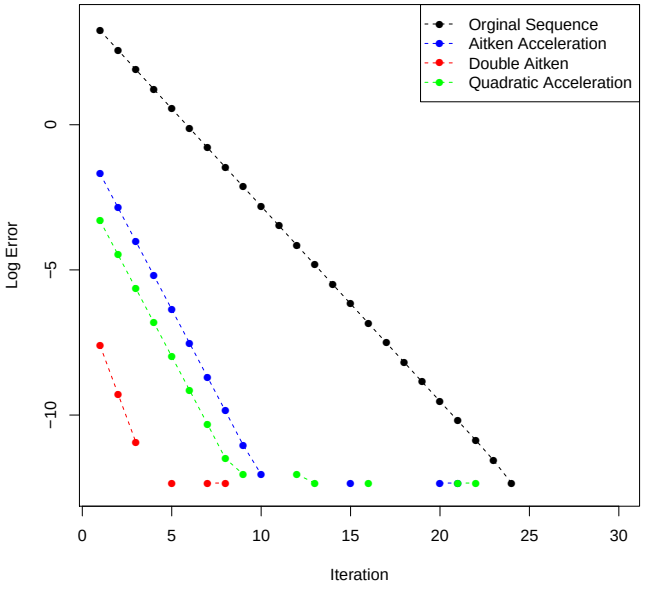



Figure 4.9: More sophisticated acceleration methods can provide a further boost to convergence. There are gaps in the plot because the more accelerated iterations have no valid log error since R cannot numerically distinguish them from the final limit.

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 |
| $\infty$ | 3203.7829457364 | 3203.7829457364 | 3203.7829457364 | 3203.7829457364 |

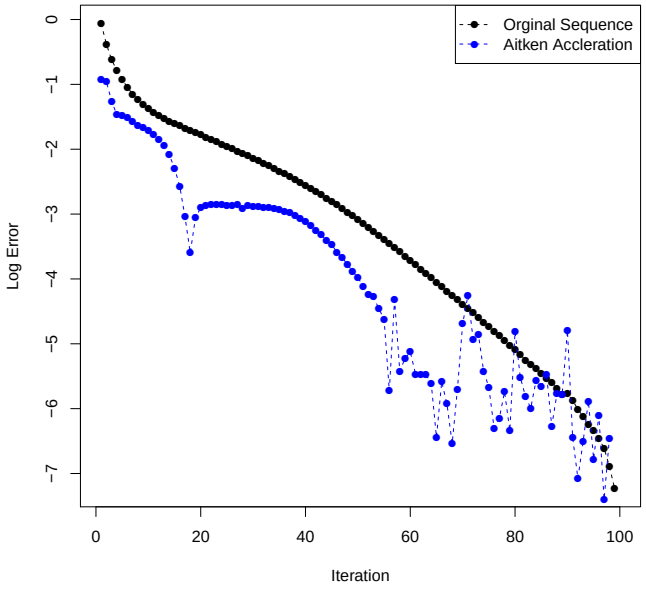



Figure 4.10: Accelerating the SAE sequence generating by the $L_1$ fitting alogrithm using Aitken's Method. The improvement in covergence is mediocre.

### 4.3.3 Accelerating the $L_1$ Fitting Alogrithm

The $L_1$ fitting algorithm is much more difficult to accelerate as can be seen in Figures 4.10 and 4.11. Even advanced acceleration methods recommended for slowly converging sequences that the Aitken method cannot tackle such as the Epsilon Algorithm[11, 26] or Lubkin's W transform [26, 40] yield much improvement[11]. The $SAE$ sequence is apparently 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 causing trouble.

[11]Several other methods proposed in [26] were such as Levin transforms were also attempted. They proved similarly unsatisfactory

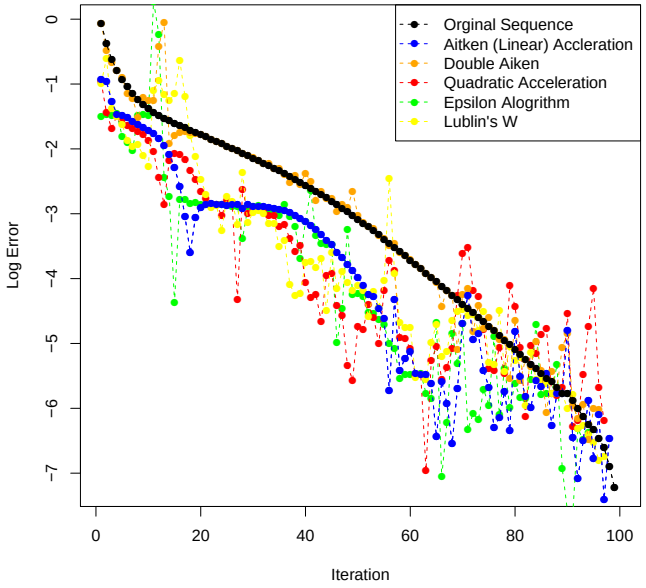



Figure 4.11: Accelerating the SAE sequence using multiple methods. Aitken's method performs the best, despite it's lack of sophistication.

# Chapter 5

# Fitting Non Linear Regression Models With the Parameter Cascade

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

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

All of the difficulty here comes from the $e^{\gamma t}$ term. If $\gamma$ were known, $\alpha$ and $\beta$ 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 $\gamma$. The non-linear model can be fitted 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.

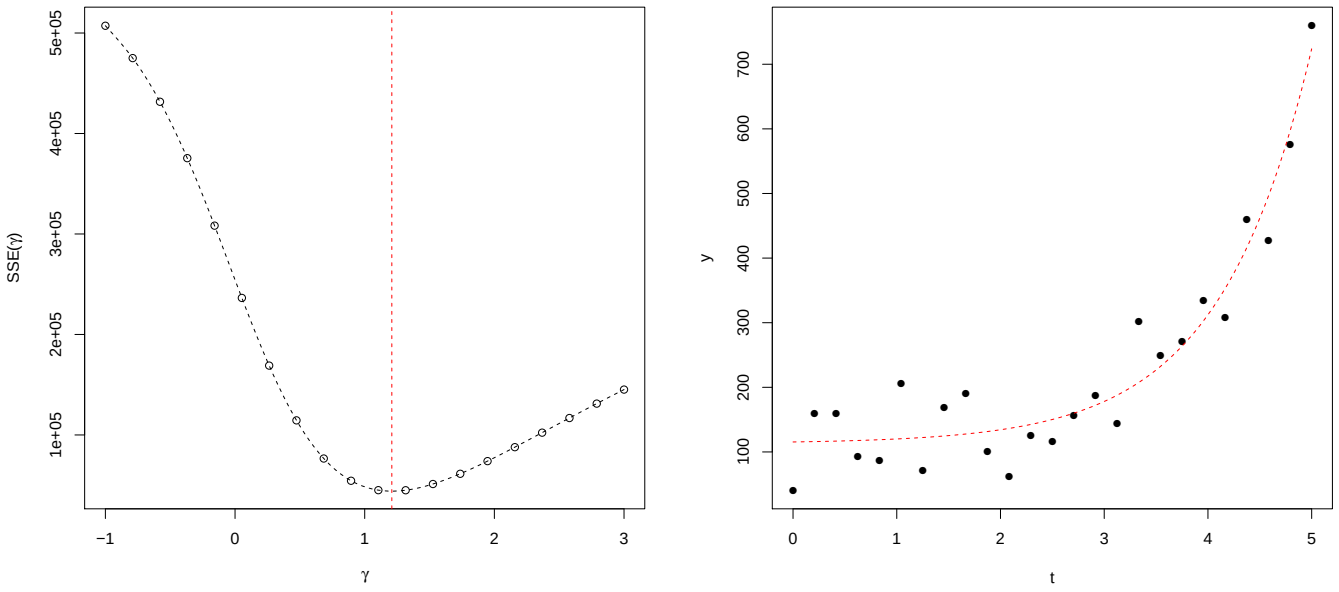



Figure 5.1: Profile Plot and Fitted Curve

## 5.1 Fitting Linear Homogenous ODEs Using the Parameter Cascade

Recall a linear homogenous 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 intial conditions. It is often more convieninet to work with ODEs in matrix form from now on. Any homogenous 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 $i$th basis vector.

So this suggests the following cascade algorithm: given a set of parameters, find the set of solutions $\{y_1(t|\theta), \ldots, y_n(t|\theta)\}$, where $y_k(t|\theta)$ denotes the solution with the $k$th basis vector as an initial condition. Then perform regression to fit the $\{y_i(t|\theta)\}$ to the observed data. The inner problem 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. 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 the previous section. Consider again the ODE $y'' - \omega^2 y = 0$. The solutions generated by the intial 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:

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

Findinng the optimal $a$ and $b$ given $\omega$ 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 be instead found by a numerical solver for each choice of $\theta$. The inner problem then consists of linear regressing the computed solutions against the observed data.

To illustrate the method, it was applied to the following ODE with $\alpha = -0.3$ and $\beta = -1.0$. To minimise the middle problem, the Nelder-Mead method was used - Brent's method was felt to be unsuitable because of the awkward topography. The results can be seen in Figure 5.2.

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

The advantage of the parameter cascade here is that it is noticebly 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 arond 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.

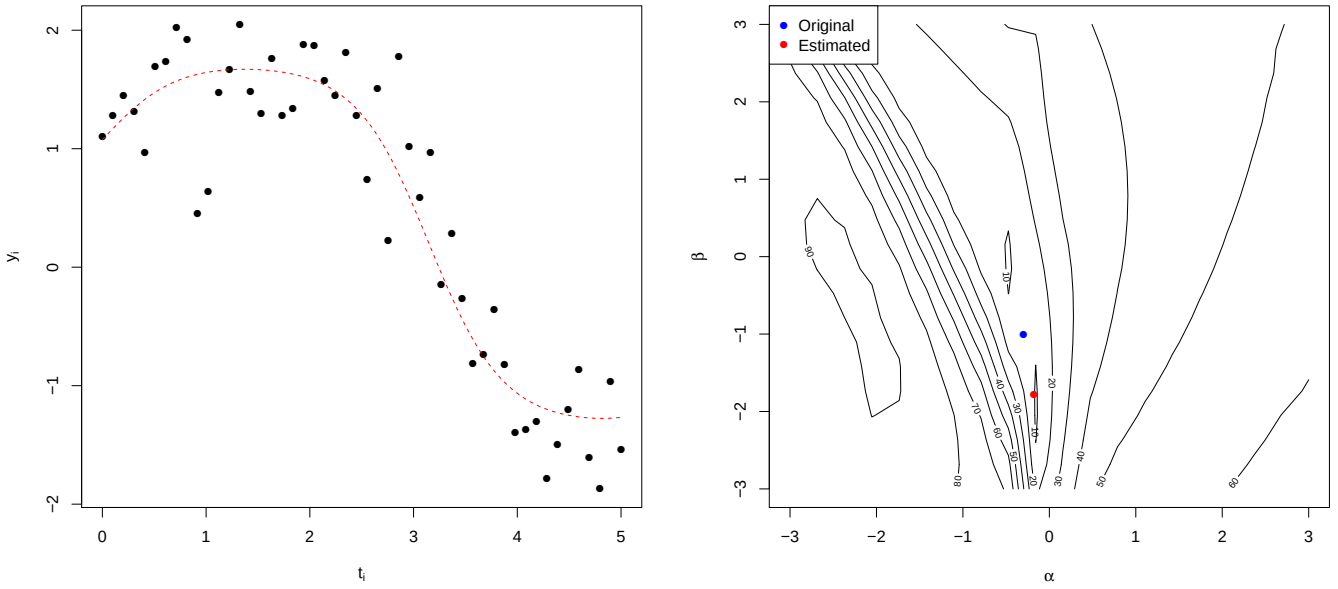



Figure 5.2: Plot of fit to simulated data, and contour plot of SSE against $\alpha$ and $\beta$. Blue dot is true values, red is estimated values.

## 5.2 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.

### 5.2.1 The Transport Equation

In Chapter 2, the Transport equation was introduced. 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, ne a middle objective function $H(\beta)$ to estimate the parameter $\beta$ was defined, but no effort was made to actually fit the model.

Recall that $H(\beta)$ was defined as the sum of squares:

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

And so:

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

To compute the gradient of $H(\beta)$, the estimates of the functions $f(x)$ and $f'(x)$ associated with a given choice of $\beta$ are needed.

This understates the difficulty however. The command `smooth.spline` will only return the GCV score, not the sum of squared errors. So we are forced to use a more complicated objective function than least squares unless a routine to compute them is written.

Fortunately, Brent's Method can be used to minimise $H(\beta)$ instead. As can be seen in Figure 5.3 objective function is irregular though, and care must be taken that one is close to the optimal value already.

Estimating $f(x)$ is harder than estimating $\beta$ as can be seen in Figure 5.4.

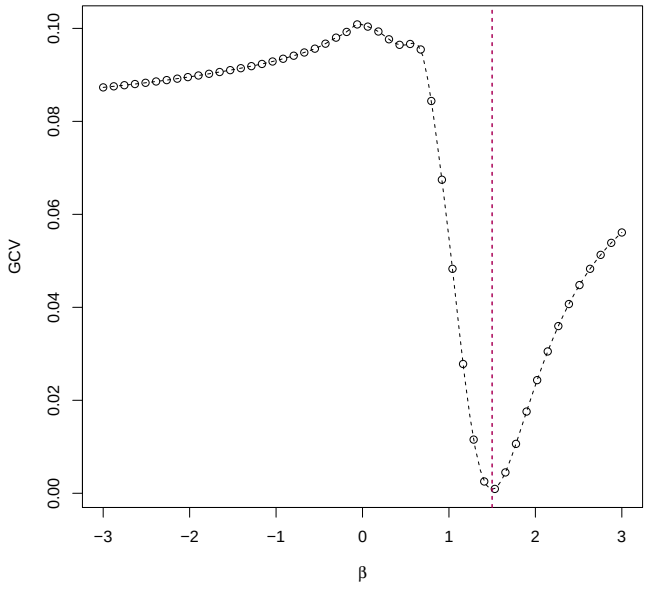
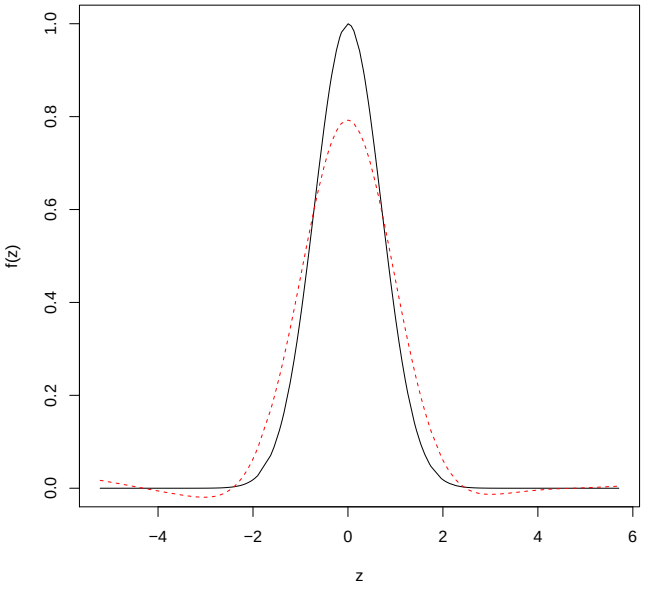
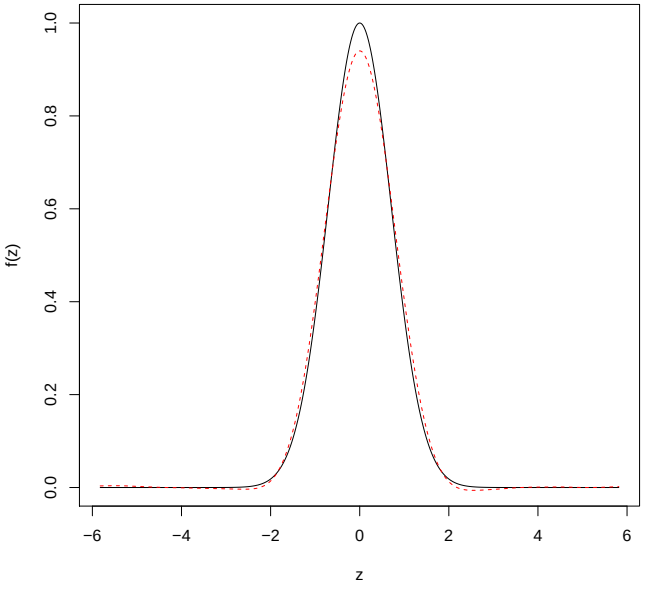



Figure 5.3: Plot of outer optimisation for the transport equation. Blue line denotes original parameter, red line denotes fitted estimate




(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. To avoid confusion between $x$ and $x - \beta t$, $z$ was made the independent variable for this plot. 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.

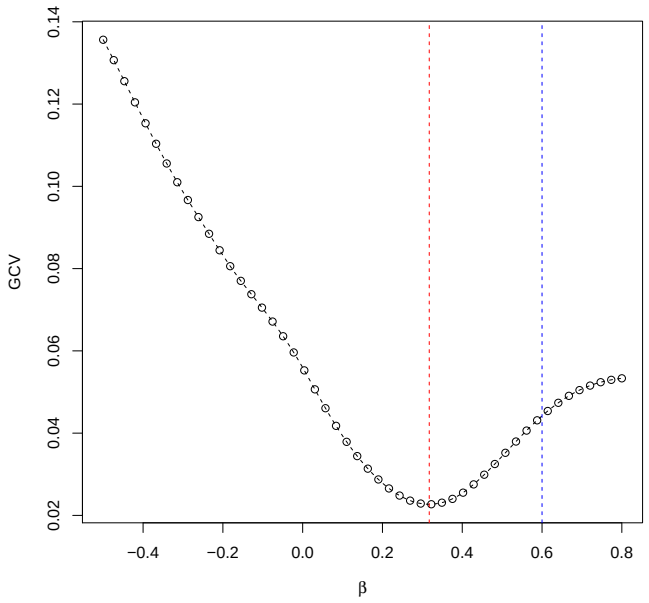



Figure 5.5: Plot of outer optimisation for the modified transport equation with only 30 sample points. Blue line denotes original parameter, red line denotes fitted estimate

### 5.2.2 A Less Trivial PDE

The transport equation is a little trivial, so the methodology will be applied to a more difficult PDE. Instead of having a constant velocity, let it vary with position by having the velocity equal $\beta x$ instead of $\beta$. 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 $\beta$ for this PDE is ill-conditioned in the sense that `smooth.spline` will crash for some meshes.

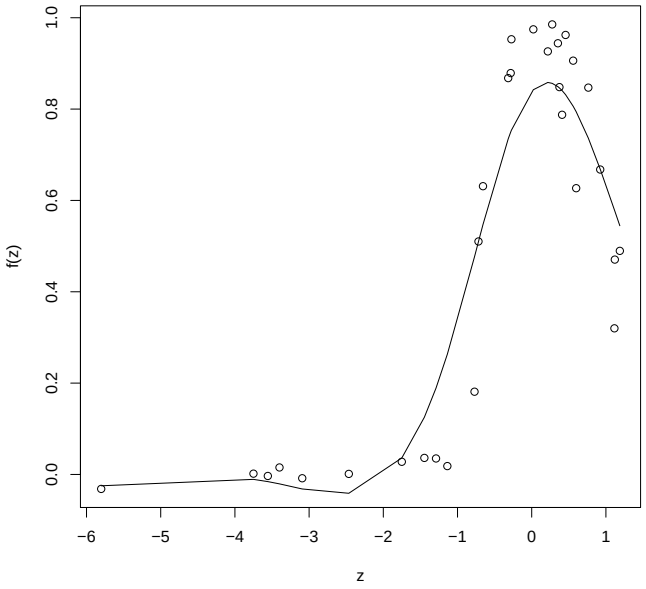



Figure 5.6: Plotted estimate of $f(x)$ for the modified transport equation with 30 sample points.

# Chapter 6

# Conclusion and Further Research

## 6.1 Quasi-linear Differential Problems

Throughout this theis, it has been possible to either 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 touched upon.

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 substanially more complicated, and require more technical knowledge, and even ingenuity 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.

### 6.1.1 Inviscid Burger's Equation

Even if a solution exists, an estimation strategy might be difficult to derive. 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 some arbitary 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 clean seperation between the problem of estimating $f(\cdot)$ and $\beta$ since $u(x,t)$ appears on the righthand side and scales $\beta$.

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.

### 6.1.2 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.

Consider the following quasi-linear model of genetic drift in a population proposed[1] by R.A. Fisher:[9]

$$\frac{\partial u(x,t)}{\partial t} + \beta_1 \frac{\partial u(x,t)}{\partial x} = \beta_2 u(x,t)(1 - u(x,t)) \tag{6.1}$$

This problem is similar to the previous PDEs we discussed, it even admits travelling wave solutions of the form $f(x + ct)$ as Fisher himself noted. Nonetheless, it is a much more difficult problem despite the apparently modest increase in complexity. One would likely have to consult a textbook that covers non-linear PDEs that can generate waves in fair degree of detail to be able to devise a fitting strategy. This is quite a specialised subject!

The overall result is that as the complexity of the differential equation increases, more and more time will be needed to model it correctly.

[1] In the *Annals of Eugenics...*

# Appendices

# Appendix A

# Overview of Optimisation Methods

This appendix provides an overview of the ideas in numerical optimisation used throughout this thesis. [7, 25] are accesible texts for those who wish for more detail.

## 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 $\alpha$ isn't too big. Gradient descent is simple but it only converges linearly under ideal conditions. [7, 25, 41]. If the objective function isn't sufficiently ideal, gradient descent might only converge sublinearly [41].

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 of course 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 readily apparent that the alogrithm is converging sublinearly and that the rate of convergence is absolutely and utterly woeful. It takes around 250 iterations before the error falls below $10^{-3}$, around 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 $10^{-n}$. So it could 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 $\nabla f(\mathbf{x})$. Suppose the second derivatives were available as well. 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 including that so long as one is already near the optimal point and that the Hessian at the optimal point is not pathological.[7, 13, 19, 20, 25] If these assumptions do not hold, undesriable 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, because the second derivative is zero at the optimal point.[1][13] While the Newton-Raphson method is undeniably a huge improvement over Gradient Descent here, it nonetheless completely fails to achieve the usual quadratic convergence.

The biggest weaknesses of Newton's Method the cost of constantly computing the Hessians, the possibilty that $-\mathbf{H}_n^{-1}\mathbf{g}_n$ fails to be a descent direction, and the possibilty 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.[13] 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 $m = 2$.

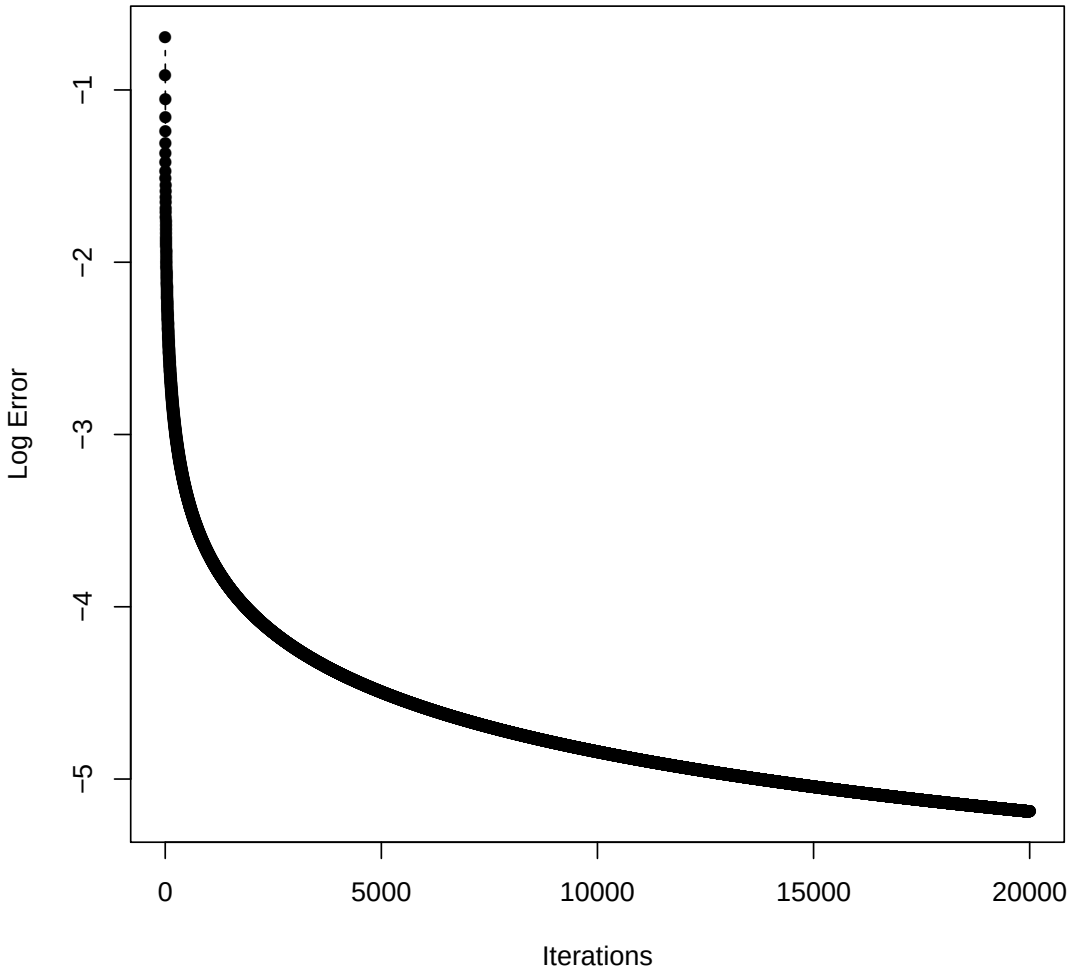



Figure A.1: Plot of the log errors from applying Gradient Descent to the function $f(x) = x^4$.

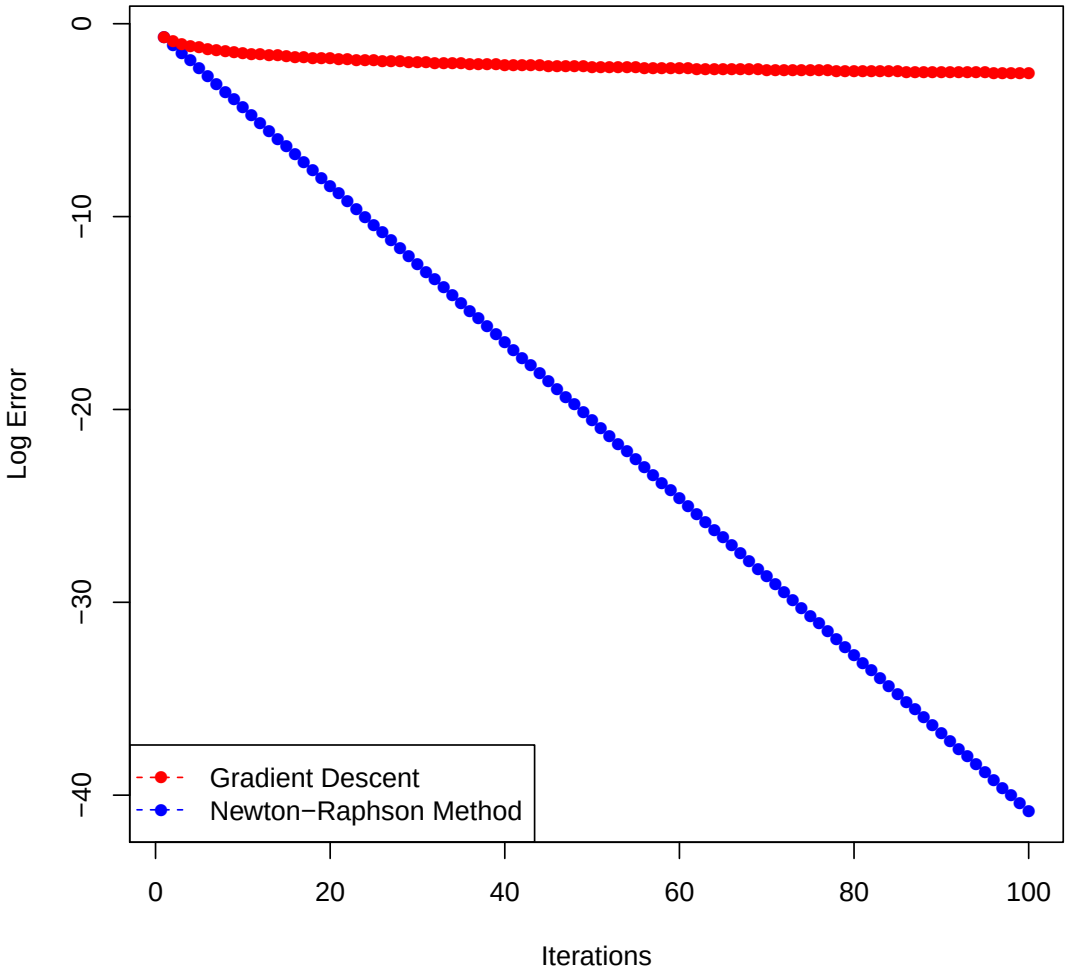



Figure A.2: Plot of the log errors from applying Gradient Descent and the Newton-Raphson Method to the function $f(x) = x^4$.

## A.2 The Chord Method

Chord Methods attempt to approximate the Hessian matrix by using a constant matrix $\mathbf{Q}$. The next 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 might look crude, but it is useful. In some cases for example, it is faster to compute the Hessian once at start and stick with it throughout the entire algorithm than to constantly recompute it [14]. In other words, set $\mathbf{Q} = \mathbf{H}_0$. . This would be epecially so 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 $\mathbf{g}_\mathbf{n}^\top\mathbf{Q}\mathbf{g}_\mathbf{n}$ 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[14], an informal argument will be provided here which closely follows the presentation in [13].Let $\mathbf{g}(\mathbf{x})$ denote the mapping $\mathbf{g}(\mathbf{x}) = \mathbf{x} + \mathbf{Q}\nabla f(\mathbf{x})$. Note 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 $\mathbf{x}^*$ :

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

For brevity, we 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. It is very rarely the case that the Hessian at the limit point is available though. 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:

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

Bearing in mind 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 qudaratics don't have any saddle points or surfaces on which the second derivative is zero so that a well defined search direction is guaranteed. [19, 20, 25]

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]

**Linearly Convergent Methods:** Not every method that changes $\mathbf{Q}_n$ on each interation converges superlinearly. A straightforward example of linear convergence[3] 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, which 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, though at a decent pace so long as it is a reasonable approximation to the observed Fisher Information.[4]

**Quasi-Newton Methods:** Quasi-Newton Methods use the computed gradients to construct approximations to the true Hessians as the algorithm proceeds.[25] These methods produce a sequence of psuedo-Hessians $\mathbf{B}_n$ that satisfy the *Secant Condition:*

[2] As noted in [25], it is actually only required that the $\mathbf{Q}_n$ approximates $\mathbf{H}_n$ along the directions which the alogorithm is searching.

[3] For the sake of simplicity it is assumed that all the methods discussed here do in fact converge.

[4] 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 doesn not mean that Fisher's Method of Scoring achieves superlinear convergence when applied to one specific set of data.

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

In one dimension, finding a $B_n$ that satisfies the Secant Condition is equivalent to computing a finite difference approximation to the second deriviative:

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

When the sequence $x_n$ converges, the denominator in the finite difference approximation converges to zero as well, so the that the rate of convergence is faster than for a finite difference approximimation which uses a fixed denominator $h$.[14]

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[28]. 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[25, 28].

Quasi-Newton Methods converge superlinearly, but are not quadratically convergent like Newton's Method. [25] 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.

**Finite Differences:** In several places in this thesis finite difference methods are used to approximate derivatives.[5]

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. A straightforward scheme to approximated $\mathbf{H}_n$ that only requires additional function evalutions is given by:

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

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:

[5] Refer to [14, 22, 25] for more detail.

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

The approximation error will first shrink and then get larger and larger as 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 question of the convergence rate for the finite difference method is more complex 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[6] the errors satisfy the following recurrence relation [14]:[7]

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

If the objective function is evaluated to a degree of accuracy close to machine precision, it would be 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 were 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.
- If $\|\mathbf{e}_n\|$ is of similar order of magnitude to $h$ so that $\|\mathbf{e}_n\| \lesssim Ch$ for $C > 0$; 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 guranteed.

To surmise: 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 $\bar{\epsilon}$.

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

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

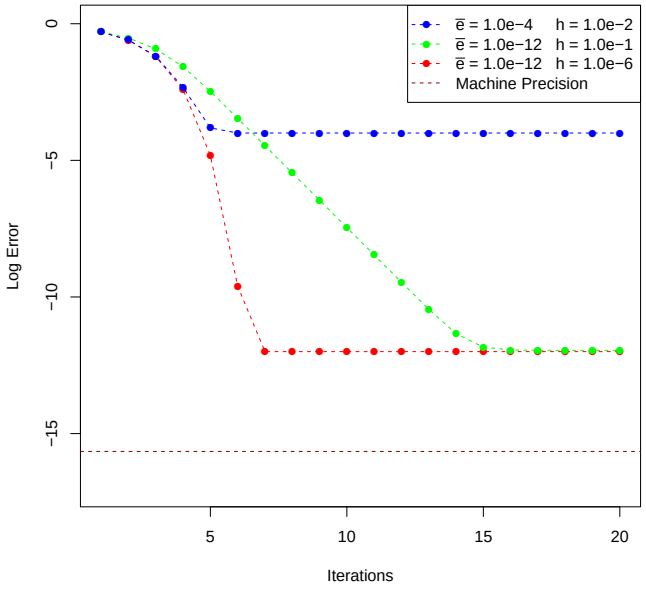



Figure A.3: Plot of the log iterates produced from the recurrence relation $e_{n+1} = \bar{e} + (e_n + h)e_n$ with $e_0 = 0.5$, for various choices of $\bar{e}$ and $h$.

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.4 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 $\alpha$ is small enough, then 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 approxmition to 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 funtion 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 $\alpha$ to some very small value. The first problem with this approach is that $\alpha$ might be unnecessarily small so that convergence is needlessly slow. The second problem is that the quality of a choice of $\alpha$ is governed by the higher order derivatives of the objective function, especially the higher order derivatives at the optimal point [8]. It will often be the case that such information is not available. Imagine the frustration if one had started an optimisation routine with a value of $\alpha$ that looked reasoanable for the initial values, and then came back 12 hours later to find the routine had failed as it approached the optimal point because it $\alpha$ was ultimately too high. Third, a globally valid choice of $\alpha$ might be far too low in some places by definition. More flexibility would allow $\alpha$ to be bigger where viable to gain a faster rate of convergence.

The more sophisticated approach is to allow $\alpha$ to vary on each iteration:

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

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 $\{\mathbf{x}_n + \alpha\mathbf{p}_n | \alpha > 0\}$ 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 of course attempt to find the value $\alpha^*$ 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. [25]

Besides setting simply $\alpha_n = \alpha$ for all $n$, the simplest line search method is known as a *Backtracking Line Search*. In this case, $\alpha$ is set to an initial value $\alpha_0$ 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(x, p)
    α ← α_0
    while f(x + αp) is unsatisfactory do
        α ← ρα
    end while
    return (x + αp)
end function
```

In order to ensure the line search method converges,[9] the steps taken are required

[8] The literature very often works with Lipschitz continuity instead of differentiability. For simplicity in exposition, diffentiability will be used here.

[9] There are other conditions, but the Wolfe conditions are standard. Refer to [25] for more detail

to satisfy the Wolfe Conditions.[10]

$$f(\mathbf{x}_n + \alpha_n \mathbf{p}_n) \leq f(\mathbf{x}_n) + c_1 \alpha_n \mathbf{p}_n^\top \nabla f(\mathbf{x}_n) \quad \textbf{(W1)}$$
$$|\mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha_n \mathbf{p}_n)| \leq c_2 |\mathbf{p}_n^\top \nabla f(\mathbf{x}_n)| \quad \textbf{(W2)}$$

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

$$\phi(\alpha_n) \leq \phi(0) + c_1 \alpha_n \phi'(0) \quad \textbf{(W1')}$$
$$|\phi'(\alpha_n)| \leq c_2 |\phi'(0)| \quad \textbf{(W2')}$$

The first condition ensures sufficient decrease in the objective function, the second ensures a sufficent 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.4.1 More sophisticated Line Search Algorithms

More sophisticated line search alogrithms 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 $\alpha$. 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 $\alpha$ 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 thorugh the values $\phi(0), \phi'(0), \phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$. The next value of $\alpha$ 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[11] they are also sometimes unstable and can have multiple critical points.

More exotic choices proposed in the literature include the posibility of using rational functions to approximate $\phi(\alpha)$ instead of polynomials, as discussed in [1]. An example of such an approximation is:

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

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

[10] Strictly, these are the Strong Wolfe conditions. The usual Wolfe conditions only require that $-\mathbf{p}_n^\top \mathbf{g}_{n+1} \leq -c_2 \mathbf{p}_n^\top \mathbf{g}_n$.

[11] 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 [37].

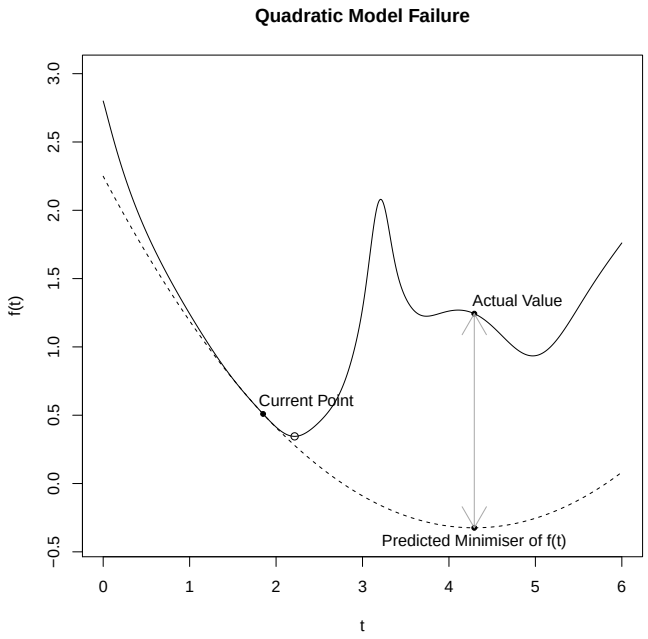



Figure A.4: Extrapolating too far out can lead to disaster!

# 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 for the sake of argument 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 algorithm is designed for optimising problems where the exact value of the objective function $f(\cdot)$ is unavailable. Instead, it is assumed that there is parameter $h$ which controls the degree of the accuarcy. It is further assumed that the lower $h$, the lower the error. It is usually the case that getting a higher degree of accuaracy means a higher run time.

For example, if the objective function is of the form $\mathbb{E}[f(\theta)]$ and the expection 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 evauluating $f(\cdot)$ at the point $\mathbf{x}$ with precision level $h$ and $\epsilon(\mathbf{x}; h)$ 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 [15, 17, 25].

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

Here $[\nabla_h f(\mathbf{x})]_i$ denotes the $i$th component of $\nabla_h f(\mathbf{x})$ and $\mathbf{e}_i$ is the $i$th 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.[2]

In the event of any of the following occuring, it is deemed that more precision is needed:

- A point achieving sufficient reduction cannot be found after a 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 $\mathbf{x}_n$. Bear in mind however that since Implicit Filtering only ever approximately evaluates the objective function, it is not necessarily the case that $m < n$ implies $f(\mathbf{x}_m) \geq f(\mathbf{x}_n)$.

A variation of the Gauss-Newton algoritm of 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.[16]

## B.2 Convergence Theory

For Implicit filtering to converge, the error needs to decrease sufficiently rapidly relative to sequence of precision parameters $h_n$. The specific condtions depend on the specific variation of Implicit Filtering used.[15, 16, 17, 25] The convergence criteria generally look something like:

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

[2]More precisely, the next point $\mathbf{x}_{k+1}$ is required to satisfy a condition of the form $f(\mathbf{x}_{k+1}; h) \leq f(\mathbf{x}_k; h) - c\mathbf{d}_k^\top [\nabla_h f(\mathbf{x}_k)]$, where $\mathbf{d}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $0 < c < 1$. Note that there is no requirement to decrease the norm of the approximate gradient $\nabla_h f(x)$.

In a statistical context, being too naive and 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 (B.2) gives that

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

In this case, there is no guarantee that the method would 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.

## B.3 Assessment of the Method

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

- Implicit Filtering is complex to code, and is thus difficult to maintain and debug. The R code used to fit the ODE (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.4 below.
- The results of the fitting are sensitive to the value of the shrink factor $\delta$ choosen.
- It can be necessary to add a penalty term to the objective function to ensure convergence.

## B.4 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:[3]

[3]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).

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

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

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

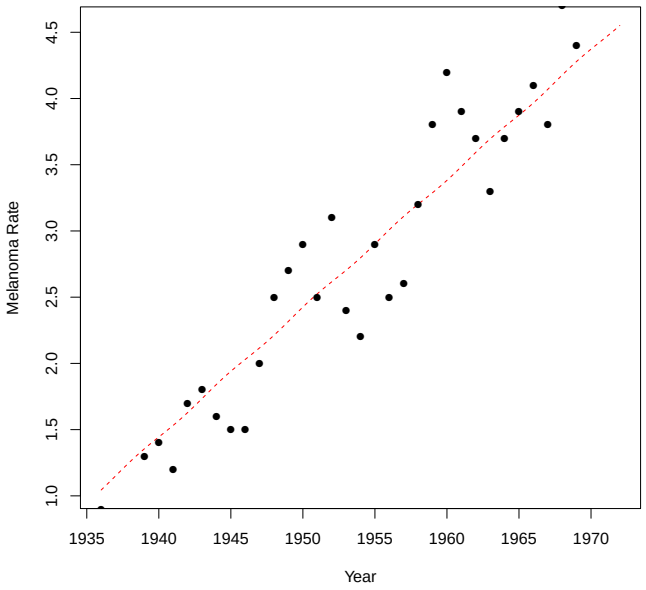
The value of $PENSSE$ is influenced by $\omega$ and $\mu$ because $f(t)$ is required to be a solution of (B.3) with given values of $\omega$ and $\mu$. The Implicit Filtering algorithm will not converge correctly without the penalty term as illustrated in Figure B.2.

To compute $PENSSE$, the package `deSolve` was used to numerically solve (B.3) with the appropriate values of $\omega$ and $\mu$. The precision factor $h$ determined the stepsize 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.

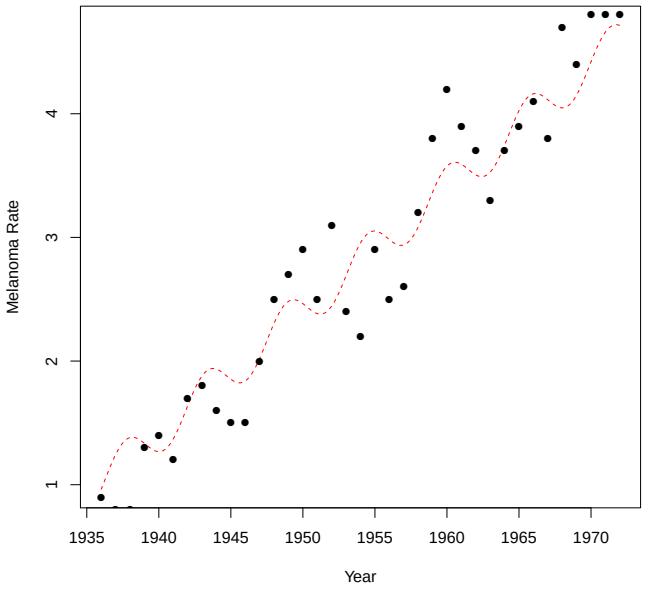
As can be seen in Table B.1, the algorithm takes a long time to run. It can be seen in both the table and Figure B.1 that changing the value of $\delta$ can introduce qualitative changes in behaviour. The algorithm is much quicker for $\delta = 0.9$, presumably because the algorithm is converging to 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 inadaquate in the course of its execuction and further evaluations are needed to compute the approximate gradients $\nabla_h f(\cdot)$. For case where $\delta = 0.9, PENSSE$ was computed over 3700 times with various values of $h$ used.

| $\delta$ | 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 $\delta$.




(a)




(b)

Figure B.1: Fitting the ODE (B.3) to the Melanoma data. The exact value of the shrink value $\delta$ effects 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 sinusodial plus linear trend as can be seen in (b).

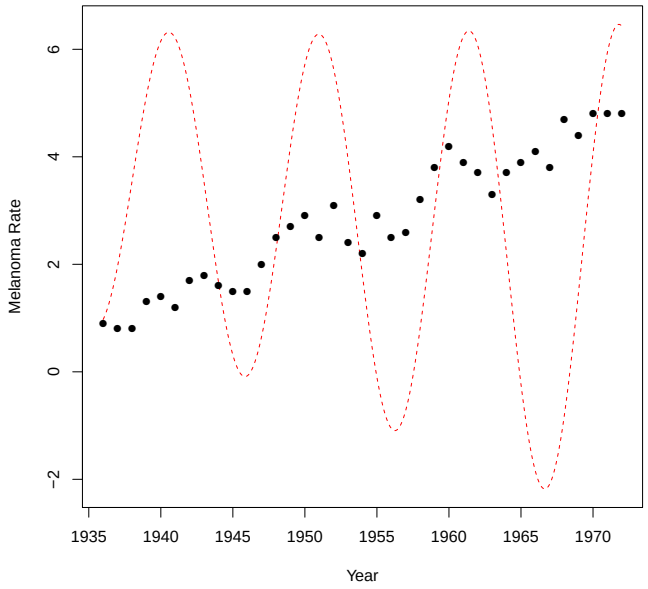



Figure B.2: Without a penalty term, Implicit Filtering entirely fails to fit the ODE (B.3) to the melanoma data.

# Bibliography


[1] Jonathan Barzilai and Aharon Ben-Tal. Nonpolynomial and inverse interpolation for line search: synthesis and convergence rates. *SIAM Journal on Numerical Analysis*, 19(6):1263–1277, 1982.

[2] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[5] Jiguo Cao and James O Ramsay. Parameter cascades and profiling in functional data analysis. *Computational Statistics*, 22(3):335–351, 2007.

[6] Kwun Chuen Gary Chan. Acceleration of expectation-maximization algorithm for length-biased right-censored data. *Lifetime data analysis*, 23(1):102–112, 2017.

[7] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 76. John Wiley & Sons, 2013.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[9] Ronald Aylmer Fisher. The wave of advance of advantageous genes. *Annals of eugenics*, 7(4):355–369, 1937.

[10] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

[11] PR Graves-Morris, DE Roberts, and A Salam. The epsilon algorithm and related topics. *Journal of Computational and Applied Mathematics*, 122(1-2):51–80, 2000.

[12] David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[13] Eugene Isaacson and Herbert Bishop Keller. *Analysis of numerical methods*. Courier Corporation, 2012.

[14] Carl T Kelley. *Iterative methods for linear and nonlinear equations*, volume 16. Siam, 1995.

[15] Carl T Kelley. *Implicit filtering*, volume 23. SIAM, 2011.

[16] CT Kelley. Implicit filtering and nonlinear least squares problems. In *IFIP Conference on System Modeling and Optimization*, pages 71–90. Springer, 2001.

[17] C.T. Kelley. A brief introduction to implicit filtering. `https://projects.ncsu.edu/crsc/reports/ftp/pdf/crsc-tr02-28.pdf`, 2002. [Online; accessed 12-October-2019].

[18] Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3):502–506, 1953.

[19] Kenneth Lange. *Optimization*. Springer, 2004.

[20] Kenneth Lange. *Numerical analysis for statisticians*. Springer Science & Business Media, 2010.

[21] Kenneth Lange. The MM algorithm. `https://www.stat.berkeley.edu/~aldous/Colloq/lange-talk.pdf`, April 2007. [Online, accessed 18-September-2019].

[22] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam, 2007.

[23] Steve McConnell. *Code complete*. Pearson Education, 2004.

[24] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[25] J Nocedal and SJ Wright. *Numerical Optimisation*. Springer verlag, 1999.

[26] Naoki Osada. *Acceleration methods for slowly convergent sequences and their applications*. PhD thesis, Nagoya University, 1993.

[27] Yudi Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.

[28] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[29] J. O. Ramsay, Hadley Wickham, Spencer Graves, and Giles Hooker. *fda: Functional Data Analysis*, 2018. R package version 2.4.8.

[30] James Ramsay. Functional data analysis. *Encyclopedia of Statistics in Behavioral Science*, 2005.

[31] Jim O Ramsay, Giles Hooker, David Campbell, and Jiguo Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[32] JO Ramsay, G Hooker, and S Graves. *Functional data analysis with R and MATLAB*. Springer Science & Business Media, 2009.

[33] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[34] Larry Schumaker. *Spline functions: basic theory*. Cambridge University Press, 2007.

[35] Christopher G Small. A survey of multidimensional medians. *International Statistical Review/Revue Internationale de Statistique*, pages 263–277, 1990.

[36] Karline Soetaert, Thomas Petzoldt, and R. Woodrow Setzer. Solving differential equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010.

[37] Arie Tamir. Line search techniques based on interpolating polynomials using function values only. *Management Science*, 22(5):576–586, 1976.

[38] Gerald Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.

[39] Keller Vandebogart. Method of quadratic interpolation. `http://people.math.sc.edu/kellerlv/Quadratic_Interpolation.pdf`, September 2017. [Online; accessed 13-September-2019].

[40] Jet Wimp. *Sequence transformations and their applications*. Elsevier, 1981.

[41] Stephen Wright. Optimization for data analysis. In Michael W. Mahoney, John C. Duchi, and John C. Duchi, editors, *The Mathematics of Data*, chapter 2, pages 49–98. American Mathematical Society and IAS/Park City Mathematics Institute and Society for Industrial and Applied Mathematics, 2018.

[42] Tong Tong Wu, Kenneth Lange, et al. The MM alternative to EM. *Statistical Science*, 25(4):492–505, 2010.