Appendix A

Rates Of Convergence[1]

Throughout this thesis, it will sometimes be desireable to compare the rates of convergence of different methods. Suppose there is a vector-valued sequence $\mathbf{x}_n$ that converges to a value $\mathbf{x}^*$.

Linear Convergence: A convergent sequence is said to *converge linearly*[2] to $\mathbf{x}^*$ (with convergence rate μ) if there is a $0 < \mu < 1$ such that:

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

If a sequence $\mathbf{x}_n$ converges linearly with constant μ, then $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \approx \mu\|\mathbf{x}_n - \mathbf{x}^*\|$ for n sufficiently large. A simple example of a linearly converging sequence is given by $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ 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: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$.

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

[2] 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 A.1: Illustrating the different classes of convergence.

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:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|^p} = \mu \tag{A.2}$$

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 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 (A.2). To cover these situations, a sequence is also said to converge linearly/sublinearly/superlinearly if there is an associated auxillary sequence ϵ_n such that $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq \epsilon_n$ for all $n \geq 0$, and the sequence ϵ_n converges linearly/sublinearly/superlinearly to zero.[3]

Linear Convergence and Iterated Mappings: Nearly all estimation algorithms used in statistics start with an initial estimate θ_0 and generate a sequence of estimates by $\theta_{n+1} = \mathbf{M}(\theta_n)$ for some mapping $\mathbf{M}(\cdot)$. The alogrithm is stopped when the generated

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

sequence has converged within a tolerance of the limit θ^*. Examples include the Newton-Raphson Method, Fisher's Method of Scoring, Gradient Descent, the EM Algorithm, Block Relaxation, and many imputation methods. As shall be seen, a statistically 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:

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

The situaion 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 $\|f'(\mathbf{x}^*)\| < 1$ the sequence will converge linearly at worst, though faster than linear convergence is potentially possible if 0 is an eigenvalue of $\mathbf{f}'(\mathbf{x}^*)$.[4] If $\mathbf{f}'(\mathbf{x}^*) = \mathbf{0}$, the convergence will be superlinear.

[4]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.

Appendix B

Overview of Optimisation Methods

More sophisticated optimisation algorithms such as the Newton-Raphson method are specifically designed so that $\mathbf{f}'(\mathbf{x}^*) = 0$, so that the convergence is superlinear.

This chapter provides an overview of the ideas in optimisation used throughout this thesis. [7, 25] are accesible texts for those who wish for more detail. It starts by defining Orders of Convergence.

B.1 Gradient Descent and the Newton-Raphson Method

The simplest derivative-based optimisation algorithm is known as *Gradient Descent*:

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

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

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

This means that $f(\mathbf{x}_{n+1}) < f(\mathbf{x}_n)$ so long as α isn't too big. Gradient descent is simple but 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 B.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 B.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 $m = 2$.

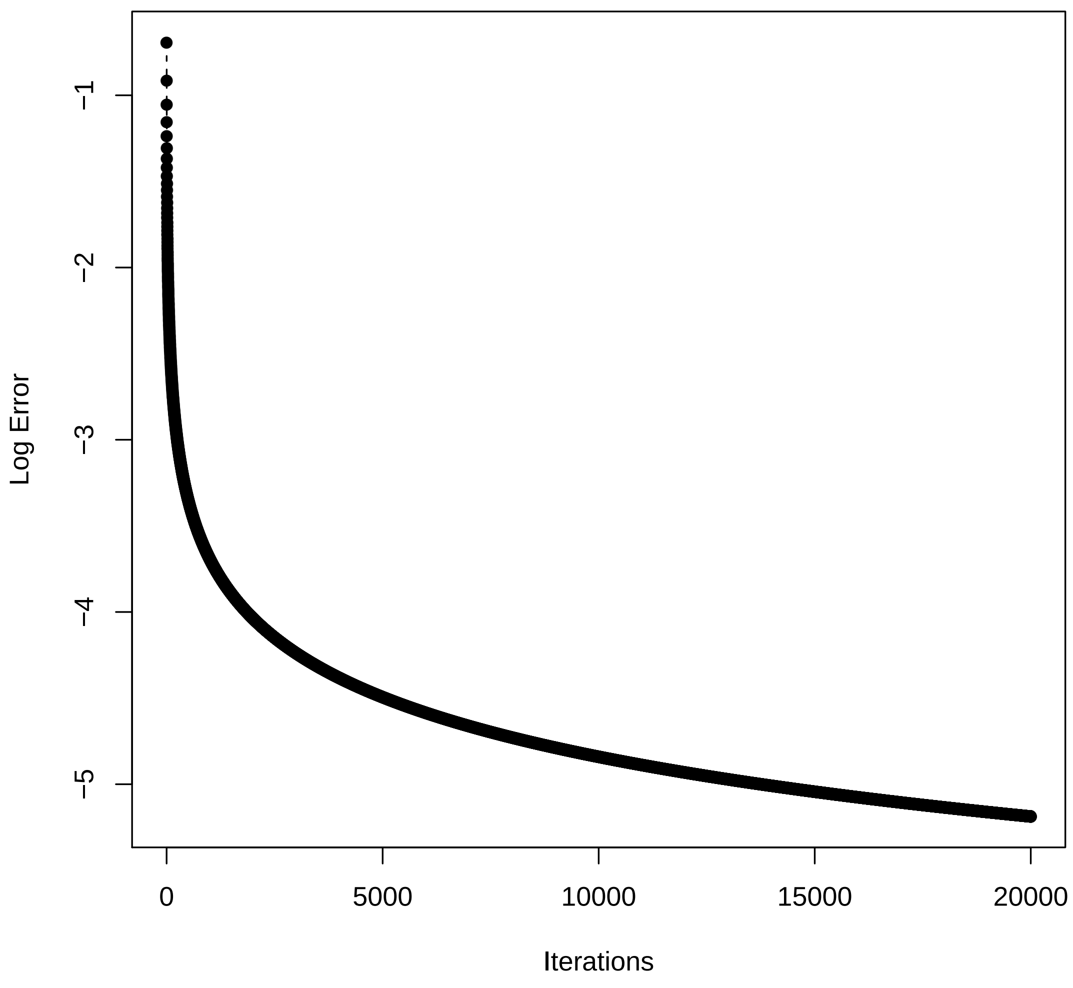


Figure B.1: Plot of the log errors from applying Gradient Descent to the function $f(x) = x^4$.

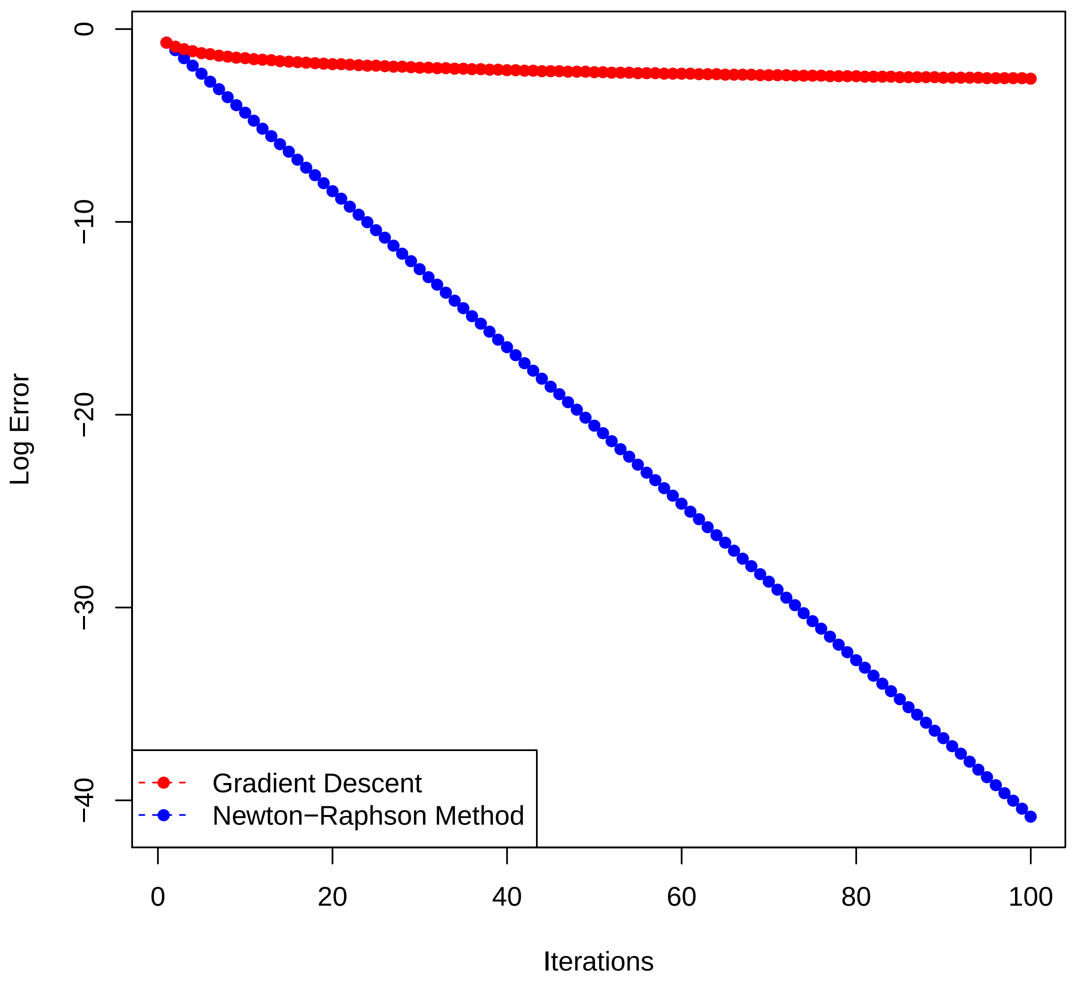


Figure B.2: Plot of the log errors from applying Gradient Descent and the Newton-Raphson Method to the function $f(x) = x^4$.

B.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 $\mathbf{g_n}^\top\mathbf{Q}\mathbf{g_n}$ since:

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

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

It can be shown that the Chord Method converges linearly[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.

B.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 B.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 B.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]

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

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.

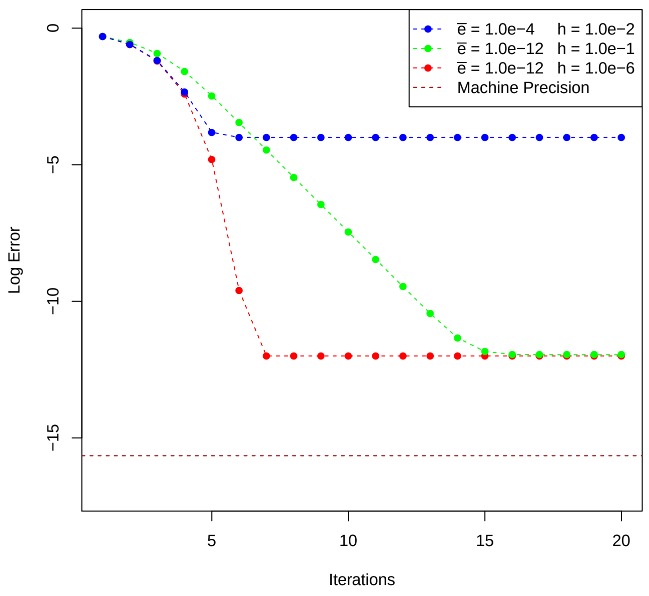


Figure B.3: Plot of the log iterates produced from the recurrence relation $e_{n+1} = \bar{e} + (e_n + h)e_n$ with $e_0 = 0.5$, for various choices of $\bar{e}$ and h.

Figure B.3 illustrates the recurrence relation in Equation B.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.

B.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:

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

This means that if α 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 B.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 α to some very small value. The first problem with this approach is that α might be unnecessarily small so that convergence is needlessly slow. The second problem is that the quality of a choice of α is governed by the higher order derivatives of the objective function, especially the higher order derivatives at the optimal point [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 α 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 α was ultimately too high. Third, a globally valid choice of α might be far too low in some places by definition. More flexibility would allow α to be bigger where viable to gain a faster rate of convergence.

The more sophisticated approach is to allow α to vary on each iteration:

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

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 α^* that minimises $\phi(\alpha)$, but this is generally regarded as excessive. Usually one only iterates until sufficient improvement has been found and uses that point for the next iteration. [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, α is set to an initial value α_0 and repeatedly shrunk down to $\rho\alpha$, where $0 < \rho < 1$ until a satisfactory point is found.

Algorithm 1 Sketch of Backtracking Line Search

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

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]

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

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

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

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.

B.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 α. The constructed quadratic $m(\alpha)$ has the properties that $m(0) = \phi(0), m'(0) = \phi'(0)$ and $m(\bar{\alpha}) = \phi(\bar{\alpha})$ The next value of α is taken to be the minimiser of $m(\alpha)$.

If the line search has been running for longer, so that values $\phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$ are available, then cubic interpolation is possible thorugh the values $\phi(0), \phi'(0), \phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$. The next value of α is taken as the minimum of the interpolating cubic, which entails solving a quadratic equation. While cubic methods can converge more rapidly than quadratic methods[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:

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

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].

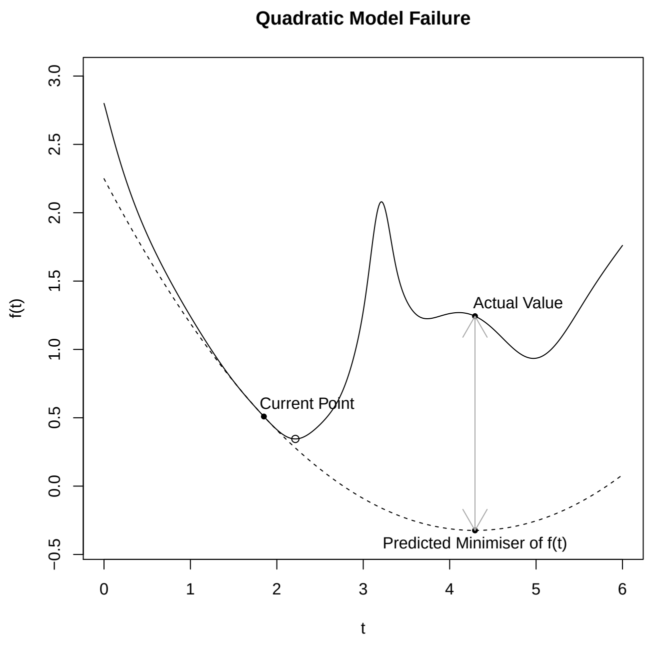


Figure B.4: Extrapolating too far out can lead to disaster!

Appendix C

Implicit Filtering[1]

The weakness of the finite difference approach discussed in Section B.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.

C.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].

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

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

C.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:

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

[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 (C.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.

C.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 (C.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 C.4 below.
- The results of the fitting are sensitive to the value of the shrink factor δ choosen.
- It can be necessary to add a penalty term to the objective function to ensure convergence.

C.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 (C.1).

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

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

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

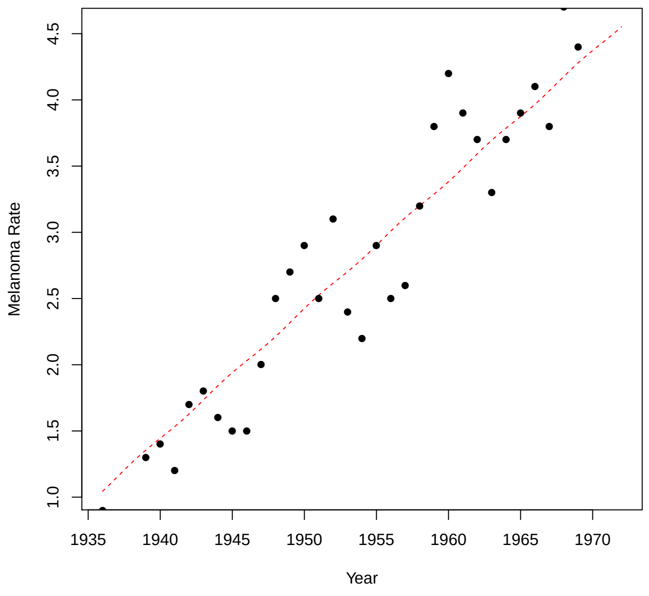
The value of $PENSSE$ is influenced by ω and μ because $f(t)$ is required to be a solution of (C.3) with given values of ω and μ. The Implicit Filtering algorithm will not converge correctly without the penalty term as illustrated in Figure C.2.

To compute $PENSSE$, the package `deSolve` was used to numerically solve (C.3) with the appropriate values of ω and μ. 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.

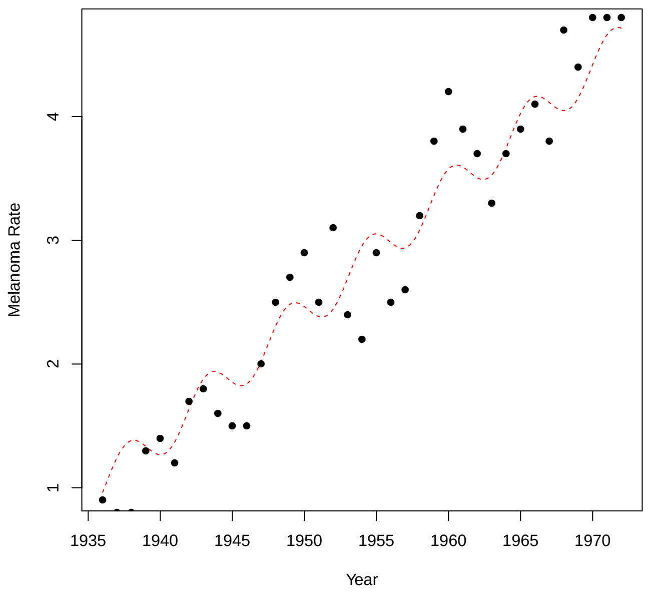
As can be seen in Table C.1, the algorithm takes a long time to run. It can be seen in both the table and Figure C.1 that changing the value of δ can introduce qualitative changes in behaviour. The algorithm is much quicker for $\delta = 0.9$, presumably because the algorithm is converging to a different fit than for the other cases. For the fastest case where $\delta = 0.9$, 200 values of the $PENSSE$ sequence are generated before the sequence converges to within a tolerance of 10^{-4}. This statistic substantially underestimates the actual amount of work done since Implicit Filtering rejects many evaluations as being inadaquate in the course of its exechution 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.

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

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



(a)



(b)

Figure C.1: Fitting the ODE (C.3) to the Melanoma data. The exact value of the shrink value δ 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).

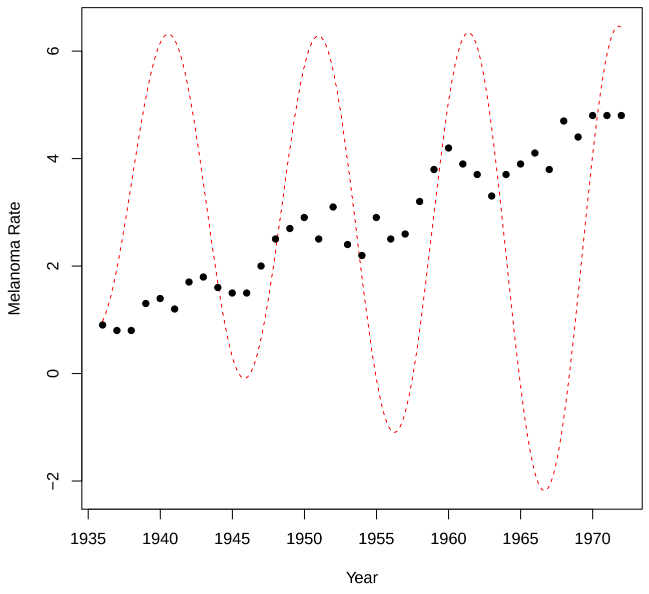


Figure C.2: Without a penalty term, Implicit Filtering entirely fails to fit the ODE (C.3) to the melanoma data.

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.