INFORMATION GEOMETRY FOR NONLIENAR LEAST-SQUARES DATA FITTING AND NUMERICAL CALCULATION OF THE SUPERCONDUCTING SUPERHEATING FIELD

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mark K. Transtrum

August 2011

INFORMATION GEOMETRY FOR NONLIENAR LEAST-SQUARES DATA FITTING AND NUMERICAL CALCULATION OF THE SUPERCONDUCTING SUPERHEATING FIELD

Mark K. Transtrum, Ph.D.

Cornell University 2011

This thesis consist of two parts, each of which consist of two chapters. First we explore the information geometric properties of least squares data fitting, particularly for so-called "sloppy" models. Second we describe a calculation of the superconducting superheating field, relevant for advancing gradients in particle accelerator resonance cavities.

Parameter estimation by nonlinear least squares minimization is a ubiquitous problem that has an elegant geometric interpretation: all possible parameter values induce a manifold embedded within the space of data. The minimization problem is then to find the point on the manifold closest to the data. By interpreting nonlinear models as a generalized interpolation scheme, we find that the manifolds of many models, known as sloppy models, have boundaries and that their widths form a hierarchy. We describe this universal structure as a hyper-ribbon. The hyper-ribbon structure explains many of the difficulties associated with fitting nonlinear models and suggests improvements to standard algorithms. We add a "geodesic acceleration" correction to the standard Levenberg-Marquardt algorithm and observe a dramatic increase in success rate and convergence speed on many fitting problems.

We study the superheating field of a bulk superconductor within the Ginzburg-Landau, which is valid only near T_c, and Eilenberger theory, which is valid at all temperatures. We calculate as functions of both the Ginzburg-Landau parameter κ and reduced temperature $t = T/T_c$ the superheating field $H_{\rm sh}$ and the critical momentum k_c describing the wavelength of the unstable perturbations to flux penetration. By mapping the two-dimensional linear stability theory into a one-dimensional eigenfunction problem for a linear operator, we solve the problem numerically. Within the Ginzburg-Landau theory, We demonstrate agreement between the numerics and analytics, and show convergence to the known results at both small and large κ. Within the Eilenberger theory we demonstrate agreement with the results of Ginzburg-Landau theory near T_c, but find discrepancies with the temperature-dependent results for large κ. We speculate that this discrepancy is due to a lack of convergence at low temperatures due to small length scales of the perturbations analogous to the small length scales associated with the vortex cores of the mixed state.

BIOGRAPHICAL SKETCH

The author was born and raised in a small town in Southeastern Idaho, before moving to Pinedale, Wyoming during High School. He attended a year of school at Brigham Young University in Provo, Utah before spending two years in the area around Rome, Italy serving a mission for the Church of Jesus Christ of Latter Day Saints. Upon returning to BYU, he received a BS with a double major in Physics and Mathematics. While in Provo, he married his wife Jenny, who then accompanied him to Ithaca where he has pursued graduate studies in Physics. Upon graduating, Mark and Jenny, now joined by their one-year-old daughter Camilla, will move to Houston, Texas where Mark will study modeling of complex biological systems at M. D. Anderson Cancer Center.

To my beautiful wife, who agreed to follow me to Ithaca. The road has indeed been long.

ACKNOWLEDGEMENTS

It is my personal belief that all science should begin with an acknowledgement of the intellectual predessors upon which the current work is built. I am therefore happy to first acknowledge my indebtedness to the many colleagues and students of my advisor, Jim Sethna, who have thought about Sloppy Models over the last decade. The list must include Chris Myers, Kevin Brown, Josh Waterfall, Ryan Gutenkunst, and Bryan Daniels to name a few.

I would also like to acknowledge the contribution of my own colleagues and coauthors. Ben Machta, whose insights into the sloppy model geometry and algorithm development were very helpful, and Gianluigi Catelani, whose help in understanding Eilenberger theory was crucial, each deserve due recognition. Many others have offered helpful suggestions and guidance; I would be remiss not to mention Saul Teukolsky, Georg Hoffstaetter, Cyrus Umrigar, and the other members of the Sethna group, all of whom have provided help feedback and made work exciting and enjoyable.

Of course my advisor Jim Sethna deserves much credit. His guidance and direction at several critical moments were crucial. In addition he has taught me more about doing science that I could ever describe here, and much that I probably don't yet realize. His obvious enjoyment in solving a challenging problem is contagious and has made that last few years a lot of fun.

Finally but most importantly, I would like to acknowledge my family, particularly my wife, to whom I dedicate this thesis, and my daughter Camilla, whose love and support I value immensely. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis focuses on two very different topics. First, we consider the information geometric properties of nonlinear least squares data fitting, particularly the properties of so-called “sloppy” models. We use these geometric ideas to not only understand properties of nonlinaer models, but to also inspire algorithms for finding best fits. Second we study the metastability of the superconducting Meissner state in an external magnetic field, relevant for the advancing the accelerating gradients in superconducting rf resonance cavities. In this chapter we give a brief introduction to each of these topics.

1.1 Introduction to Sloppy Models

Inferring unknown model parameters from data can be a challenging problem when there are more than a few parameters. A typical approach is to construct a least squares cost function quantifying the deviation of model predictions from experiment which is then minimized. The resulting best fit parameters represent the most likely parameter values, in a maximum likelihood sense, assuming that the errors are Gaussian distributed. In the quadratic approximation, the correlation matrix of parameter uncertainties is then given by the inverse Fisher information matrix. This matrix is usually very ill conditioned, implying that some parameter combinations are very poorly constrained by the data. The ill-posed nature of this inverse problem is the hallmark feature of sloppy models.

Although large inverse problems of this sort have been observed to be ill-posed for some time, only recently has a systematic study of the origin of this phenomenon been undertaken. In the context of systems biology, where differential equation models involving large numbers of reaction parameters and relatively little data makes parameter inference nearly impossible, it was observed that not only was the Fisher information horribly ill-conditioned, but that its the eigenvalues had a peculiar structure. In particular, the eigenvalues were nearly evenly spaced on a log scale[18], implying that not only were some parameter combinations unconstrained, but that parameter combinations were constrained in a hierarchical manner, with each combination allowed to fluctuate by roughly a constant factor more than the previous.

The peculiar eigenvalue structure of the model in reference [18] was later shown to be shared by many other systems biology models[48]. The general insensitivity of model behavior to specific parameter values has interesting consequences for biology, such as robustness and evolvability[30]; however, the properties of sloppiness are not unique to biological problems. Sloppiness is also observed in models of particle accelerators, interatomic potentials, insect flight, variational quantum wave functions, scaling functions of critical phenomenon, and artificial neural networks.

The ubiquity of the phenomenon of sloppiness is puzzling. On the one hand, the observation of sloppiness in so many diverse contexts suggests some sort of universality. Indeed, the Fisher information matrices of a class of sloppy models can be shown to belong to a universal ensemble. This matrix class is characterized by the Vandermonde matrix which is responsible for the peculiar sloppy spectrum[99]. Why a Vandermonde matrix would arise in each context is still puzzling. There are no shortages of possible explanations for sloppiness: over parameterizations, insufficient data, permutation invariance, reaction saturation, etc., but no single explanation is sufficiently broad to explain all the observed instances. Perhaps sloppiness has many causes that each result in the same eigenvalue structure.

On the other hand, the eigenvalues of the Fisher information are dependent on how the model has been parameterized, suggesting that sloppiness cannot be anything more than bad parameterization. In each case, one could argue, that if the modeler had been sufficiently clever to choose a more natural parameterization (from a phenomenological point of view) that sloppiness would never have been observed. In this regard, the universality of sloppiness appears to be a consequence of the psychological preference for humans to choose parameter ill-suited to describing model behavior, rather than any mathematical reason.

The work presented in chapter 2 seeks to resolve this dilemma by approaching the problem from an information geometric viewpoint. In this approach, models are associated with geometric manifolds with parameters as coordinates. We then search for geometric properties of sloppiness, independent of the parameterization. If any universal properties exist for sloppy model manifolds then we can understand that sloppiness implies something about the model itself and not just its parameterization. Furthermore, the hope is that such an investigation may reveal some connection among the various disparate models from which sloppiness emerges. Are variation quantum wave functions statistically similar to systems biology models for any deep reason, or merely by coincidence?

We find that several geometric properties exist common to many sloppy models. Specifically, sloppy models are bounded with a hierarchy of widths. This hierarchy of widths is closely related to the hierarchy of eigenvalues of the Fisher information: large eigenvalues correspond to long directions on the manifold. Furthermore, there is a hierarchy of curvatures and parameter-effects curvatures as well. These hierarchies can each be explained by a simple picture that unites sloppy models from the disparate fields. We view models as a type of generalized interpolation scheme. Whenever there are more parameters than effective degrees of freedom in the model, then the hierarchy of model widths and curvatures will arise. The current picture is that whether the model is describing a particle accelerator or insect flight, the mechanics of the modeling is the same: a function is constructed to interpolate between observed observations. One usually understand that a model is useful because of its ability to make predictions for new experiments. However, a model is not always able to predict an outcome with a desired uncertainty. By understanding models as a generalized interpolation scheme, we can understand that uncertainties in predictions become small when a model is interpolating in a high-dimensional space, but that uncertainties become large when the model is required to extrapolate.

1.2 Least squares fitting algorithms

In addition to providing insight about the nature of nonlinear modeling, the information geometry picture provides motivation into the practical difficulties associated with finding fits. Most fitting algorithms suffer from parameter evaporation, in which parameters are quickly pushed to extreme values, causing the algorithm to get lost on plateaus in parameter space. This phenomenon is caused by the narrow widths of the model manifold, and parameter evaporation can be understood geometrically as the algorithm running into the model boundary. Furthermore, even when the algorithms avoid the boundaries, they become sluggish as they must follow the narrow canyon to the best fit.

The most common algorithm for least squares data fitting is the Levenberg-

Marquardt algorithm, which is in turn based upon the Gauss-Newton algorithm. The Gauss-Newton method is an iterative quasi-Newton algorithm based on a local linearization of the residuals. The Gauss-Newton method suffers terribly from parameter evaporation, although it converges very quickly if the starting point is sufficiently near the best fit. In chapter 2 we present a new interpretatoin of the Gauss-Newton method in terms of geodesic flow on the model manifold. The Levenberg-Marquardt damps the Gauss-Newton step in such a way to effectively interpolate between the Gauss-Newton step and gradient descent. By an appropriately chosen damping strategy, the Levenberg-Marquardt is able to more effectively avoid parameter evaporation while retaining the fast convergence of the Gauss-Newton method. The Levenberg-Marquardt algorithm also has a geometric interpretation in terms of geodesic flow on the model graph.

Although the Levenberg-Marquardt method is far superior to the Gauss-Newton method, it still struggles to find good fits, particularly on large, sloppy problems. It can also become very sluggish if it must navigate a narrow canyon to find a best fit. To remedy these problems we discuss several geometrically motivated improvements to the Levenberg-Marquardt algorithm in chapter 3. Most notably, we introduce a geodesic acceleration correction to the usual Levenberg-Marquardt step. The geodesic acceleration proves very useful for improving both the quality of fits found and the speed of the algorithm for converging. It is also noteworthy that geodesic acceleration is relatively inexpensive to compute, making it very competitive as a replacement for the Levenberg-Marquardt algorithm.

1.3 Introduction to superconducting cavities

In the construction of particle accelerators, resonance cavities are constructed in which a resonating electric field is set up to accelerate particles. In order to reduce resistive losses, the cavities are made of superconducting material. Although the DC resistivity of a superconductor is zero, the AC resistivity relevant for the resonance cavities is nonzero, although it is still very small. Because of this very small resistance, the cavities have very high quality factors, Q, typically larger than 10^{10}. Superconductivity is, therefore, a critical component of the cavities.

One of the limiting factors that determines the maximum accelerating field is the response of the superconductor to the induced magnetic field. One of the hallmark features of superconductivity is the Meissner effect, in which a bulk superconducting material expels an externally applied magnetic field. If the applied magnetic field is sufficiently large, however, superconductor quenches and magnetic flux penetrates the material. When this occurs in the superconducting cavity, the cavity becomes ineffective. The work in chapters 4 and 5 seeks to provide a reliable estimate of the magnetic field strength at which this transition occurs.

The nature of the transition when flux first penetrates the superconductor is theoretically interesting since it need not occur at the point where the normal metal or flux-lattice states are energetically favorable. The superconducting Meissner state, it turns out, is metastable to the penetration of flux up to a critical magnetic field known as the superheating field H_{sh}. Solving for the threshold at which the barrier to flux penetration vanishes requires a calculation of not only the configuration of the superconducting state, but additionally the second variation of the free energy to explore the state's stability to various perturbations. The calculation is further complicated by the fact that for type-II superconductors, the leading unstable modes break the translational invariance of the superconducting state.

In chapter 4, we calculate the superheating field within Ginzburg-Landau theory. There we show that the two-dimensional fluctuations to which the superconducting state is unstable can be decomposed into Fourier modes which decouple from one another. We then map the stability problem onto an eigenvalue problem of a linear differential operator. Using this technique we are able to find the superheating field by solving only ordinary differential equations (no partial differential equations are necessary).

The results of 4 are not particularly relevant to the accelerator community because the predictions made by Ginzburg-Landau theory are quantitatively accurate only near the critical temperature, far from the temperature at which cavities are actually operated. We generalize the calculation to the semi-classical theory of Eilenberger in chapter 5. The relative complexity of the of the calculation in Eilenberger theory requires the use of more sophisticated numerical techniques, in particular we use a Galerkin method to map the stability analysis to a matrix eigenvalue problem. However, the basic approach is qualitatively the same as for the Ginzburg-Landau case. The results presented in chapter 5 are not converged and require additional computer resources to give accurate results.

CHAPTER 2

GEOMETRY OF NONLINEAR LEAST SQUARES WITH APPLICATIONS TO SLOPPY MODELS AND OPTIMIZATION

2.1 Abstract[1]

Parameter estimation by nonlinear least squares minimization is a common problem that has an elegant geometric interpretation: the possible parameter values of a model induce a manifold within the space of data predictions. The minimization problem is then to find the point on the manifold closest to the experimental data. We show that the model manifolds of a large class of models, known as *sloppy models*, have many universal features; they are characterized by a geometric series of widths, extrinsic curvatures, and parameter-effects curvatures, which we describe as a hyper-ribbon. A number of common difficulties in optimizing least squares problems are due to this common geometric structure. First, algorithms tend to run into the boundaries of the model manifold, causing parameters to diverge or become unphysical before they have been optimized. We introduce the model graph as an extension of the model manifold to remedy this problem. We argue that appropriate priors can remove the boundaries and further improve the convergence rates. We show that typical fits will have many evaporated parameters unless the data are very accurately known. Second, 'bare' model parameters are usually ill-suited to describing model behavior; cost contours in parameter space tend to form hierarchies of plateaus and long narrow canyons. Geometrically, we understand this inconvenient parameterization as an extremely skewed coordinate basis and show that it induces a large parameter-effects curvature on

[1]This chapter has been published in *Physical Review E* with couathors Benjamin B. Machta and James P. Sethna.

the manifold. By constructing alternative coordinates based on geodesic motion, we show that these long narrow canyons are transformed in many cases into a single quadratic, isotropic basin. We interpret the modified Gauss-Newton and Levenberg-Marquardt fitting algorithms as an Euler approximation to geodesic motion in these natural coordinates on the model manifold and the model graph respectively. By adding a geodesic acceleration adjustment to these algorithms, we alleviate the difficulties from parameter-effects curvature, improving both efficiency and success rates at finding good fits.

2.2 Introduction

An ubiquitous problem in mathematical modeling involves estimating parameter values from observational data. One of the most common approaches to the problem is to minimize a sum of squares of the deviations of predictions from observations. A typical problem may be stated as follows: given a regressor variable, t, sampled at a set of points $\{t_m\}$ with observed behavior $\{y_m\}$ and uncertainty $\{\sigma_m\}$, what values of the parameters, θ, in some model $f(t, \theta)$, best reproduce or explain the observed behavior? This optimal value of the parameters is known as the best fit.

To quantify how good a fit is, the standard approach is to assume that the data can be reproduced from the model plus a stochastic term that accounts for any discrepancies. That is to say

$$y_m = f(t_m, \theta) + \zeta_m,$$

where ζ_m are random variables assumed to be independently distributed according to $N(0, \sigma_m)$. Written another way, the residuals given by

$$r_m(\theta) = \frac{y_m - f(t_m, \theta)}{\sigma_m}, \tag{2.1}$$

are random variables that are independently, normally distributed with zero mean and unit variance. The probability distribution function of the residuals is then

$$P(\vec{r}, \theta) = \frac{1}{(2\pi)^{M/2}} \exp\left(-\frac{1}{2} \sum_{m=1}^{M} r_m(\theta)^2 \right), \tag{2.2}$$

where M is the number of residuals. The stochastic part of the residuals is assumed to enter through its dependence on the observed data, while the parameter dependence enters through the model. This distinction implies that while the residuals are random variables, the matrix of derivatives of the residuals with respect to the parameters is not. We represent this Jacobian matrix by $J_{m\mu}$:

$$J_{m\mu} = \partial_\mu r_m.$$

In this paper, we employ the convention that Greek letters index parameters, while Latin letters index data points, model points, and residuals.

For a given set of observations $\{y_m\}$, the distribution in Eq. (2.2) is a likelihood function, with the most likely, or best fit, parameters being those that minimize the cost function, C, defined by

$$C(\theta) = \frac{1}{2} \sum_m r_m(\theta)^2, \tag{2.3}$$

which is a sum of squares. Therefore, if the noise is Gaussian (normally) distributed, minimizing a sum of squares is equivalent to a maximum likelihood estimation.

If the model happens to be linear in the parameters it is a linear least squares problem and the best fit values of the parameters can be expressed analytically in terms of the observed data and the Jacobian. If, however, the model is nonlinear, the best fit cannot be found so easily. In fact, finding the best fit of a nonlinear problem can be a very difficult task, notwithstanding the many algorithms that are designed for this specific purpose.

For example, a nonlinear least squares problem may have many local minima. Any search algorithm that is purely local will at best converge to a local minima and fail to find the global best fit. The natural solution is to employ a search method designed to find a global minima, such as a genetic algorithm or simulated annealing. We will not address such topics in this paper, although the geometric framework that we develop could be applied to such methods. We find, surprisingly, that most fitting problems do not have many local minima. Instead, we find a universality of cost landscapes, as we discuss later in section 2.4, consisting of only one, or perhaps very few, minima.

Instead of difficulties from local minima, the best fit of a nonlinear least squares problem is difficult to find because of *sloppiness*, particularly if the model has many parameters. Sloppiness is the property that the behavior of the model responds very strongly to only a few combinations of parameters, known as stiff parameter combinations, and very weakly to all other combinations of parameters, which are known as sloppy parameter combinations. Although the sloppy model framework has been developed in the context of systems biology [19, 18, 22, 30, 47, 48, 46], models from many diverse fields have been shown to lie within the sloppy model universality class [99].

In this paper we present the geometric framework for studying nonlinear least squares models. This approach has a long, interesting history, originating with Jeffreys in 1939 [56], and later continued by Rao [84, 85] and many others [3, 75].

An equivalent, alternative formulation began with Beale in 1960 [13], and continued with the work of Bates and Watts [9, 10, 8, 12] and others [29, 28, 27]. The authors have used this geometric approach previously to explain the extreme difficulty of the data fitting process [94]; of which this work is a continuation.

In section 2.3 we present a review of the phenomenon of sloppiness and describes the *model manifold*, i.e. the geometric interpretation of a least squares model. The geometric picture naturally illustrates two major difficulties that arise when optimizing sloppy models. First, parameters tend to diverge or drift to unphysical values, geometrically corresponding to running off the edge of the manifold, as we describe in section 2.4. This is a consequence of the model manifold having boundaries that give it the shape of a curving hyper-ribbon in residual space with a geometric hierarchy of widths and curvatures. We show, in section 2.5 that the *model graph*, the surface formed by plotting the residual output versus the parameters, can help to remove the boundaries and improve the fitting process. Generalizing the model graph suggests the use of priors as additional residuals, as we do in section 2.6. We see there that the natural scales of the experiment can be a guide to adding priors to the cost function that can significantly improve the convergence rate.

The second difficulty is that the model's 'bare' parameters are often a poor coordinate choice for the manifold. In section 2.7 we construct new coordinates, which we call *extended geodesic coordinates*. The coordinates remove the effects of the bad coordinates all the way to the edge of the manifold. The degree to which extended geodesic coordinates are effective at facilitating optimization is related to the curvature of the manifold. Section 2.8 discusses several measures of curvature and explores curvature of sloppy models. We show that the *parameter-effects* curvature is typically the dominant curvature of a sloppy model, explaining why extended geodesic coordinates can be a huge simplification to the optimization process. We also show that typical best fits will usually have many evaporated parameters and then define a new measure of curvature, the *optimization curvature*, that is useful for understanding the limitation of iterative algorithms.

We apply geodesic motion to numerical algorithms in section 2.9, where we show that the modified Gauss-Newton method and Levenberg-Marquardt method are an Euler approximation to geodesic motion. We then add a geodesic acceleration correction to the Levenberg-Marquardt algorithm and achieve much faster convergence rates over standard algorithms and more reliability at finding good fits.

2.3 The Model Manifold

In this section we review the properties of sloppy models and the geometric picture naturally associated with least squares models. To provide a concrete example of sloppiness to which we can apply the geometric framework, consider the problem of fitting three monotonically decreasing data points to the model

$$y(t, \theta) = e^{-t\theta_1} + e^{-t\theta_2},$$

where $\theta_i > 0$. Although simple, this model illustrates many of the properties of more complicated models. Figure 2.1a is an illustration of the data and several progressively better fits. Because of the noise, the best fit does not pass exactly through all the data points, although the fit is within the errors.

A common tool to visualize the parameter dependence of the cost is to plot

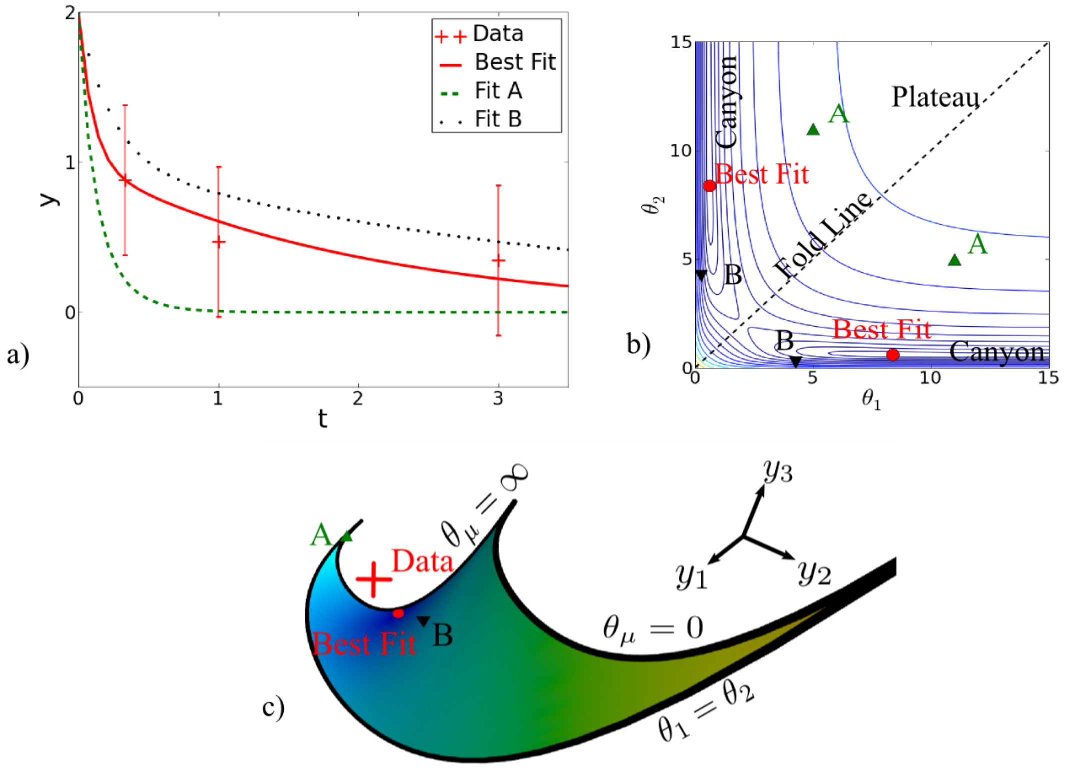


Figure 2.1: Fitting a nonlinear function to data

(Color online) (a) **Fitting a nonlinear function to data**, in this case the sum of two exponentials to three data points. Fit A has rate constants which decay too quickly, resulting in a poor fit; B is an improvement over Fit A, although the rates are too slow; the best fit minimizes the cost (the sum of the squares of the residuals, which are deviations of model from data points) (b) **Contours of constant Cost in parameter space.** Note the "plateau" in the region of large rates where the model is essentially independent of parameter changes. Note also the long, narrow canyon at lower rates, characteristic of a sloppy model. The sloppy direction is parallel to the canyon and the stiff direction is against the canyon wall. (c) **Model predictions in data space**. The experimental data is represented by a single point. The set of all possible fitting parameters induce a manifold of predictions in data space. The best fit is the point on the manifold nearest to the data. The plateau in (b) here is the small region around the short cusp near the corner. To help visualize the three dimensional structure, an animation of this manifold rotating in three dimensions in available in the online supplemental material [93] contours of constant cost in parameters space, as is done for our toy model in Figure 2.1b. This view illustrates many properties of sloppy models. This particular model is invariant to a permutation of the parameters, so the plot is symmetric for reflections about the $\theta_1 = \theta_2$ line. We refer to the $\theta_1 = \theta_2$ linear as the "fold line" for geometric reasons that will be apparent in section 2.5. Around the best fit, cost contours form a long narrow canyon. The direction along the length of the canyon is a sloppy direction, since this parameter combination hardly changes the behavior of the model, and the direction up a canyon wall is the stiff direction. Because this model has few parameters, the sloppiness is not as dramatic as it is for most sloppy models. It is not uncommon for real-life models to have canyons with an aspect ratios much more extreme than in Fig. 2.1b, typically $1000 : 1$ or more for models with 10 or more parameters [48].

Sloppiness can be quantified by considering the quadratic approximation of the cost around the best fit. The Hessian (second derivative) matrix, $H_{\mu\nu}$, of the cost at the best fit has eigenvalues that span many orders of magnitude and whose logarithms tend to be evenly spaced, as illustrated in Fig. 2.2. Eigenvectors of the Hessian with small eigenvalues are the sloppy directions, while those with large eigenvalues are the stiff directions. In terms of the residuals, the Hessian is given by

$$\begin{aligned} H_{\mu\nu} &= \partial_\mu \partial_\nu C \\ &= \sum_m \partial_\mu r_m \partial_\nu r_m + \sum_m r_m \partial_\mu \partial_\nu r_m && (2.4) \\ &\approx \sum_m \partial_\mu r_m \partial_\nu r_m. && (2.5) \\ &= \left(J^T J\right)_{\mu\nu} && (2.6) \end{aligned}$$

In the third and fourth line we have made the approximation that at the best fit the residuals are negligible. Although the best fit does not ordinarily corresponds to the residuals being exactly zero, the Hessian is usually dominated by the term in Eq. (2.5) when evaluated at the best fit. Furthermore, the dominant term, $J^T J$, is a quantity important geometrically which describes the model-parameter response for all values of the parameters independently of the data. The approximate Hessian is useful to study the sloppiness of a model independently of the data at points other than the best fit. It also shares the sloppy spectrum of the exact Hessian. We call the eigenvectors of $J^T J$ the local *eigenparameters* as they embody the varying stiff and sloppy combinations of the 'bare' parameters.

In addition to the stiff and sloppy parameter combinations near the best fit, Fig. 2.1b also illustrates another property common to sloppy models. Away from the best fit the cost function often depends less and less strongly on the parameters. The contour plot shows a large plateau where the model is insensitive to all parameter combinations. Because the plateau occupies a large region of parameter space, most initial guesses will lie on the plateau. When an initial parameter guess does begin on a plateau such as this, even finding the canyon can be a daunting task.

The process of finding the best fit of a sloppy model, usually consists of two steps. First, one explores the plateau to find the canyon. Second, one follows the canyon to the best fit. One will search to find a canyon and follow it, only to find a smaller plateau within the canyon that must then be searched to find another canyon. Qualitatively, the initial parameter guess does not fit the data, and the cost gradient does not help much to improve the fit. After adjusting the parameters, one finds a particular parameter combination that can be adjusted to

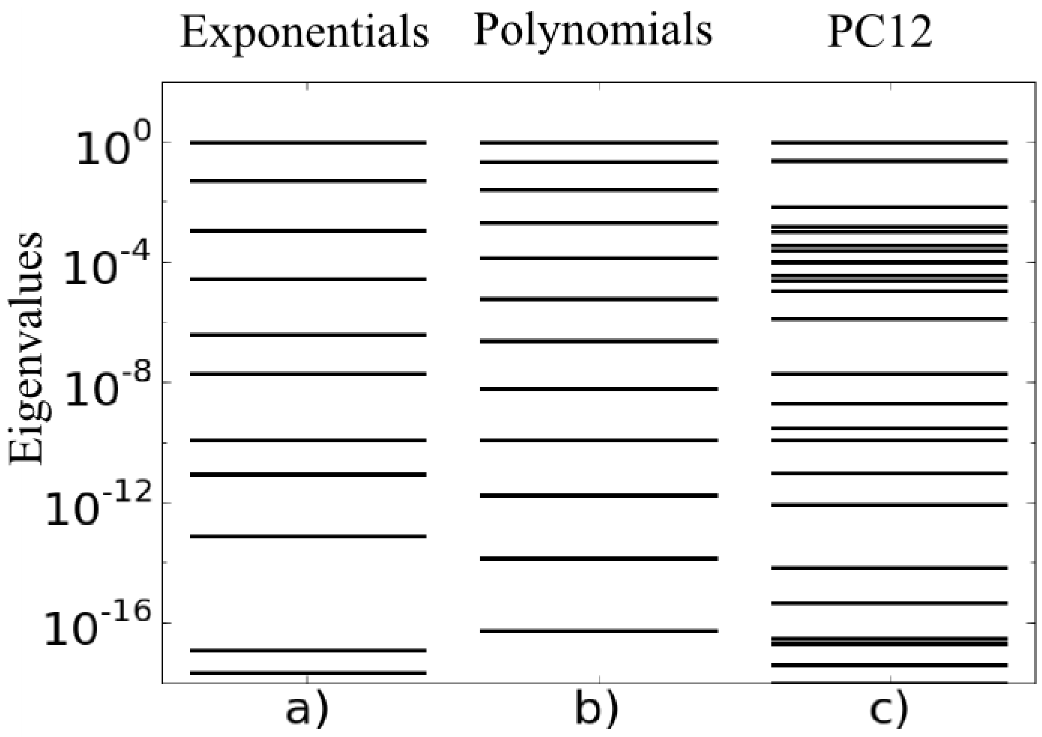


Figure 2.2: Sloppy Eigenvalues
Hessian eigenvalues for three sloppy models. Note the extraordinarily large range of eigenvalues (15-17 orders of magnitude, corresponding to to valley aspect ratios of 10^7-10^9) in Fig. 2.1b. Notice also the roughly equal fractional spacing between eigenvalues–there is no clean separation between important (stiff) and irrelevant (sloppy) direction in parameter space. a) The model formed by summing six exponential terms with rates and amplitudes. We use this model to investigate curvature in section 2.8 and as a test problem to compare algorithms in section 2.9.5. b) The linear problem of fitting polynomials is sloppy with the Hessian given by the Hilbert matrix. c) A more practical model from systems biology of signaling the epidermal growth factor in rat pheochromocytoma (PC12) cells [18], which also has a sloppy eigenvalue spectrum. Many more examples can be found in [48, 99].

fit some clump of the data. After optimizing this parameter combination (following the canyon), the fit has improved but is still not optimal. One must then search for another parameter combination that will fit another aspect of the data, i.e. find another canyon within the first. Neither of these steps, searching the plateau or following the canyon, is trivial.

Although plotting contours of constant cost in parameter space can be an useful and informative tool, it is not the only way to visualize the data. We now turn to describing an alternative geometric picture that helps to explain why the the processes of searching plateaus and following canyons can be so difficult. The geometric picture provides a natural motivation for tools to improve the optimization process.

Since the cost function has the special form of a sum of squares, it has the properties of a Euclidean distance. We can interpret the residuals as components of an M-dimensional residual vector. The M-dimensional space in which this vector lives is a Euclidean space which we refer to as *data space*. By considering Eq. (2.1), we see that the residual vector is the difference between a vector representing the data and vector representing the model (in units of the standard deviation). If the model depends on N parameters, with $N < M$, then by varying those N parameters, the model vector will sweep out an N-dimensional surface embedded within the M-dimensional Euclidean space. We call this surface the model manifold, it is sometimes also known as the expectation or regression surface [7, 12]. The model manifold of our toy model is shown in Fig. 2.1c. The problem of minimizing the cost is thus translated into the geometric problem of finding the point on the model manifold that is closest to the the data.

In transitioning from the parameter space picture to the model manifold picture, we are now faced with the problem of minimizing a function on a curved surface. Optimization on manifolds is a problem that has been given much attention in recent decades [41, 66, 67, 68, 82, 86, 87, 97, 101, 1]. The general problem of minimizing a function on a manifold is much more complicated than our problem; however, because the cost function is linked here to the structure of the manifold the problem at hand is much simpler.

The metric tensor measures distance on the manifold corresponding to infinitesimal changes in the parameters. It is induced from the Euclidean metric of the data space and is found by considering how small changes in parameters correspond to changes in the residuals. The two are related through the Jacobian matrix,

$$dr_m = \partial_\mu r_m d\theta^\mu = J_{m\mu} d\theta^\mu,$$

where repeated indices imply summation. The square of the distance moved in data space is then

$$dr^2 = (J^T J)_{\mu\nu} d\theta^\mu d\theta^\nu. \tag{2.7}$$

Eq. (2.7) is known as the first fundamental form, and the coefficient of the parameter infinitesimals is the metric tensor,

$$g_{\mu\nu} = (J^T J)_{\mu\nu} = \sum_m \partial_\mu r_m \partial_\nu r_m.$$

The metric tensor corresponds to the approximate Hessian matrix in Eq. (2.5); therefore, the metric is the Hessian of the cost at a point assuming that the point exactly reproduced the data.

Qualitatively, the difference between the metric tensor and the Jacobian matrix is that the former describes the local intrinsic properties of the manifold while the latter describes the local embedding. For nonlinear least squares fits, the embedding is crucial, since it is the embedding that defines the cost function. To understand how the manifold is locally embedded, consider a singular value decomposition of the Jacobian

$$J = U \Sigma V^T,$$

where V is an $N \times N$ unitary matrix satisfying $V^T V = 1$ and Σ is an $N \times N$ diagonal matrix of singular values. The matrix U is almost unitary, in the sense that it is an

$M \times N$ matrix satisfying $U^T U = 1$; however, UU^T is not the identity [83]. In other words, the columns of U contain N residual space vectors that are orthonormal spanning the range of J and not the whole embedding space. In terms of the singular value decomposition, the metric tensor is then given by

$$g = V \Sigma^2 V^T,$$

showing us that V is the matrix whose columns are the local eigenparameters of the metric with eigenvalues $\lambda_i = \Sigma_{ii}^2$.

The singular value decomposition tells us that the Jacobian maps metric eigenvectors onto the data space vector U_i and stretched by an amount $\sqrt{\lambda_i}$. We hence denote the columns of U the *eigenpredictions*. The product of singular values describes the mapping of local volume elements of parameter space to data space. A unit hyper-cube of parameter space is stretched along the eigenpredictions by the appropriate singular values to form a skewed, hyper-parallelepiped of volume $\sqrt{|g|}$.

The Jacobian and metric contain the first derivative information relating changes in parameters to changes in residuals or model behavior. The second derivative information is contained in the connection coefficient. The connection itself is a technical quantity describing how basis vectors on the tangent space move from point to point. The connection is also closely related to geodesic motion, introduced properly in section 2.7. Qualitatively it describes how the metric changes from point to point on the manifold. The relevant connection is the Riemann, or metric, connection; it is calculated from the metric by

$$\Gamma^{\alpha}_{\mu\nu} = \frac{1}{2} g^{\alpha\beta} (\partial_\mu g_{\beta\nu} + \partial_\nu g_{\beta\mu} - \partial_\beta g_{\mu\nu}),$$

or in terms of the residuals

$$\Gamma^{\alpha}_{\mu\nu} = g^{\alpha\beta} \sum_m \partial_\beta r_m \partial_\mu \partial_\nu r_m, \tag{2.8}$$

where $g^{\mu\nu} = (g^{-1})^{\mu\nu}$. One could now also calculate the Riemann curvature by application of the standard formulae; however, we postpone a discussion of curvature until section 2.8. For a more thorough discussion of concepts from differential geometry, we refer the reader to any text on the subject [71, 88, 36, 55].

We have calculated the metric tensor and the connection coefficients from the premise that the cost function, by its special functional form, has a natural interpretation as a Euclidean distance which induces a metric on the model manifold. Our approach is in the spirit of Bates and Watts' treatment of the subject [9, 10, 8, 12]. However, the intrinsic properties of the model manifold can be calculated in an alternative way without reference to the embedding through the methods of Jeffreys, Rao and others [56, 84, 85, 75, 3]. This approach is known as information geometry. We derive these quantities using information geometry in Appendix A.

Given a vector in data space we are often interested in decomposing it into two components; one lying within the tangent space of the model manifold at a point and one perpendicular to the tangent space. For this purpose, we introduce the projection operators P^T and P^N which act on data-space vectors and project into the tangent space and its compliment respectively. From the Jacobian at a point on the manifold, these operators are

$$P^T = \delta - P^N = J(g^{-1})J^T, \tag{2.9}$$

where δ is the identity operator. It is numerically more accurate to compute these operators using the singular value decomposition of the Jacobian:

$$P^T = UU^T.$$

Turning to the problem of optimization, the parameter space picture leads one initially to follow the naive, gradient descent direction, $-\nabla_\mu C$. An algorithm that moves in the gradient descent direction will decrease the cost most quickly for a given change in the parameters. If the cost contours form long narrow canyons, however, this direction is very inefficient; algorithms tend to zig-zag along the bottom of the canyon and only slowly approach the best fit [83].

In contrast, the model manifold defines an alternative direction which we call the Gauss-Newton direction, which decreases the cost most efficiently for a change in the behavior. If one imagines sitting on the surface of the manifold, looking at the point representing the data, then the Gauss-Newton direction in data space is the point directed toward the data but projected onto the manifold. Thus, if $\vec{v}$ is the Gauss-Newton direction in data space, it is given by

$$\begin{aligned}\vec{v} &= -P^T\vec{r} \\ &= -J(g^{-1})J^T\vec{r} \\ &= -J(g^{-1})\nabla C \\ &= -\vec{J}_\mu g^{\mu\nu}\nabla_\nu C, \qquad (2.10)\end{aligned}$$

where we have used the fact that $\nabla C = J^T r$. The components of the vector in parameter space, v^μ are related to the vector in data space through the Jacobian

$$\vec{v} = \vec{J}_\mu v^\mu; \qquad (2.11)$$

therefore, the direction in parameter space v^μ that decreases the cost most efficiently per unit change in behavior is

$$v^\mu = -g^{\mu\nu}\nabla_\nu C. \qquad (2.12)$$

The term 'Gauss-Newton' direction comes from the fact that it is the direction given by the Gauss-Newton algorithm described in section 2.9.1. Because the

Gauss-Newton direction multiplies the gradient by the inverse metric, it magnifies motion along the sloppy directions. This is the direction that will move the parameters along the canyon toward the best fit. The Gauss-Newton direction is purely geometric and will be the same in data space regardless of how the model is parametrized. The existence of the canyons are a consequence of bad parameterization on the manifold, which this parameter independent approach can help to remedy. Most sophisticated algorithms, such as conjugate gradient and Levenberg-Marquardt attempt to follow the Gauss-Newton direction as much as possible in order to not get stuck in the canyons.

The obvious connection between sloppiness and the model manifold is through the metric tensor. For sloppy models, the metric tensor of the model manifold (the approximate Hessian of Eq. (2.5)) has eigenvalues spread over many decades. This property is not intrinsic to the manifold however. In fact, one can always reparametrize the manifold to make the metric at a point any symmetric, positive definite matrix. This might naively suggest that sloppiness has no intrinsic geometric meaning, and that it is simply a result of a poor choice of parameters. The coordinate grid on the model manifold in data space is extremely skewed as in Figure 2.3. By reparametrizing, one can remove the skewedness and construct a more natural coordinate mesh. We will revisit this idea in section 2.7. We will argue in this manuscript that on the contrary, there is a geometrical component to sloppy nonlinear models that is independent of parameterization and in most cases that the human-picked 'bare' parameters naturally illuminate the sloppy intrinsic structure of the model manifold.

In the original parameterization, sections of parameter space are mapped onto very tiny volumes of data space. We remind the reader that a unit volume of parameter space is mapped into a volume of data space given by $\sqrt{|g|}$. Because many eigenvalues are nearly zero for sloppy models, the model manifold necessarily occupies a tiny sliver of data space. In fact, if a region of parameter space has larger eigenvalues by even a small factor, the cumulative effect on the product is that this region of parameter space will occupy most of the model manifold. We typically find that most of the model manifold is covered by a very small region of parameter space which corresponds to the volumes of (slightly) less skewed meshes.

We will see when we discuss curvature, that the large range of eigenvalues in the metric tensor usually correspond to a large anisotropy in the extrinsic curvature. Another geometric property of sloppy systems relates to the boundaries that the model imposes on the manifold. The existence of the boundaries for the toy model can be seen clearly in Fig. 2.1c. The surface drawn in the figure corresponds the patch of parameters within $0 \leq \theta_1, \theta_2 \leq \infty$. The three boundaries of the surface occur when the parameters reach their respective bounds. The one exception to this is the fold line, which corresponds to when the parameters are equal to one another. This anomalous boundary ($\theta_1 = \theta_2$) is called the fold line and is discussed further in section 2.5. Most nonlinear sloppy models have boundaries.

In the next section we will discuss how boundaries arise on the model manifold and why they pose problems for optimization algorithms. Then, in section 2.5 we describe another surface, the model graph, that removes the boundaries. The surface described by the model graph is equivalent to a model manifold with a linear Bayesian prior added as additional residuals. In section 2.6 we show that introducing other priors can be even more helpful for keeping algorithms away from the boundaries.

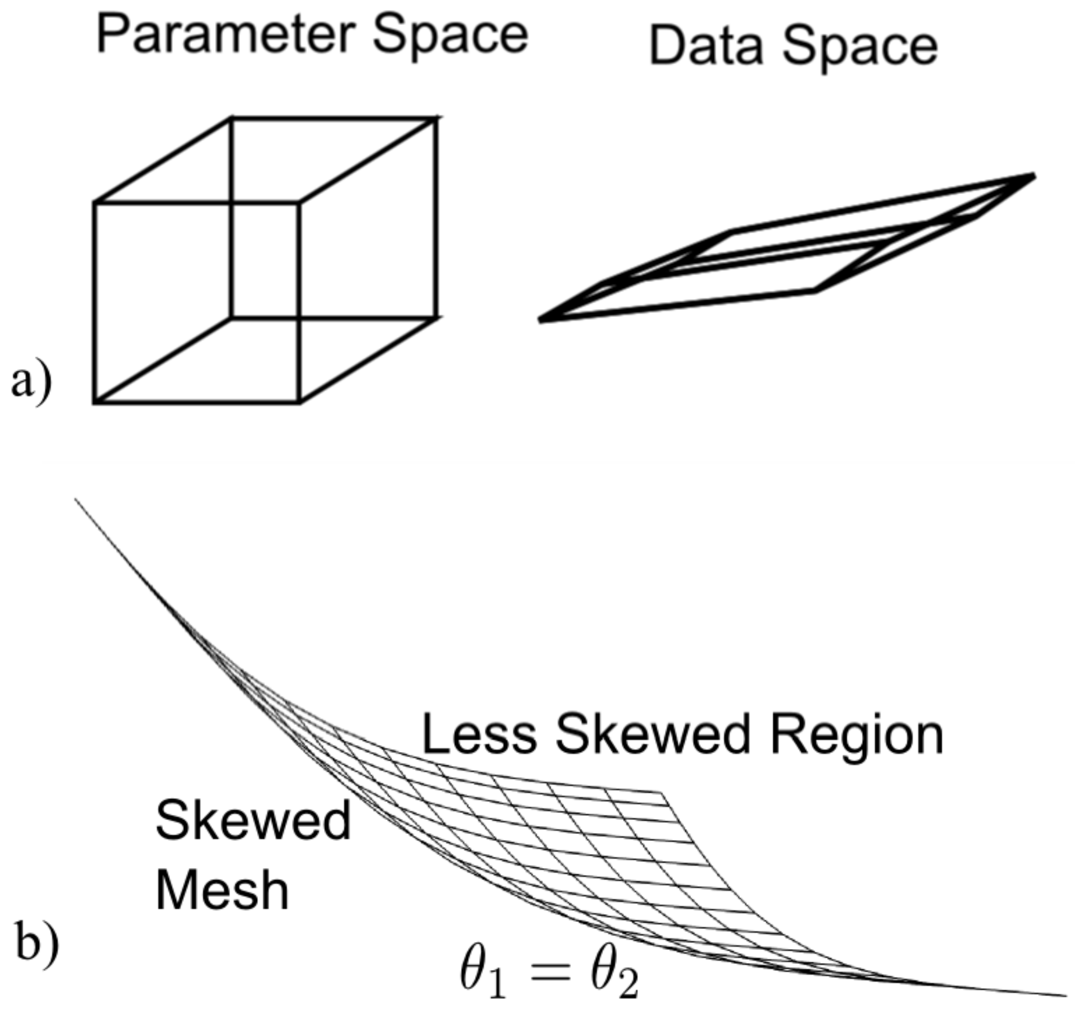


Figure 2.3: Skewed Coordinates

Skewed Coordinates. A sloppy model is characterized by a skewed coordinate mesh on the manifold. The volume of the parallel-piped is given by the determinant of the metric, which is equal to the product of the eigenvalues. Because sloppy models have many tiny eigenvalues, these volumes can be very small with extremely skewed coordinates. Our toy model has extremely skewed coordinates where the parameters are nearly equal (near the fold line). Most of the manifold is covered by regions where the coordinates are less skewed which corresponds to a very small region in parameter space.

2.4 Bounded Manifolds

Sloppiness is closely related to the existence of boundaries on the model manifold. This may seem to be a puzzling claim because sloppiness has previously been understood to be a statement relating to the *local* linearization of model space. Here we will extend this idea and see that it relates to the *global* structure of the manifold and how it produces difficulties for the optimization process.

To understand the origin of the boundaries on model manifolds, consider first the model of summing several exponentials

$$y(t, \theta) = \sum_{\mu} e^{-\theta_\mu t}.$$

We restrict ourselves to considering only positive arguments in the exponentials, which limits the range of behavior for each term to be between 0 and 1. This restriction already imposes boundaries on the model manifold, but those boundaries become much more narrow as we consider the range the model can produce by holding just a few time points fixed.

Fixing the output of the model at a few time points greatly reduces the values that the model can take on for all the remaining points. Fixing the values that the model takes on at a few data points is equivalent to considering a lower-dimensional cross section of the model manifold, as we have done in Fig. 2.4. The boundaries on this cross section are very narrow; the corresponding manifold is long and thin. Clearly, an algorithm that navigates the model manifold will quickly run into the boundaries of this model unless it is actively avoiding them.

In general, if a function is analytic, the results presented in Fig. 2.4 are fairly generic, they come from general theorems governing the interpolation of functions.

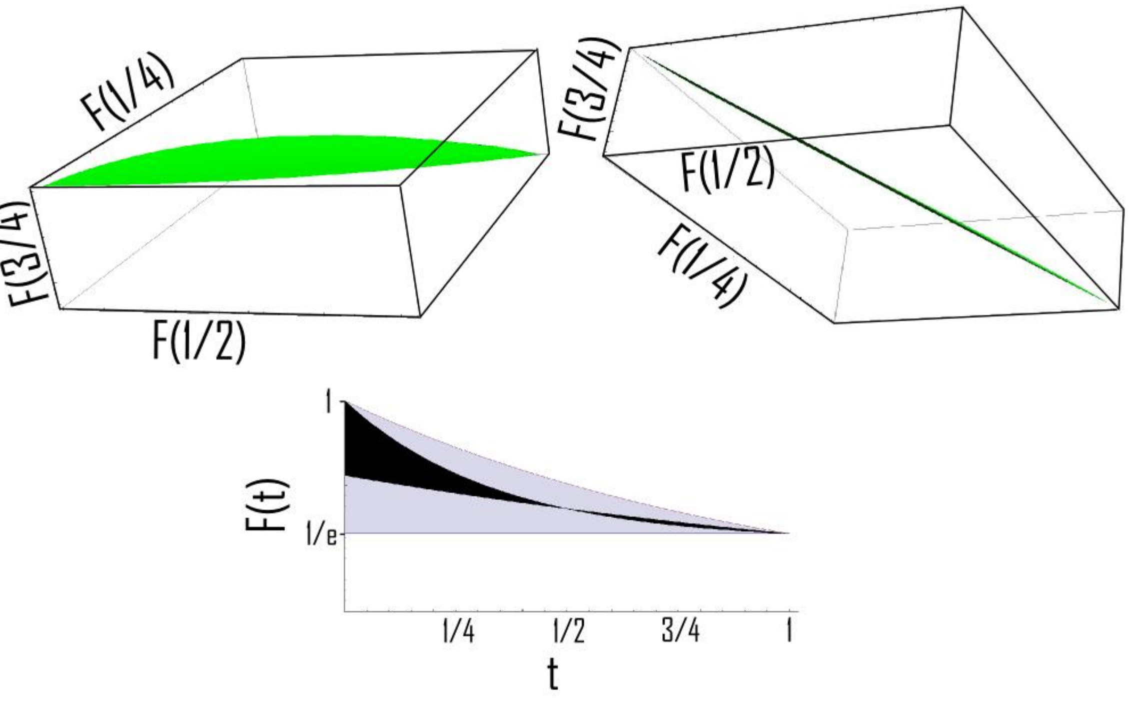


Figure 2.4: Hyper-ribbon
(Color online) Fixing a few data points greatly restricts the possible range of the model behavior between those data points (lower). This is a consequence of interpolation of analytic functions. In this case, $f(t)$ is a sum of three exponentials with six parameters (amplitudes and rates). Shown above is a three dimensional slice of possible models plotted in data space, with the value of $f(0)$ fixed to 1 and the value of $f(1)$ fixed to $1/e$. With these constraints we are left with a four dimensional surface, meaning that the manifold of possible data shown here is indeed a volume. However, from a carefully chosen perspective (upper right) this volume can be seen to be extremely thin–in fact most of its apparent width is curvature of the nearly two dimensional sheet, evidenced by being able to see both the top (green) and bottom (black) simultaneously. (An animation of points in this volume rotating in three dimensional space is available in the online supplemental material [93].) Generic aspects of this picture illustrate the difficulty of fitting nonlinear problems. Geodesics in this volume are just straight lines in three dimensions. Although the manifold seems to be only slightly curved, its extreme thinness means that geodesics travel very short distances before running into model boundaries, necessitating the diagonal cutoff in Levenberg-Marquardt algorithms as well as the priors discussed in section 2.6.

If a function is sampled at a sufficient number of time points to capture its major features, then the behavior of the function at times between the sampling can be predicted with good accuracy by an interpolating function. For polynomial fits, as considered here, a function, $f(t)$, sampled at n time points, $(t_1, t_2, ..., t_n)$, can be fit exactly by a unique polynomial of degree $n-1$, $P_{n-1}(t)$. Then at some interpolating point, t, the discrepancy in the interpolation and the function is given by

$$f(t) - P_{n-1}(t) = \frac{\omega(t) f^{(n)}(\xi)}{n!}, \tag{2.13}$$

where $f^{(n)}(t)$ is the n-th derivative of the function and ξ lies somewhere in the range $t_1 < \xi < t_n$ [89]. The polynomial $\omega(t)$ has roots at each of the interpolating points

$$\omega(t) = (t - t_1)(t - t_2)...(t - t_n).$$

By inspecting Eq. (2.13), it is clear that the discrepancy between the interpolation and the actual function will become vanishingly small if higher derivatives of the function do not grow too fast (which is the case for analytic functions) and if the sampling points are not too widely spaced (see Fig. 2.5).

The possible error of the interpolation function bounds the allowed range of behavior, δf_n, of the model at t_0 after constraining the nearby n data points, which corresponds to measuring cross sections of the manifold. Consider the ratio of successive cross sections,

$$\frac{\delta f_{n+1}}{\delta f_n} = (t - t_{n+1})(n+1) \frac{f^{n+1}(\xi)}{f^n(\xi')},$$

if n is sufficiently large, then

$$(n+1) \frac{f^{n+1}(\xi)}{f^n(\xi')} \approx \frac{1}{R};$$

therefore, we find that

$$\frac{\delta f_{n+1}}{\delta f_n} \approx \frac{t - t_{n+1}}{R} < 1$$

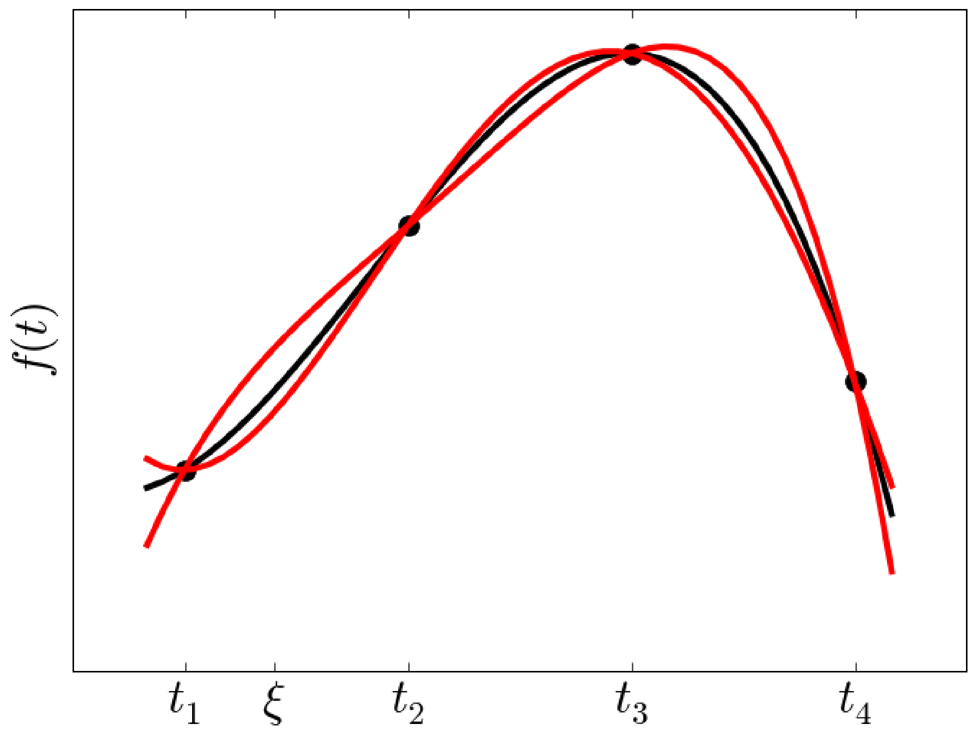


Figure 2.5: Interpolating function
(Color online) The possible values of a model at intermediate time points are restricted by interpolating theorems. Taking cross sections of the model manifold corresponds to fixing the model values at a few time points, restricting the possible values at the remaining times. Therefore, the model manifold will have a hierarchy of progressively thinner widths, much like a hyper-ribbon.

by the ratio test. Each cross section is thinner than the last by a roughly constant factor $\Delta = \delta t / R$, predicting a hierarchy of widths on the model manifold. We describe the shape of a model manifold with such a hierarchy as a hyper-ribbon. We will now measure these widths for a few sloppy models and see that the predicted hierarchy is in fact present.

As a first example, consider the sloppy model of fitting polynomials

$$f(t, \theta) = \sum_m \theta_m t^m. \tag{2.14}$$

If the parameters of the model are allowed to vary over all real values, then one can always fit M data points exactly with an $(M-1)^{th}$ degree polynomial. However, we wish to artificially restrict the range of the parameters to imitate the limited range of behavior characteristic of nonlinear models. A simple restriction is given by $\sum_m \theta_m^2 \leq 1$. This constraint enforces the condition that higher derivatives of the function become small (roughly that the radius of convergence is one) and corresponds to the unit hyper-sphere in parameter space. If this function is sampled at time points $(t_1, t_2, ..., t_n)$ then the model vector in data space can be written as

$$\vec{f} = \begin{pmatrix} 1 & t_1 & t_1^2 & \cdots \\ 1 & t_2 & t_2^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 & \cdots \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{pmatrix}. \tag{2.15}$$

The matrix multiplying the vector of parameters is an example of a Vandermonde matrix. The Vandermonde matrix is known to be sloppy and, in fact, plays an important role in the sloppy model universality class. The singular values of the Vandermonde matrix are what produce the sloppy eigenvalue spectrum of sloppy models. Reference [99] shows that these singular values are indeed broadly spaced in log. For this model, the Vandermonde matrix is exactly the Jacobian.

By limiting our parameter space to a hypersphere for the model in Eq. (2.14), the corresponding model manifold is limited to a hyper-ellipse in data space. The principal axes of this hyper-ellipse are the eigenpredictions directions we discussed in section 2.3. The lengths of the principal axes are the singular values. Consequently, there will be a hierarchy of progressively thinner boundaries on the model manifold due to the wide ranging singular values of the Vandermonde matrix. For this model, the purely local property of the metric tensor eigenvalue spectrum is intimately connected to the global property of the boundaries and shape of the model manifold.

As a second example, consider the model consisting of the sum of eight exponential terms, $y = \sum_\mu A_\mu e^{-\theta_\mu t}$. We use log-parameters, $r_{\theta\mu} = \log \theta_\mu$ and $r_{A\mu} = \log A_\mu$, to make parameters dimensionless and enforce positivity. We numerically calculate the several widths of the corresponding model manifold in Fig. 2.6a, where we see that they are accurately predicted by the singular values of the Jacobian. The widths in Fig. 2.6 were calculated by considering geodesic motion in each of the eigendirections of the metric from some point located near the center of the model manifold. We follow the geodesic motion until it reaches a boundary; the length in data space of the geodesic is the width. Alternatively, we can choose $M - N$ orthogonal unit vectors that span the space perpendicular to the tangent plane at a point and a single unit vector given by a eigenprediction of the Jacobian which lies within the tangent plane. The $M - N + 1$ dimensional hyper-plane spanned by these unit vectors intersects the model manifold along a one-dimensional curve. The width can be taken to be the length of that intersection. The widths given by these two methods are comparable.

We can show analytically that our exponential fitting problem has model manifold widths proportional to the corresponding singular values of the Jacobian in the limit of a continuous distribution of exponents, θ_μ, using an argument provided to us by Yoav Kallus. In this limit, the sum can be replaced by an integral,

$$y(t) = \int d\theta A(\theta) e^{-t\theta} = \mathcal{L}\left\{A(\theta)\right\},$$

where the model is now the Laplace transform of the amplitudes $A(\theta)$. In this limit the data can be fit without varying the exponential rates, leaving only the linear amplitudes as parameters. If we assume the data has been normalized according to $y(t = 0) \leq 1$, then it is natural to consider the hyper-tetrahedron of parameter space given by by $A_n > 0$ and $\sum A_n \leq 1$. In parameter space, this tetrahedron has a maximum aspect ratio of $\sqrt{2/M}$, but the mapping to data space distorts the tetrahedron by a constant Jacobian whose singular values we have seen to span many orders of magnitude. The resulting manifold thus must have a hierarchy of widths along the eigenpredictions equal to the corresponding eigenvalues within the relatively small factor $\sqrt{2/M}$.

As our third example, we consider a feed-forward artificial neural network [52]. For computational ease, we choose a small network consisting of a layer of four input neurons, a layer of four hidden neurons, and an output layer of two neurons. We use the hyperbolic tangent function as our sigmoid function and vary the connection weights as parameters. As this model is not known to reduce to a linear model in any limit, it serves as a test that the agreement for fitting exponentials is not special. Fig. 2.6b shows indeed that the singular values of the Jacobian agree with geodesic widths again for this model.

The results in Fig. 2.6 is one of our main results and requires some discussion. Strictly speaking, the singular values of the Jacobian have units of data space distance per unit parameter space distance, while the units of the widths are data space distance independent of parameters. In the case of the exponential model, we have used log-parameters, making the parameters dimensionless. In the neural network, the parameters are the connection weights whose natural scale is one. In general, the exact agreement between the singular values and the widths may not agree if the parameters utilize different units or have another natural scale. One must note, however, that the enormous range of singular values implies that the units would have to be radically different from natural values to lead to significant distortions.

Additionally, the two models presented in Fig. 2.6 are particularly easy to fit to data. The fact that from a centrally located point, geodesics can explore

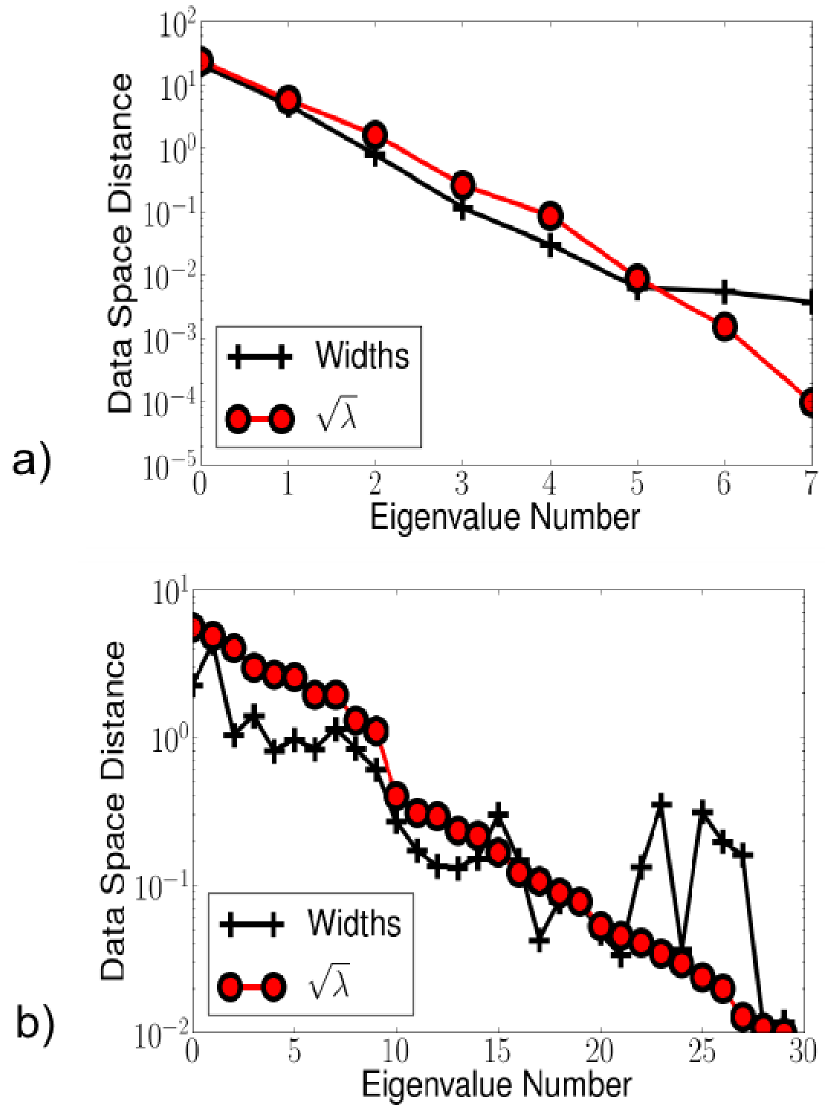


Figure 2.6: Cross-sectional widths
(Color online) a) Geodesic cross-sectional widths of an eight dimensional model manifold along the eigendirections of the metric from some central point, together with the square root of the eigenvalues (singular values of the Jacobian) [94]. Notice the hierarchy of these data-space distances – the widths and singular values each spanning around four orders of magnitude. To a good approximation, the cross-sectional widths are given by singular values. In the limit of infinitely many exponential terms, this model becomes linear. b) Geodesic cross-sectional widths of a feed-forward artificial neural network. Once again, the widths nicely track the singular values.

nearly the entire range of model behavior suggests that the boundaries are not a serious impediment to the optimization. For more difficult models, such as the PC12 model in systems biology [18], we find that the the widths estimated from the singular values and from geodesic motion disagree. The geodesic widths are much smaller than the singular value estimates. In this case, although the spacing between geodesic widths is the same as the spacing between the singular values, they are smaller by several orders of magnitude. We believe that most typical starting points of this model lie near a hyper-corner of the model manifold. If this is the case, then geodesics will be unable to explore the full range of model behavior without reaching a model boundary. We argue later in this section that this phenomenon is one of the main difficulties in optimization, and in fact, we find that the PC12 model is a much more difficult fitting problem than either the exponential or neural network problem.

We have seen that sloppiness is the result of skewed coordinates on the model manifold, and we will argue later in section 2.7 that algorithms are sluggish as a result of this poor parameterization. Fig. 2.6 tells us that the 'bare' model parameters are not as perverse as one might naively have thought. Although the bare-parameter directions are inconvenient for describing the model behavior, the local singular values and eigenpredictions of the Jacobian are useful estimates of the model's global shape. The fact that the local stiff and sloppy directions coincide with the global long and narrow directions is a nontrivial result that seems to hold for most models.

To complete our description of a typical sloppy model manifold requires a discussion of curvature, which we postpone until section 2.8.4. We will see that in addition to a hierarchy of boundaries, the manifold typically has a hierarchy of extrinsic and parameter-effects curvatures whose scales are set by the smallest and widest widths respectively.

We argue elsewhere [94], that the ubiquity of sloppy models, appearing everywhere from models in systems biology [48], insect flight [99], variational quantum wave functions, inter-atomic potentials [40], and a model of the next-generation international linear collider [46], implies that a large class of models have very narrow boundaries on their model manifolds. The interpretation that multiparameter fits are a type of high-dimensional analytic interpolation scheme, however, also explains why so many models are sloppy. Whenever there are more parameters than effective degrees of freedom among the data points, then there are necessarily directions in parameter space that have a limited effect on the model behavior, implying the metric must have small eigenvalues. Because successive parameter directions have a hierarchy of vanishing effect on model behavior, the metric must have a hierarchy of eigenvalues.

We view most multiparameter fits as a type of multi-dimensional interpolation. Only a few stiff parameter combinations need to be tuned in order to find a reasonable fit. The remaining sloppy degrees of freedom do not alter the fit much, because they fine tune the interpolated model behavior, which, as we have seen, is very restricted. This has important consequences for interpreting the best fit parameters. One should not expect the best fit parameters to necessarily represent the physical values of the parameters, as each parameter can be varied by many orders of magnitude along the sloppy directions. Although the parameter values at a best fit cannot typically be trusted, one can still make falsifiable predictions about model behavior without knowing the parameter values by considering an ensemble of parameters with reasonable fits [19, 18, 22, 47].

For our fitting exponential example, part of the model boundary was the 'fold lines' where pairs of the exponents are equal (see Fig. 2.1). No parameters were at extreme values, but the model behavior was nonetheless singular. Will such internal boundaries arise generically for large nonlinear models? Model boundaries correspond to points on the manifold where the metric is singular. Typical boundaries occur when parameters are near their extreme values (such as $\pm\infty$ or zero), where the model becomes unresponsive to changes in the parameters. Formally, a singularity will occur if the basis vectors on the model manifold given by $\vec{e}_\mu = \partial_\mu \vec{r}$ are linearly dependent, which is to say there exist a set of nonzero α^μ's for which

$$\alpha^\mu \vec{e}_\mu = 0. \tag{2.16}$$

In order to satisfy Eq. (2.16) we may vary $2N$ parameters (the N values of α^μ plus the N parameters of the model) to satisfy M equations. Therefore if $M < 2N$ there will exist nontrivial singular points of the metric at non-extreme values of the parameters.

For models with $M > 2N$, we do not expect Eq. (2.16) to be exactly satisfied generically except at extreme values of the parameters when one or more of the basis vectors vanish, $\vec{e}_\mu = 0$. However, many of the data points are interpolating points as we have argued above, and we expect qualitatively to be able to ignore several data points without much information loss. In general, we expect that Eq. (2.16) could be satisfied to machine precision at nontrivial values of the parameters even for relatively small N.

Now that we understand the origin of boundaries on the model manifold, we can discuss why they are problematic for the process of optimization. It has been observed in the context of training neural networks, that metric singularities (i.e. model boundaries) can have a strong influence on the fitting [2]. More generally, the process of fitting a sloppy model to data involves the frustrating experience of applying a black box algorithm to the problem which appears to be converging, but then returns a set of parameters that does not fit the data well and includes parameter values that are far from any reasonable value. We refer to this drift of the parameters to extreme values as parameter evaporation [2]. This phenomenon is troublesome not just because it causes the algorithm to fail. Often, models are more computationally expensive to evaluate when they are near the extreme values of their parameters. Algorithms will often not just fail to converge, but they will take a long time in the process.

After an algorithm has failed and parameters have evaporated, one may resort to adjusting the parameter values by hand and then reapplying the algorithm. Hopefully, iterating this process will lead to a good fit. Even if one eventually succeeds in finding a good fit, because of the necessity of adjusting parameters by hand, it can be a long and boring process.

Parameter evaporation is a direct consequence of the boundaries of the model manifold. To understand this, recall from section 2.3 that the model manifold defines a natural direction, the Gauss-Newton direction, that most algorithms try to follow. The problem with blindly following the Gauss-Newton direction is that it is purely local and ignores the fact that sloppy models have boundaries. Consider our example model; the model manifold has boundaries when the rates become infinite. If an initial guess has over-estimated or under-estimated the parameters, the Gauss-Newton direction can point toward the boundary of the manifold, as does fit A in Fig. 2.7. If one considers the parameter space picture, the Gauss-

[2]The term parameter evaporation was originally used to describe the drift of parameters to infinite values in the process of Monte Carlo sampling [17]. In this case the tendency of parameters to run to unphysical values is a literal evaporation caused by the finite temperature of the stochastic process. We now use the term to also describe deterministic drifts in parameters to extreme values in the optimization process.

Newton direction is clearly nonsensical, pointing away from the best fit. Generally, while on a plateau region, the gradient direction is better at avoiding the manifold boundaries. However, nearer the best fit, the boundary is less important and the Gauss-Newton direction is much more efficient than the downhill direction, as is the case for fit B in Fig. 2.7.

Since the model manifold typically has several narrow widths, it is reasonable to expect that a fit to noisy data will evaporate many parameters to their limiting values (such as ∞ or zero), as we explore in section 2.8.7. We therefore do not want to prevent the algorithm from evaporating parameters altogether. Instead, we want to prevent the algorithm from prematurely evaporating parameters and becoming stuck on the boundary (or lost on the plateau). Using the two natural directions to avoid the manifold boundaries while navigating canyons to the best fit is at the heart of the difficulty in optimizing sloppy models. Fortunately, there exists a natural interpolation between the two pictures which we call the model graph and is the subject of the next section. This natural interpolation is exploited by the Levenberg-Marquardt algorithm, which we discuss in section 2.9.

2.5 The Model Graph

We saw in Section 2.4 that the geometry of sloppiness explains the phenomenon of parameter evaporation as algorithms push parameters toward the boundary of the manifold. However, as we mentioned in Section 2.3, the model manifold picture is a view complementary to the parameter space picture, as illustrated in Fig. 2.1.

The parameter space picture has the advantage that boundaries typically do not exist (i.e. they lie at parameter values equal to ∞). If model boundaries

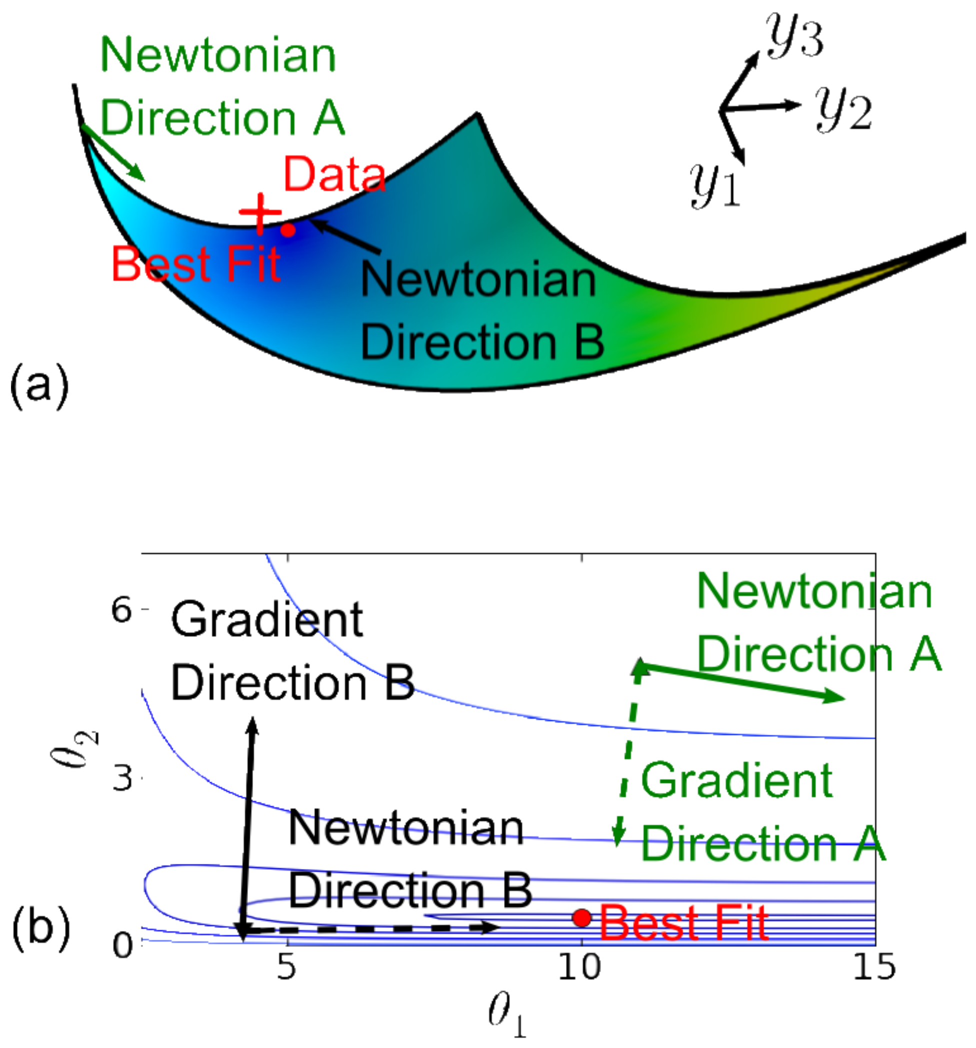


Figure 2.7: Gradient and Newtonian directions in data and parameter space
a) (Color online) **Falling off the edge of the model manifold.** The manifold in data space defines a "natural" direction, known as the Gauss-Newton direction, in which an algorithm will try to follow to the best fit. Often this direction will push parameters toward the edge of the manifold. b) **Gradient and Gauss-Newton directions in Parameter space**. The manifold edge corresponds to infinite values of the parameters. Following the Gauss-Newton direction to the edge of the manifold will cause parameters to evaporate while on the plateau. While in a canyon, however, the Gauss-Newton direction gives the most efficient direction to the best fit.

occur for parameter values that are not infinite, but are otherwise unphysical, for example, $\theta = 0$ for our toy model, it is helpful to change parameters in such a way as to map these boundaries to infinity. For the case of summing exponentials, it is typical to work in $\log \theta$, which puts all boundaries at infinite parameter values and has the added bonus of being dimensionless (avoiding problems of choice of units). In addition to removing boundaries, the parameter space does not have the complications from curvature; it is a flat, Euclidean space.

The disadvantage of the parameter space picture is that motion in parameter space is extremely disconnected from the behavior of the model. This problem arises as an algorithm searches the plateau looking for the canyon and again when it follows the winding canyon toward the best fit.

The model manifold picture and the parameter space picture can be combined to utilize the strengths of both approaches. This combination is called the model graph because it is the surface created by the graph of the model, i.e. the behavior plotted against the parameters. The model graph is an N dimensional surface embedded in an $M + N$ dimensional Euclidean space. The embedding space is formed by combining the M dimensions of data space with the N dimensions of parameter space. The metric for the model graph can be seen to be

$$g_{\mu\nu} = g^0_{\mu\nu} + \lambda D_{\mu\nu}, \tag{2.17}$$

where $g^0_{\mu\nu} = \left(J^T T\right)_{\mu\nu}$ is the metric of the model manifold and $D_{\mu\nu}$ is the metric of parameters space. We discuss common parameter space metrics below. We have introduced the free parameter λ in Eq. (2.17) which gives the relative weight of the parameter space metric to the data space metric. Most of the work in optimizing an algorithm comes from a suitable choice of λ, known as the damping parameter or the Levenberg-Marquardt parameter.

If $D_{\mu\nu}$ is the identity, then we call the metric in Eq. (2.17) the Levenberg metric because of its role in the Levenberg algorithm [64]. Another possible choice for $D_{\mu\nu}$ is to populate its diagonal with the diagonal elements of $g^0_{\mu\nu}$ while leaving the off-diagonal elements zero. This choice appears in the Levenberg-Marquardt algorithm [69] and has the advantage that the resulting method is invariant to rescaling the parameters, e.g. it is independent of units. It has the problem, however, that if a parameter evaporates then its corresponding diagonal element may vanish and the model graph metric becomes singular. To avoid this dilemma, one often chooses D to have diagonal elements given by the largest diagonal element of g^0 yet encountered by the algorithm [72]. This method is scale invariant but guarantees that D is always positive definite. We discuss these algorithms further in section 2.9.

It is our experience that the Marquardt metric is much less useful than the Levenberg metric for preventing parameter evaporation. While it may seem counter-intuitive to have a metric (and by extension an algorithm) that is sensitive to whether the parameters are measured in inches or miles, we stress that the purpose of the model graph is to *introduce* parameter dependence to the manifold. Presumably, the modeler is measuring parameters in inches because inches are a more natural unit for the model. By disregarding that information, the Marquardt metric is losing a valuable sense of scale for the parameters and is more sensitive to parameter evaporation. The concept of the natural units will be important in the discussion of priors in section 2.6. On the other hand, the Marquardt method is faster at following a narrow canyon and the best choice likely depends on the particular problem.

If the choice of metric for the parameter space is constant, $\partial_\alpha D_{\mu\nu} = 0$, then the connection coefficients of the model graph (with all lowered indices) are the same as for the model manifold given in Eq. (2.8). The connection with a raised index will include dependence on the parameter space metric:

$$\Gamma^{\mu}_{\alpha\beta} = (g^{-1})^{\mu\nu} \sum_m \partial_\nu r_m \partial_\alpha \partial_\beta r_m,$$

where g is given by Eq. (2.17).

By considering the model graph instead of the model manifold, we can remove the problems associated with the model boundaries. We return to our example problem to illustrate this point. The embedding space for the model graph is $3 + 2 = 5$ dimensional, so we are restricted to viewing 3 dimensional projections of the embedding space. In Fig. 2.8 we illustrate the model graph (Levenberg metric) for $\lambda = 0$, which is simply the model manifold, and for $\lambda \neq 0$, which shows that boundaries of the model manifold are removed in the graph. Since the boundaries occur at $\theta = \infty$, they are infinity far from the origin on the model graph. Even the boundary corresponding to the fold line has been removed, as the fold has opened up like a folded sheet of paper. Since generic boundaries correspond to singular points of the metric, the model graph has no such boundaries as its metric is positive definite for any $\lambda > 0$.

After removing the boundaries associated with the model manifold, the next advantage of the model graph is to provide a means of seamlessly interpolating between the natural directions of both data space and parameter space. The damping term, λ, appearing in Eq. (2.17) is well suited for this interpolation in sloppy models. If we consider the Levenberg metric, the eigenvectors of the model manifold metric, g^0, are unchanged by adding a multiple of the identity. However, the corresponding eigenvalues are shifted by the λ parameter. It is the sloppy eigenvalues that are dangerous to the Gauss-Newton direction. Since the eigenvalues of a

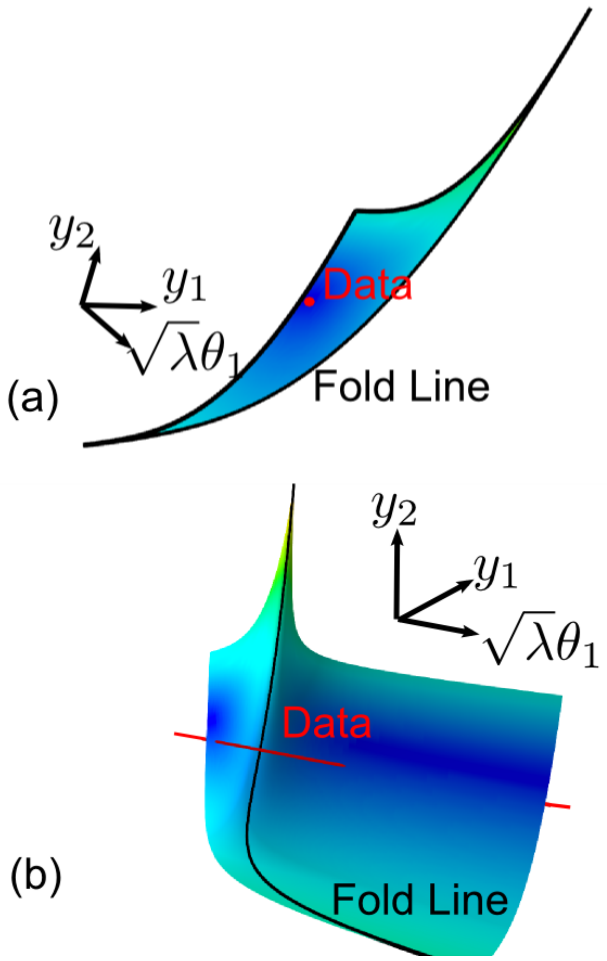


Figure 2.8: The Model Graph
(Color online) The effect of the damping parameter is to produce a new metric for the surface induced by the graph of the model versus the input parameters. (a) **Model Graph, $\lambda = 0$.** If the parameter is zero, then the resulting graph is simply the original model manifold, with no extent in the parameter directions. Here we see a flat two dimensional cross section; the z-axis is a parameter value multiplied by $\sqrt{\lambda} = 0$. (b) **Model Graph $\lambda \neq 0$.** If the parameter is increased, the surface is "stretched" into a higher dimensional embedding space. This is an effective technique for removing the boundaries, as no such boundary exists in the model graph. However, this comes at a cost of removing the geometric connection between the cost function and the structure of the surface. For very large damping parameters, the model graph metric becomes a multiple of the parameter space metric, which rotates the Gauss-Newton direction into the gradient direction. The damping term therefore interpolates between the parameter space metric and the data space metric. A three-dimensional animation of this figure is available in the online supplemental material [93].

sloppy model span many orders of magnitude, this means that all the eigenvalues that were originally less than λ are cutoff at λ in the model graph metric, and the larger eigenvalues are virtually unaffected. By adjusting the damping term, we can essentially wash out the effects of the sloppy directions and preserve the Gauss-Newton direction from the model manifold in the stiff directions. Since the eigenvalues span many orders of magnitude, the parameter does not need to be finely tuned; it can be adjusted very roughly and an algorithm will still converge, as we will see in section 2.9. We demonstrate how λ can interpolate between the two natural directions for our example model in Fig. 2.9.

2.6 Priors

In Bayesian statistics, a *prior* is an a-priori probability distribution in parameter space, giving information about the relative probability densities for the model as parameters are varied. For example, if one has pre-existing measurements of the parameters $\theta_m = \theta_m^0 \pm \sigma_m$ with normally distributed uncertainties, then the probability density would be $\prod_m 1/\sqrt{2\pi\sigma_m^2} \exp\left[-(\theta_m - \theta_m^0)^2/(2\sigma_m^2\right]$ before fitting to the current data. This corresponds to a negative-log-likelihood cost that (apart from an overall constant) is the sum of squares, which can be nicely interpreted as the effects of an additional set of "prior residuals"

$$r_m = (\theta_m - \theta_m^0)/\sigma_m \tag{2.18}$$

(interpreting the pre-existing measurements as extra data points). In this section, we will explore the more general use of such extra terms, not to incorporate information about parameter values, but rather to incorporate information about the ranges of parameters expected to be useful in generating good fits.

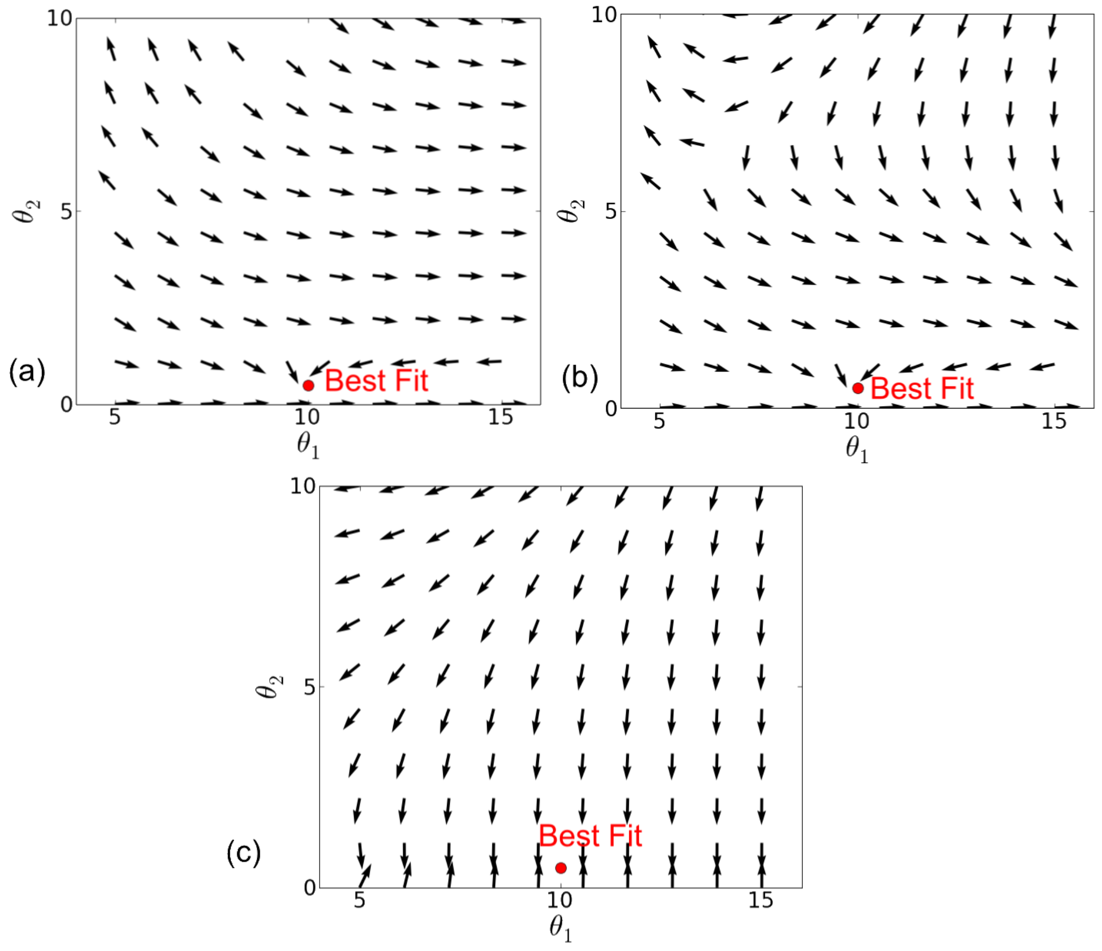


Figure 2.9: Gradient and Newtonian directions in parameter space
(Color online) (A)**Gauss-Newton Directions.** The Gauss-Newton direction is prone to pointing parameters toward infinity, especially in regions where the metric has very small eigenvalues. (B) **Rotated Gauss-Newton Directions.** By adding a small damping parameter to the metric, the Gauss-Newton direction is rotated into the gradient direction. The amount of rotation is determined by the eigenvalues of the metric at any given point. Here, only a few points are rotated significantly. (C) **Gradient Directions.** For large values of the damping parameter, the natural direction is rotated everywhere into the gradient direction.

That is, we want to use priors to prevent parameter combinations which are not constrained by the data from taking excessively large values – we want to avoid parameter evaporation. To illustrate again why this is problematic in sloppy models, consider a linear sloppy model with true parameters θ_0, but fit to data with added noise ξ_i. The observed best fit is then shifted to $\theta = \theta_0 + (J^T J)^{-1}(J^T)\xi$. The measurement error in data space ξ_i is thus multiplied by the inverse of the poorly conditioned matrix $g = J^T J$, so even a small measurement error produces a large parameter-space error. In section 2.8.7, we will see in nonlinear models that such noise will generally shift the best fits to the boundary (infinite parameter values) along directions where the noise is large compared to the width of the model manifold. Thus for example in fitting exponentials, positive noise in the data point at $t_0 = 0$ and negative noise at the data point at the first time $t_1 > 0$ can lead to one decay rate that evaporates to infinity, tuned to fit the first data point without affecting the others.

In practice, it is not often useful to know that the optimum value of a parameter is actually infinite – especially if that divergence is clearly due to noise. Also, we have seen in Fig. 2.7a that, even if the best fit has sensible parameters, algorithms searching for the best fits can be led toward the model manifold boundary. If the parameters are diverging at finite cost, the model must necessarily become insensitive to the diverging parameters, often leading the algorithm to get stuck. Even a very weak prior whose residuals diverge at the model manifold boundaries can prevent these problems, holding the parameters in ranges useful for fitting the data.

In this section, we advocate the use of priors for helping algorithms navigate the model manifold in finding good fits. These priors are pragmatic; they are not introduced to buffer a model with ‘prior knowledge’ about the system, but to use the data to guess the parameter ranges outside of which the fits will become insensitive to further parameter changes. Our priors do not have meaning in the Bayesian sense, and indeed should probably be relaxed to zero at late stages in the fitting process.

The first issue is how to guess what ranges of parameter are useful in fits – outside of which the model behavior becomes insensitive to the parameter values. Consider, for example, the Michaelis-Mentin reaction, a saturable reaction rate often arising in systems biology (for example Reference [18]):

$$\frac{d[x^*]}{dt} = \frac{k_x[y^*][x]}{1 + km_x[x]}. \tag{2.19}$$

Here there are two parameters k_x and km_x, governing the rate of production of $[x^*]$ from $[x]$ in terms of the concentration $[y^*]$, where $[x]+[x^*] = x_{max}$ and $[y]+[y^*] = y_{max}$.

Several model boundaries can be identified here. If k_x and $km_x x_{max}$ are both very large, then only their ratio affects the dynamics. In addition if km_x is very small then it has no effect on the model. Our prior should enforce our belief that $km_x[x]$ is typically of order 1. If it were much larger than one, than we could have modeled the system with one less parameter $k = k_x/km_x$ and if it were much less than one, the second term in the denominator could have been dropped entirely. Furthermore, if the data is best fit by one of these boundary cases, say $km_x x_{max} \to \infty$, it will be fit quite well by taking $km_x x_{max} >> 1$, but otherwise finite. In a typical model we might expect that $km_x x_{max} = 10$ will behave as if it were infinite.

We can also place a prior on k_x. Dimensional analysis here involves the time scale at which the model is predictive. The prior should match the approximate time scale of the model's predictions to the rate of the modeled reaction. For example, if an experiment takes time series data with precision on the order of seconds with intervals on the order minutes, then a 'fast' reaction is any that takes place faster than a few seconds and a slow reaction is any that happens over a few minutes. Even if the real reaction happens in microseconds, it makes no sense to extract such information from the model and data. Similarly, a slow reaction that takes place in years could be well fit by any rate that is longer than a few minutes. As such we want a prior which prevents $k_x y_{max} x_{max}/\tau$ from being far from 1, where τ is the typical timescale of the data, perhaps a minute here. In summary, we want priors to constrain both $km_x x_{max}$ and $k_x x_{max} y_{max}/\tau$ to be of order one.

We have found that a fairly wide range of priors can be very effective at minimizing the problems associated with parameter evaporation during fitting. To choose them, we propose starting by changing to the natural units of the problem by dividing by constants, such as time scales or maximum protein concentrations, until all of the parameters are dimensionless. (Alternatively, priors could be put into the model in the original units, at the expense of more complicated book-keeping.) In these natural units we expect all parameters to be order 1.

The second issue is to choose a form for the prior. For parameters like these, where both large and near-zero values are to be avoided, we add two priors for every parameter, one which punishes high values, and one which punishes small values:

$$Pr(\theta) = \begin{pmatrix} \sqrt{w_h \theta} \\ \sqrt{w_l/\theta} \end{pmatrix}. \tag{2.20}$$

This prior has minimum contribution to the cost when $\theta^2 = \frac{w_l}{w_h}$ so in the proper units we choose $w_h = w_l$. With these new priors, the metric becomes

$$\begin{aligned} g_{\mu\nu} &= \partial_\mu r^{0i} \partial_\nu r^{0i} + \partial_\mu Pr(\theta) \partial_\nu Pr(\theta) && (2.21) \\ &= g^0_{\mu\nu} + \delta_{\mu\nu} (\frac{w_l}{\theta^\mu} + w_h \theta^\mu), && (2.22) \end{aligned}$$

which is positive definite for all (positive) values of θ. As boundaries occur when the metric has an eigenvalue of zero, no boundaries exist for this new model manifold. This is reminiscent of the metric of the model graph with the difference being that we have permanently added this term to the model. The best fit has been shifted in this new metric.

It remains to choose w_h and w_l. Though the choice is likely to be somewhat model specific, we have found that a choice between .001 and 1 tends to be effective. That weights of order 1 can be effective is somewhat surprising. It implies that good fits can be found while punishing parameters for differing only an order of magnitude from their values given by dimensional analysis. That this works is a demonstration of the extremely ill-posed nature of these sloppy models, and the large ensemble of potential good fits in parameter space.

A complimentary picture of the benefit of priors takes place in parameter space, where they contribute to the cost:

$$C = C_0 + \sum_i w_h \theta_i / 2 + w_l / (2\theta_i). \tag{2.23}$$

The second derivative of the extra cost contribution with respect to the log of the parameters is given by $\frac{\partial^2}{\partial \log(\theta)^2} \left(\frac{Pr(\theta)^2}{2} \right) = \frac{w_h \theta}{2} + \frac{w_l}{2\theta}$. This is positive definite and concave, making the entire cost surface large when parameters are large. This in turn makes the cost surface easier to navigate by removing the problems associated with parameter evaporation on plateaus.

To demonstrate the effectiveness of this method, we use the PC12 model with 48 parameters described in [18]. We change to dimensionless units as described above. To create an ensemble, we start from 20 initial conditions, with each parameter taken from a Gaussian distribution in its log centered on 0 (the expected value from dimensional analysis), with a $\sigma = \log 10$ (so that the bare parameters range over roughly two orders of magnitude from .1 to 10). We put a prior as described above centered on the initial condition, with varying weights. These correspond to the priors that we would have calculated if we had found those values by dimensional analysis instead. After minimizing with the priors, we remove them and allow the algorithm to re-minimize. The results are plotted in Fig. 2.10.

Strikingly, even when a strong prior is centered at parameter values a factor of ~ 100 away from their 'true' values, the addition of the prior in the initial stages of convergence dramatically increases the speed and success rate of finding the best fit.

In section 2.5, we introduced the model graph and the Levenberg-Marquardt algorithm, whose rationale (to avoid parameter evaporation) was similar to that motivating us here to introduce priors. To conclude this section, we point out that the model graph metric, Eq. (2.17), and the metric for our particular choice of prior, Eq. (2.22), both serve to cut off large steps along sloppy directions. Indeed, the Levenberg-Marquardt algorithm takes a step identical to that for a model with quadratic priors (Eq. (2.18)) with $\sigma_m \equiv 1/\sqrt{\lambda}$, except that the center of the prior is not a fixed set of parameters θ_0, but the current parameter set θ^*. (That is, the second derivative of the sum of the squares of these residuals, $\sum_m [\sqrt{\lambda}(\theta - \theta^*)]^2$ gives $\lambda\delta_{\mu\nu}$, the Levenberg term in the metric.) This Levenberg term thus acts as a 'moving prior' – acting to limit individual algorithmic steps from moving too far toward the model boundary, but not biasing the algorithm permanently toward

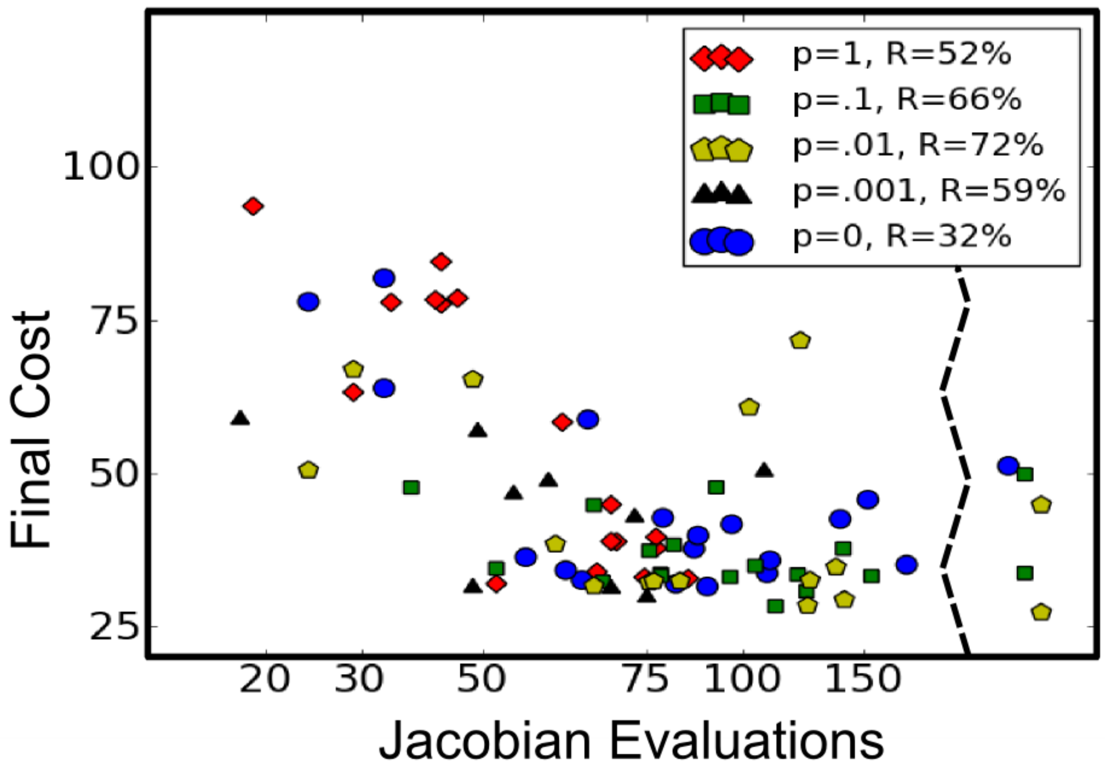


Figure 2.10: Algorithm performance with priors
(Color online) The final cost is plotted against number of Jacobian evaluations for five strengths of priors. After minimizing with priors, the priors are removed and a maximum of 20 further Jacobian evaluations are performed. The prior strength is measured by p, with $p = 0$ meaning no prior. The success rate is R. The strongest priors converge the fastest, with medium strength priors showing the highest success rate.

sensible values. Despite the use of a variable λ that can be used to tune the algorithm toward sensible behavior (Fig. 2.9), we shall see in section 2.9 that the Levenberg-Marquardt algorithm often fails, usually because of parameter evaporation. When the useful ranges of parameters can be estimated beforehand, adding priors can be a remarkably effective tool.

2.7 Extended Geodesic Coordinates

We have seen that the two difficulties of optimizing sloppy models are that algorithms tend to run into the model boundaries and that model parametrization tends to form long, curved canyons around the best fit. We have discussed how the first problem can be improved by the introduction of priors. We now turn our attention to the second problem. In this section we consider the question of whether we can change the parameters of a model in such a way as to remove this difficulty. We construct coordinates geometrically by considering the motion of geodesics on the manifold.

Given two nearby points on a manifold, one can consider the many paths that connect them. If the points are very far away, there may be complications due to the boundaries of the manifold. For the moment, we assume that the points are sufficiently close that boundaries can be ignored. The unique path joining the two points whose distance is shortest is known as the geodesic. The parameters corresponding to a geodesic path can be found as the solution of the differential equation

$$\ddot{x}^\mu + \Gamma^\mu_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta = 0, \tag{2.24}$$

where $\Gamma^\mu_{\alpha\beta}$ are the connection coefficients given by Eq. (2.8) and the dot means differentiation with respect to the curve's affine parametrization. Using two points as boundary values, the solution to the differential equation is then the shortest distance between the two points. Alternatively, one can specify a geodesic with an initial point and direction. In this case, the geodesic is interpreted as the path drawn by parallel transporting the tangent vector (also known as the curve's velocity). This second interpretation of geodesics will be the most useful for understanding the coordinates we are about to construct. The coordinates that we consider are polar-like coordinates, with $N-1$ angular coordinates and one radial coordinate.

If we consider all geodesics that pass through the best fit with a normalized velocity, $v^\mu v_\mu = 1$, then each geodesic is identified by $N-1$ free parameters, corresponding to direction of the velocity at the best fit. (The normalization of the velocity does not change the path of the geodesic – only the time it takes to traverse the path.) These $N-1$ free parameters will be the angular coordinates of the new coordinate system. There is no unique way of defining the angular coordinates. One can choose N orthonormal unit vectors at the best fit, and let the angular coordinates define a linear combination of them. We typically choose eigendirections of the metric (the eigenpredictions of section 2.3). Having specified a geodesic with the $N-1$ angular coordinates, the radial coordinate represents the distance moved along the geodesic. Since we have chosen the velocity vector to be normalized to one, the radial component is the parametrization of the geodesic.

We refer to these coordinates as extended geodesic coordinates and denote their Cartesian analog by γ^μ. These coordinates have the special property that those geodesics that pass through the best fit appears as straight lines in parameter space. (It is impossible for all geodesics to be straight lines if the space is curved.)

In general, one cannot express this coordinate change in an analytic form. The quadratic approximation to this transformation is given by

$$\gamma^\nu \approx \theta^\nu_{bf} + v^\nu_\mu \delta\theta^\mu + \frac{1}{2}\Gamma^\nu_{\alpha\beta}\delta\theta^\alpha\delta\theta^\beta. \qquad (2.25)$$

The coordinates given in Eq. (2.25) are known as Riemann normal coordinates or geodesic coordinates. Within the general relativity community, these coordinates are known as locally inertial reference frames because they have the property that $\Gamma^{\alpha}_{\mu\nu}(x = 0) = 0$, that is, the Christoffel symbols vanish at the special point around which the coordinates are constructed [71].

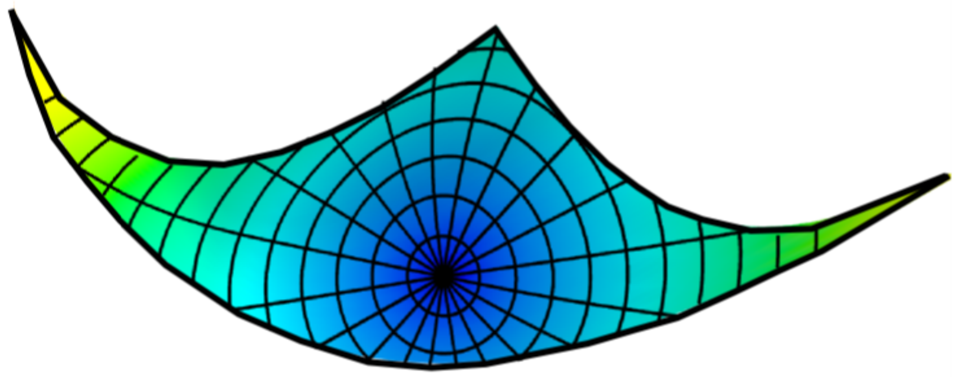
Let us now consider the shape of cost contours for our example model using extended geodesic coordinates. We can consider both the shape of the coordinate mesh on the manifold in data space, as well as the shape of the cost contours in parameter space. To illustrate the dramatic effect that these coordinates can have, we have adjusted the data so that the best fit does not lie so near the boundary. The results are in Fig. 2.7.

The extended geodesic coordinates were constructed to make the elongated ellipse that is characteristic of sloppy models become circular. It was hoped that by making the transformation nonlinear, it would straighten out the an-harmonic "banana" shape, rather than magnify it. It appears that this wish has been granted spectacularly. Not only has the banana been straightened out within the region of the long narrow canyon, but the entire region of parameter space, including the plateau, has been transformed into one manageable, isotropic basin. Indeed, the cost contours of Fig. 2.7b are near-perfect circles, all the way to the boundary where the rates go to zero, infinity, or are equal.

To better understand how this elegant result comes about, let's consider how the cost changes as we move along a geodesic that passes through the best fit. The cost then becomes parametrized by the same parameter describing the geodesic, which we call τ. The chain rule gives us,

$$\frac{d}{d\tau} = \frac{d\theta^{\mu}}{d\tau}\frac{\partial}{\partial\theta^{\mu}} = v^{\mu}\partial_{\mu},$$

a)



b)

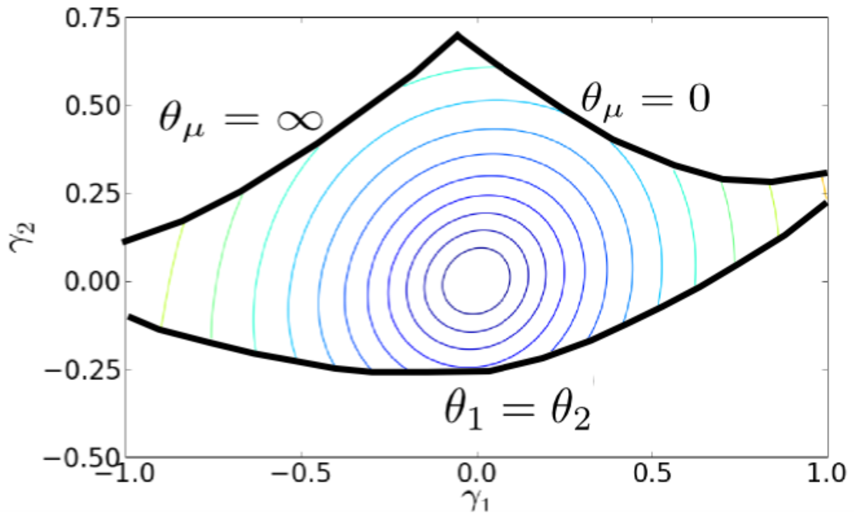


Figure 2.11: Geodesic Coordinates

(Color online) a) **Extended Geodesic Coordinates.** The parameters of a model are not usually well suited to describing the behavior of a model. By considering the manifold induced in data space, one can construct more natural coordinates based on geodesic motion that are more well-suited to describing the behavior of a model (black grid). These coordinates remove all parameter-effects curvature and are known as extended geodesic coordinates. Note that we have moved the data point so that the best fit is not so near a boundary in this picture. b) **Cost Contours in Extended Geodesic Coordinates.** Although the summing exponential model is nonlinear, that non-linearity does not translate into large extrinsic curvature. This type of non-linearity is known as parameter-effects curvature, which the geodesic coordinates remove. This is most dramatically illustrated by considering the contours of constant cost in geodesic coordinates. The contours are nearly circular all the way out to the fold line and the boundary where the rates are infinite.

where $v^\mu = \dot{\theta}^\mu$. Applying this twice to the cost gives:

$$\frac{d^2C}{d\tau^2} = v^\mu v^\nu g_{\mu\nu} + r_m P^N_{mn} \partial_\mu \partial_\nu r_n \frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}. \tag{2.26}$$

The term $v^\mu v^\nu g_{\mu\nu}$ in Eq. (2.26) is the arbitrarily chosen normalization of the velocity vector and is the same at all points along the geodesic. The interesting piece in Eq. (2.26) is the expression

$$P^N = \delta - J\left(J^T J\right)^{-1} J^T,$$

which we recognize as the projection operator that projects out of the tangent space (or into the normal bundle).

Recognizing P^N in Eq. (2.26), we see that any deviation of the quadratic behavior of the cost will be when the non-linearity forces the geodesic out of the tangent plane, which is to say that there is an extrinsic curvature. When there is no such curvature, then the cost will be isotropic and quadratic in the extended geodesic coordinates.

If the model happens to have as many parameters as residuals, then the tangent space is exactly the embedding space and the model will be flat. This can be seen explicitly in the expression for P^N, since J will be a square matrix if $M = N$, with a well-defined inverse:

$$\begin{aligned} P^N &= \delta - J\left(J^T J\right)^{-1} J^T \\ &= \delta - J J^{-1}\left(J^T\right)^{-1} J^T \\ &= 0. \end{aligned}$$

Furthermore, when there are as many parameters as residuals, the extended geodesic coordinates can be chosen to be the residuals themselves, and hence the cost contours will be concentric circles.

In general, there will be more residuals than parameters; however, we have seen in section 2.4 that many of those residuals are interpolating points that do not supply much new information. Assuming that we can simply discard a few residuals, then we can “force” the model to be flat by restricting the embedding space. It is, therefore, likely that for most sloppy models, the manifold will naturally be much more flat than one would have expected. We will see when we discuss curvature in section 2.8 that most of the non-linearities of a sloppy model do not produce extrinsic curvature, meaning the manifold is typically much more flat that one would have guessed.

Non-linearities that do not produce extrinsic curvature are described as parameter-effects curvature [9]. As the name suggests these are “curvatures” that can be removed through a different choice of parameters. By using geodesics, we have found a coordinate system on the manifold that removes all parameter-effects curvature at a point. It has been noted previously that geodesics are linked to zero parameter-effects curvature [58].

We believe it to be generally true for sloppy models that non-linearities are manifested primarily as parameter-effects curvature as we argue in [94] and in section 2.8. We find similar results when we consider geodesic coordinates in the PC12 model, neural networks, and many other models. Just as for the summing exponential problem that produced Fig. 2.7b, cost contours for this real-life model are nearly circular all the way to the model’s boundary.

Although the model manifold is much more flat than one would have guessed, how does that result compare for the model graph? We observed in section 2.5, that the model graph interpolates between the model manifold and the parameter space picture. If we find the cost contours for the model graph at various values of

λ, we can watch the cost contours interpolate between the circles in Fig. 2.7b and the long canyon that is characteristic of parameter space. This can be seen clearly in Fig. 2.12.

With any set of coordinates, it is important to know what portion of the manifold they cover. Extended geodesic coordinates will only be defined in some region around the best fit. It is clear from Fig. 2.7 that for our example problem the region for which the coordinates are valid extends to the manifold boundaries. Certainly there are regions of the manifold that are inaccessible to the geodesic coordinates. Usually, extended geodesic coordinates will be limited by geodesics reaching the boundaries, just as algorithms are similarly hindered in finding the best fit.

2.8 Curvature

In this section, we discuss the various types of curvature that one might expect to encounter in a least-squares problem and the measures that could be used to quantify those curvatures. Curvature of the model manifold has had many interesting applications. It has been illustrated by Bates and Watts that the curvature is a convenient measure of the non-linearity of a model [9, 10, 12]. When we discuss the implications of geometry on numerical algorithms this will be critical, since it is the non-linearity that makes these problems difficult.

Curvature has also been used to study confidence regions [10, 50, 28, 34, 100], kurtosis (deviations from normality) in parameter estimation [49], and criteria for determining if a minimum is the global minimizer [32]. We will see below that the large anisotropy in the metric produces a similar anisotropy in the curvature of sloppy models. Furthermore, we use curvature as a measure of how far an algorithm

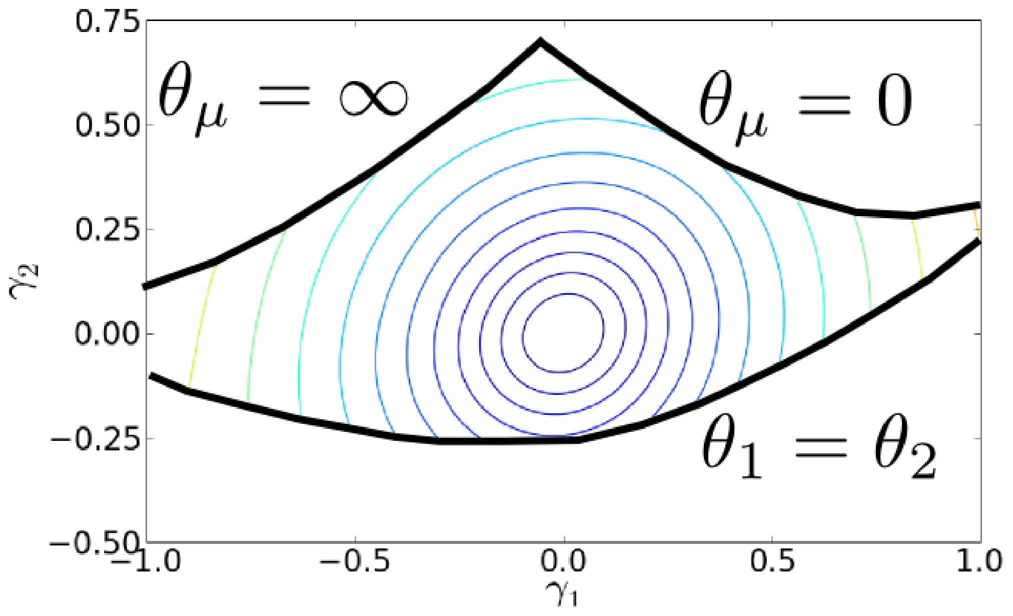


Figure 2.12: Cost contours in geodesic coordiantes
(Color online) By changing the value of the Levenberg-Marquardt parameter, the course of the geodesics on the corresponding model graph are deformed, in turn distorting the shape of the cost contours in the geodesic coordinates. a) $\lambda = 0$ is equivalent to the model manifold. The cost contours for a relatively flat manifold, such as that produced by the sum of two exponentials, are nearly perfect, concentric circles. The geodesics can be evaluated up to the boundary of the manifold, at which point the coordinates are no longer defined. Here we can clearly see the stiff, long manifold direction (vertical) and the sloppy, thin manifold direction (horizontal) b) **Small** λ, (λ much smaller than any of the eigenvalues of the metric) will produce cost contours that are still circular, but the manifold boundaries have been removed. In this case the fold line has disappeared, and cost contours that ended where parameters evaporated now stretch to infinity. c) **Moderate** λ creates cost contours that begin to stretch in regions where the damping parameter significantly affects the eigenvalue structure of the metric. The deformed cost contours begin to take the plateau and canyon structures of the contours in parameter space. d) **Large** λ effectively washes out the information from the model manifold metric, leaving just a multiple of the parameter space metric. In this case, the contours are those of parameter space – a long narrow curved canyon around the best fit. This figure analogous to Fig. 2.1b, although the model here is a more sloppy (and more realistic) example. An animation of the transition from small to large damping parameter is available in the online supplemental material [93].

can accurately step (section 2.8.6) and to estimate how many parameters a best fit will typically evaporate (section 2.8.7).

In our discussion of geodesic coordinates in section 2.7, we saw how some of the non-linearity of a model could be removed by a clever choice of coordinates. We also argued that the non-linearity that could not be removed by a coordinate change would be expressed as an extrinsic curvature on the expectation surface. Non-linearity that does not produce an extrinsic curvature is not irrelevant; it can still have strong influence on the model and can still limit the effectiveness of optimization algorithms. Specifically, this type of non-linearity changes the way that distances are measured on the tangent space. They may cause the basis vectors on the tangent space to expand, shrink, or rotate. We follow the nomenclature of Bates and Watts and refer to this type of non-linearity as parameter-effects curvature [9, 12]. We emphasize that this is not a "real" curvature in the sense that it does not cause the shape of the expectation surface to vary from a flat surface, but its effects on the behavior of the model is similar to the effect of real curvature. This "curvature" could be removed through a more convenient choice of coordinates, which is precisely what we have done by constructing geodesic coordinates in section 2.7. A functional definition of parameter-effects curvature would be the non-linearities that are annihilated by operating with P^N. Alternatively, one can think of the parameter-effects curvature as the curvatures of the coordinate mesh. We discuss parameter-effects curvature in section 2.8.3.

Bates and Watts refer to all non-linearity that cannot be removed by changes of coordinates as intrinsic curvature [12]. We will not follow this convention; instead, we follow the differential geometry community and further distinguish between intrinsic or Riemann curvature (section 2.8.1) and extrinsic or embedding curvature [88] (section 2.8.2). The former refers to the curvature that could be measured on a surface without reference to the embedding. The latter refers to the curvature that arises due to the manner in which the model has been embedded. From a complete knowledge of the extrinsic curvature, one could also calculate the intrinsic curvature. Based on our discussion to this point, one would expect that both the intrinsic and the extrinsic curvature should be expressible in terms of some combination of P^N and $\partial_\mu \partial_\nu r_m$. This turns out to be the case, as we will shortly see.

All types of curvature appear in least squares models, and we will now discuss each of them.

2.8.1 Intrinsic (Riemann) Curvature

The embedding plays a crucial role in nonlinear least squares fits – the residuals embed the model manifold explicitly in data space – we will be primarily interested in the extrinsic curvature. However, because most studies of differential geometry focus on the intrinsic curvature, we discuss it.

The Riemann curvature tensor, $R^\alpha_{\beta\gamma\delta}$ is one measure of intrinsic curvature. Since intrinsic curvature makes no reference to the embedding space, curvature is measured by moving a vector, V^μ, around infinitesimal closed loops and observing the change the curvature induces on the vector, which is expressed mathematically by

$$R^\alpha_{\beta\gamma\delta} V^\beta = \nabla_\gamma \nabla_\delta V^\alpha - \nabla_\delta \nabla_\gamma V^\alpha.$$

This expression in turn can be written independently of V^μ in terms of the Christoffel symbols and their derivatives by the standard formula

$$R^{\alpha}_{\beta\gamma\delta} = \partial_\gamma \Gamma^{\alpha}_{\beta\delta} - \partial_\delta \Gamma^{\alpha}_{\beta\gamma} + \Gamma^{\epsilon}_{\beta\delta}\Gamma^{\alpha}_{\epsilon\gamma} - \Gamma^{\epsilon}_{\beta\gamma}\Gamma^{\alpha}_{\epsilon\delta}.$$

From this we can express $R^{\alpha}_{\beta\gamma\delta}$ in terms of derivatives of the residuals. Even though $R^{\alpha}_{\beta\gamma\delta}$ depends on derivatives of Γ, suggesting that it would require a third derivative of the residuals, one can in fact represent it in terms of second derivatives and P^N,

$$R_{\alpha\beta\gamma\delta} = \partial_\alpha \partial_\gamma r_m P^N_{mn} \partial_\beta \partial_\delta r_n - \partial_\alpha \partial_\delta r_m P^N_{mn} \partial_\beta \partial_\gamma r_n,$$

which the Gauss-Codazzi equation extended to the case of more than one independent normal direction [36].

The toy model that we have used throughout this work to illustrate concepts has intrinsic curvature. The curvature becomes most apparent when viewed from another angle, as in Fig. 2.13.

Intrinsic or Riemann curvature is an important mathematical quantity that is described by a single, four-index tensor; however, we do not use intrinsic curvature to study optimization algorithms. Extrinsic and parameter-effects curvature in contrast not be simple tensors but will depend on a chosen direction. These curvatures are the key to understanding nonlinear least squares fitting.

2.8.2 Extrinsic Curvature

Extrinsic curvature is easier to visualize than intrinsic curvature since it makes reference to the embedding space, which is where one naturally imagines curved surfaces. It is important to understand that extrinsic and intrinsic curvature are fundamentally different and are not merely different ways of describing the same concept. In differentiating between intrinsic and extrinsic curvature, the simplest

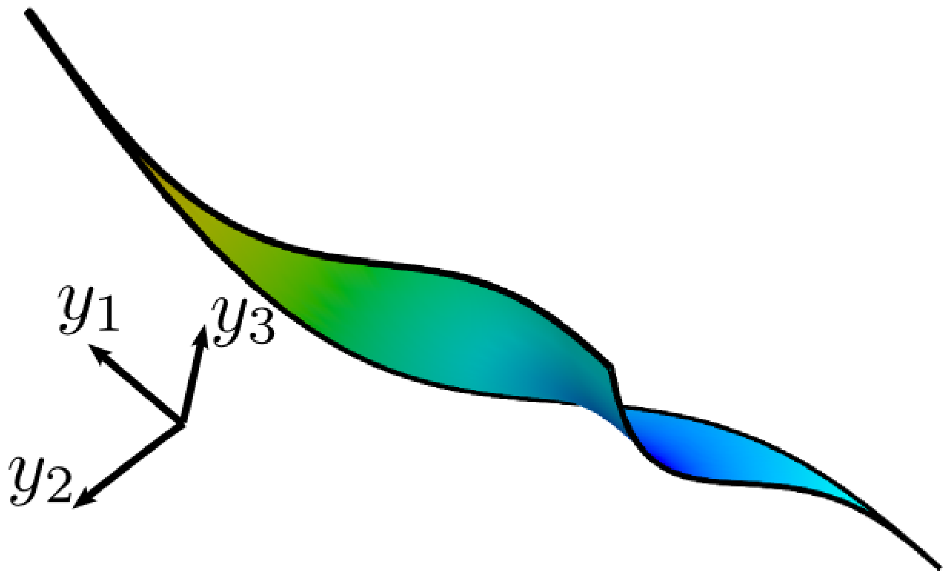


Figure 2.13: Intrinsic curvature
(Color online) **Intrinsic and Extrinsic Curvature.** Intrinsic Curvature is inherent to the manifold and cannot be removed by an alternative embedding. A model that is the sum of two exponential terms has all types of curvature. This is the same model manifold as in Fig. 2.1c, viewed from an alternative angle to highlight the curvature. From this viewing angle, the extrinsic curvature becomes apparent. This is also an example of intrinsic curvature. An animation of this surface rotating in three dimensions is available in the online supplemental material [93].

illustrative example is a cylinder, which has no intrinsic curvature but does have extrinsic curvature. One could imagine taking a piece of paper, clearly a flat, two dimensional surface embedded in three dimensional space, and roll it into a cylinder. Rolling the paper does not affect distances on the surface, preserving its intrinsic properties, but changes the way that it is embedded in three dimensional space. The rolled paper remains intrinsically flat, but it now has an extrinsic curvature. A surface whose extrinsic curvature can be removed by an alternative, isometric embedding is known as a ruled surface [53]. While an extrinsic curvature does not always imply the existence of an intrinsic curvature, an intrinsic curvature

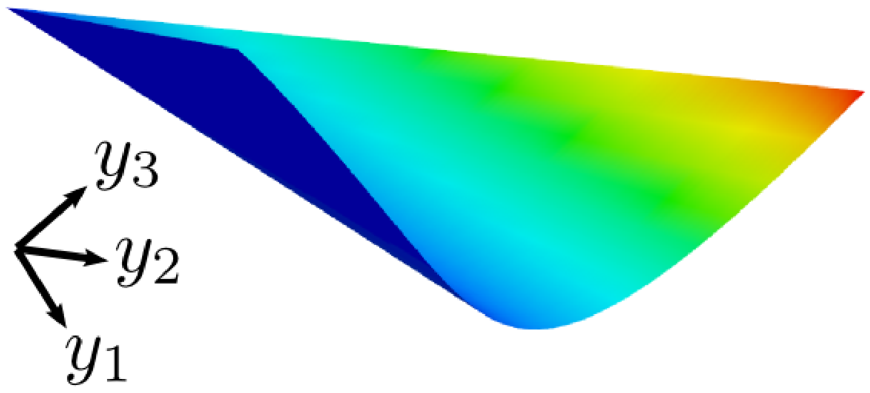


Figure 2.14: Ruled surface
(Color online) **A ruled surface** has no intrinsic curvature; however, it may have extrinsic curvature. The model manifold formed from a single exponential rate and amplitude is an example of a ruled surface. This model could be isometrically embedded in another space to remove the curvature. An animation of this surface rotating in three dimensional space is available in the online supplemental material [93].

requires that there also be extrinsic curvature. Our toy model, therefore, also exhibits extrinsic curvature as in Fig. 2.13. One model whose manifold is a ruled surface is given by a two parameter model which varies an exponential rate and an amplitude:

$$y = Ae^{-\theta t}.$$

The manifold for this model with three data points is drawn in Fig. 2.14 [3].

[3]This example is also a separable nonlinear least squares problem. Separable problems containing a mixture of linear and nonlinear parameters are amenable to the method known as variable projection [45, 59, 44]. Variable projection consists of first performing a linear least squares optimization on the linear parameters, making them implicit functions of the nonlinear parameters. The geometric effect of this procedure is to reduce the dimensionality of the model manifold, effectively selecting a sub-manifold which now depends upon the location of the data. We will not discuss this method further in this paper, but we note that it is likely to have interesting geometric properties.

There are two measures of extrinsic curvature that we discuss. The first is known as geodesic curvature as it measures the deviation of a geodesic from a straight line in the embedding space. The second measure is known as the shape operator. These two measures are complimentary, and should be used together to understand the way a space is curved. Both geodesic curvature and the shape operator have analogous measures of parameter-effects curvature that will allow us to compare the relative importance of the two types of curvature.

Measures of extrinsic and parameter effects curvature to quantify non-linearities have been proposed previously by Bates and Watts [9, 8, 12]. Although the measure they use is equivalent to the presentation of the next few sections, their approach is different. The goal of this section is to express curvature measures of non-linearity in a more standard way using the language of differential geometry. By so doing, we hope to make the results accessible to a larger audience.

Geodesic Curvature

Consider a geodesic parametrized by τ, tracing a path through parameter space, $\theta^\mu(\tau)$, which in turn defines a path through residual space, $\vec{r}(\theta(\tau))$. The parametrization allows us to discuss the velocity, $\vec{v} = \frac{d\vec{r}}{d\tau}$, and the acceleration, $\vec{a} = \frac{d\vec{v}}{d\tau}$. A little calculus puts these expressions in a more practical form:

$$\vec{v} = \dot{\theta}^\mu \partial_\mu \vec{r},$$

$$\vec{a} = \dot{\theta}^\mu \dot{\theta}^\nu P^N \partial_\mu \partial_\nu \vec{r}.$$

Notice that the normal projection operator emerges naturally in the expression for $\vec{a}$.

For any curve that has instantaneous velocity and acceleration vectors, one can

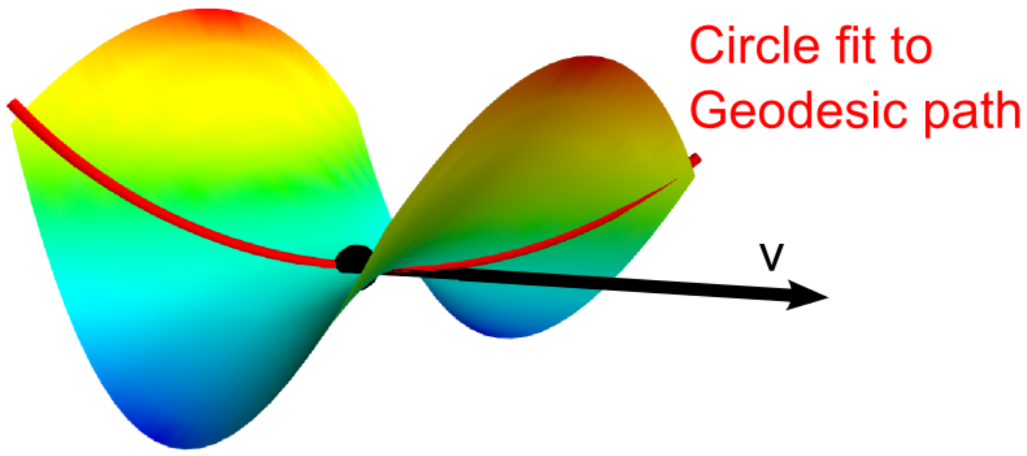


Figure 2.15: Geodesic Curvature
(Color online) **Geodesic Curvature.** A direction on a curved surface define a geodesic. The deviation of the geodesic from a straight line in the embedding space is measured by the geodesic curvature. It is the inverse radius of the circle fit to the geodesic path at the point. A three-dimensional animation of this surface is available in the online supplemental material [93].

find a circle that local approximates the path. The circle has radius

$$R = \frac{v^2}{|\vec{a}|},$$

and a corresponding curvature

$$K = R^{-1} = \frac{|\vec{a}|}{v^2}.$$

Because the path that we are considering is a geodesic, it will be as near a straight line in data space as possible without leaving the expectation surface. That is to say, the curvature of the geodesic path will be a measure of how the surface is curving within the embedding space, i.e. an extrinsic curvature. The curvature associated with a geodesic path is illustrated in Fig. 2.15.

In our previous discussion of geodesics, we saw that a geodesic is fully specified by a point and a direction. Therefore we can define the geodesic curvature of any point on the surface, corresponding to a direction, v^μ, by

$$K(v) = \frac{|v^\mu v^\nu P^N \partial_\mu \partial_\nu \vec{r}|}{v^\alpha v_\alpha}. \tag{2.27}$$

At each point an the surface, there is a different value of the geodesic curvature for each direction on the surface.

Shape Operator

Another measure of extrinsic curvature, complimentary to the geodesic curvature, is the shape operator, $S_{\mu\nu}$. While the geodesic curvature requires us to choose an arbitrary direction on the surface, the shape operator requires us to choose an arbitrary direction normal to the surface.

To understand the shape operator, let us first consider the special case of an N-dimensional surface embedded in an $N+1$-dimensional space. If this is the case, then at any point on the surface there is a unique (up to a sign) unit vector normal to the surface, $\hat{n}$. If this is the case, $S_{\mu\nu}$ is given by

$$S_{\mu\nu} = \hat{n} \cdot (\partial_\mu \partial_\nu \vec{r}). \tag{2.28}$$

$S_{\mu\nu}$ is known as the shape operator because it describes how the surface is shaped around the unit normal, $\hat{n}$. It is a symmetric, covariant rank-2 tensor. We are usually interested in finding the eigenvalues of the shape operator with a single raised index:

$$S^\mu_\nu = g^{\mu\alpha} S_{\alpha\nu}.$$

The eigenvectors of S^μ_ν are known as the principal curvature directions, and the eigenvalues are the extrinsic curvatures in those directions. In the case that there

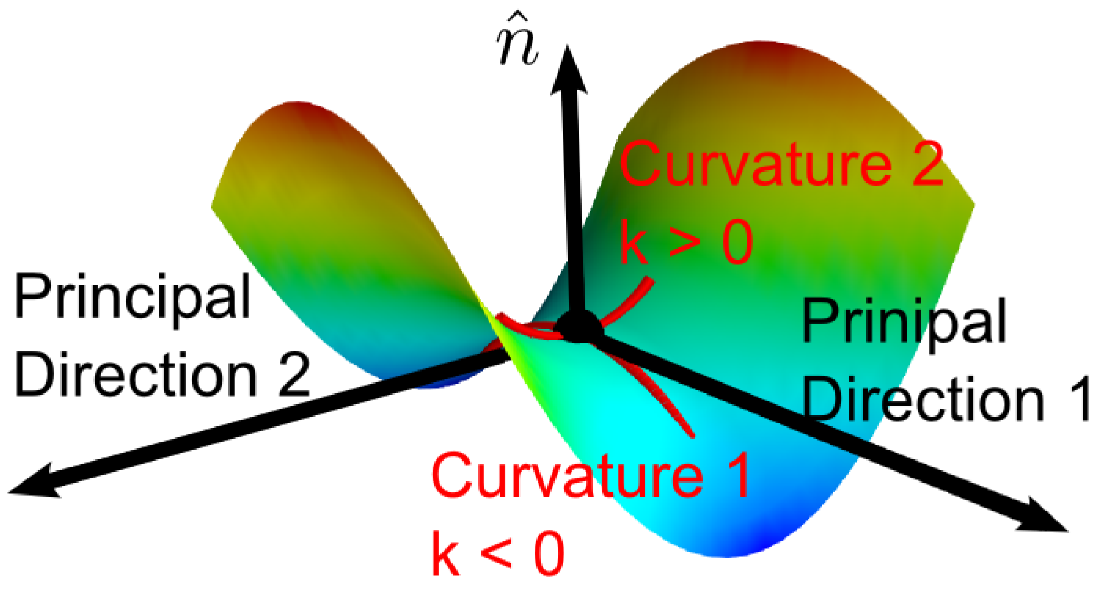


Figure 2.16: Shape operator
(Color online) **Shape Operator.** Specifying a direction normal to a curved surface, $\hat{n}$, defines a shape operator. The eigenvalues of the shape operator are the principle curvatures and the corresponding eigenvectors are the directions of principle curvature. A three-dimensional animation of this surface is available in the online supplemental material [93].

is only one direction normal to the surface, then the (absolute value of the) eigenvalues of S^{μ}_{ν}, are equal to the geodesic curvatures in the respective eigendirections. The eigenvalues, $\{k_\mu\}$, may be either positive or negative. Positive values indicate that the curvature is toward the direction of the normal, while negative values indicate that it is curving away, as illustrated in Fig. 2.16.

In general, there will not be an unique normal vector. If an N-dimensional surface is embedded in an M-dimensional space, then there will $M-N$ independent shape operators, and one is left to perform an eigenvalue analysis for each as described above [88]. Fortunately, for the case of a least squares problem, there is a natural direction to choose: the normal component of the unfit data, $-P^N\vec{r}$, making the shape operator

$$S_{\mu\nu} = -\frac{\vec{r} P^N \partial_\mu \partial_\nu \vec{r}}{|P^N \vec{r}|}, \tag{2.29}$$

where we introduce the minus as convention. In general, around an arbitrary vector $\vec{V}$, the shape operator becomes

$$S(\vec{V})_{\mu\nu} = \frac{\vec{V} P^N \partial_\mu \partial_\nu \vec{r}}{|P^N \vec{V}|}. \tag{2.30}$$

It should now be clear why these two measures of extrinsic curvature (geodesic curvature and the shape operator) are complimentary. The geodesic curvature is limited by having to choose a direction tangent to the surface, but gives complete information about how that direction is curving into the space normal to the surface. In contrast, the shape operator gives information about all the directions on the surface, but only tells how those directions curve relative to a single normal direction.

2.8.3 Parameter-effects Curvature

We are now prepared to discuss parameter-effects curvature. We repeat that parameter-effects curvature is not a curvature of the manifold. Instead, it is a measure of the curvatures of the coordinate mesh on the surface. In our experience, parameter-effects curvature is typically the largest of the three types we have discussed. By its very nature, this curvature depends on the choice of the parametrization. By constructing extended geodesic coordinates in section 2.7, we were able to remove the parameter-effects curvature from the model (at a point). In this section we will discuss how to measure the parameter-effects curvature and compare it to the other curvatures that we discussed above.

To understand the meaning of parameter-effects curvature, let us begin by considering a linear model with no curvature of any type. For simplicity, we consider the parametrization of the xy-plane given by

$$\begin{aligned} x &= \epsilon\theta_1 + \theta_2 \\ y &= \theta_1 + \epsilon\theta_2. \end{aligned}$$

This parametrization will produce a skewed grid as $\epsilon \to 1$, characteristic of linear sloppy models, such as fitting polynomials. This grid is illustrated in Fig. 2.17a for $\epsilon = 1/2$. By reparametrizing the linear model, we can introduce parameter-effects curvature. For example, if we replace the parameters with their squares (which may be useful if we wish to enforce the positivity of the parameters' effects)

$$\begin{aligned} x &= \epsilon\theta_1^2 + \theta_2^2 \\ y &= \theta_1^2 + \epsilon\theta_2^2, \end{aligned}$$

then the corresponding coordinate mesh will become compressed and stretched, as seen in Fig. 2.17b. Alternatively, if we reparametrize the model as

$$\begin{aligned} x &= (\epsilon\theta_1 + \theta_2)^2 \\ y &= \left(\theta_1^2 + \epsilon\theta_2^2\right)^2, \end{aligned}$$

in order to limit the region of consideration to the upper-right quarter plane, then the coordinate mesh will stretch and rotate into itself, depicted in Fig. 2.17c. With more than two parameters, there is additionally a torsion parameter-effects curvature in which the lines twist around one another. None of these reparametrization change the intrinsic or extrinsic properties of the model manifold; they merely change how the coordinates describe the manifold. The extent to which coordinate mesh is nonlinear is measured by the parameter-effects curvature.

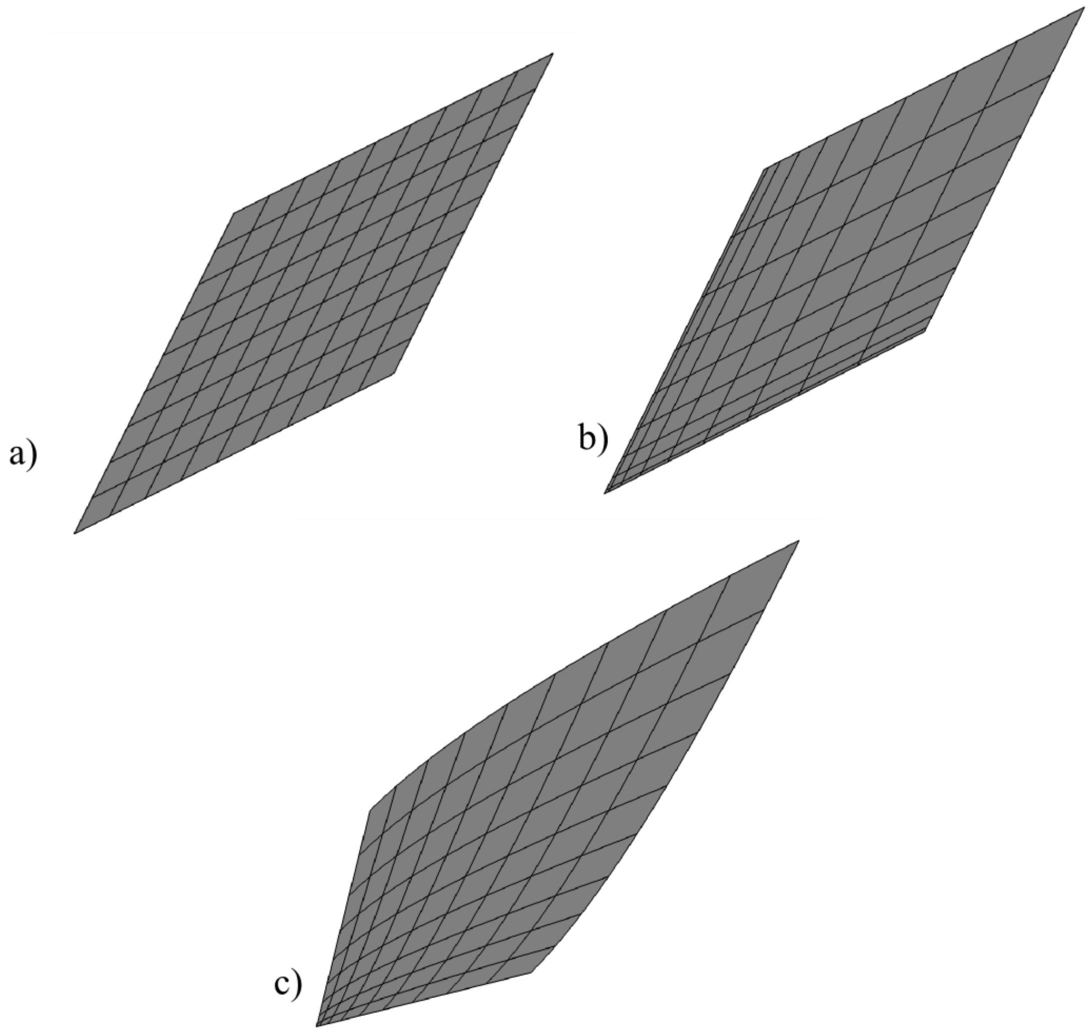


Figure 2.17: Parameter effects curvature
a) **Linear Grid.** A sloppy linear model may have a skewed coordinate grid, but the shape of the grid is constant, having no parameter effects curvature. b) **Compressed Grid.** By reparametrizing the model, the grid may become stretched or compressed in regions of the manifold. c) **Rotating, Compressed Grid.** Another parametrization may not only stretch the grid, but also cause the coordinates to rotate. Parameter-effects curvature describes the degree to which the coordinates are stretching and rotating on the manifold. With more than two parameters, there is also a torsion parameter-effects curvature (twisting).

We now consider how to quantify parameter-effects curvature. We have discussed the normal and tangential projection operators, P^N and P^T, and argued that the normal projection operator would extract the extrinsic and intrinsic curvature from the matrix of second derivatives. Looking back on our expressions for curvature up to this point, we see that each involves P^N. The complimentary parameter-effects curvature can be found by replacing P^N with P^T in each expression. Thus, in analogy with Eq. (2.27), we can define the parameter-effects geodesic curvature by

$$K^p(v) = \frac{|v^\mu v^\nu P^T \partial_\mu \partial_\nu \vec{r}|}{v^\alpha v_\alpha}. \tag{2.31}$$

Likewise, we can define a parameter-effects shape operator by comparison with Eq. (2.29),

$$S^p_{\mu\nu} = -\frac{\vec{r} P^T \partial_\mu \partial_\nu \vec{r}}{|P^T \vec{r}|}.$$

Recall that for an N-dimensional space embedded in an M-dimensional space, there are $M - N$ independent shape operators. This is because the space normal to the tangent space (into which we are projecting the non-linearity) is of dimension $M - N$. The parameter-effects analog must therefore have N independent shape operators, since the projection space (the tangent space) is N-dimensional. Therefore, we are naturally led to define a parameter-effects shape-operator with an additional index to distinguish among the N possible tangent directions,

$$S^P_{m\mu\nu} = P^T_{mn} \partial_\mu \partial_\nu r_n.$$

If we resolve these shape operators into the natural basis on the tangent space, $S^P_{m\mu\nu} = S^{p\alpha}_{\mu\nu} \partial_\alpha r_m$, we find

$$S^{P\alpha}_{\mu\nu} = g^{\alpha\beta} \partial_\beta \vec{r} \cdot \partial_\mu \partial_\nu \vec{r} = \Gamma^\alpha_{\mu\nu}.$$

Therefore, the parameter-effects curvature is correctly interpreted as the connection coefficients. With this understanding, it is clear that geodesic coordinates remove parameter-effects curvature, since they are the coordinates constructed to give $\Gamma = 0$.

Finally, we note that from a complete knowledge of all the curvatures (for all directions) one can determine the matrix of second derivatives completely. Although we do not demonstrate this here, we note it is a consequence of having a flat embedding space.

2.8.4 Curvature in Sloppy Models

Based on our analysis thus far, we should have two expectations regarding the curvature of sloppy models. First, because of the large spread of eigenvalues of the metric tensor, unit distances measured in parameter space correspond to large ranges of distances in data space. Conversely, one has to move the parameters by large amounts in a sloppy direction in order to change the residuals by a significant amount. Because of this, we expect that the anharmonicities in the sloppy directions will become magnified when we consider the curvature in those directions. We expect strong anisotropies in the curvatures of sloppy models, with the largest curvatures corresponding to the sloppiest directions.

Secondly, as we saw in section 2.7, by changing coordinates to extended geodesic coordinates, we discovered that the manifold generated by our sloppy model was surprisingly flat, i.e. had low intrinsic curvature. We have seen that if the model happens to have equal number of data points as parameters, then the model will always be flat. Since many of the data points in a typical sloppy model are just interpolation points, we believe that in general sloppy models have lower extrinsic curvature than one would have naively guessed just by considering the magnitude of the non-linearities. This explains perhaps why we will find that the dominant curvature of sloppy models is the parameter-effects one.

We can better understand the size of the various curvatures by considering the interpretation presented in section 2.4 that sloppy models are a generalized interpolation scheme. If we choose N independent data points as our parametrization, then the interpolating polynomial, $P_{N-1}(t)$ in Eq. (2.13) is a linear function of the parameters. As discussed below that equation, the manifold in each additional direction will be constrained to within $\epsilon = \delta f_{N+1}$ of $P_{N-1}(t)$. Presuming that this deviation from flatness smoothly varies along the jth largest width $W_j \sim \delta f_j$ of the manifold (i.e., there is no complex or sensitive dependence on parameters), the geodesic extrinsic curvature is

$$K = \epsilon / W_j^2, \tag{2.32}$$

predicting a range of extrinsic curvatures comparable to the range of inverse eigenvalues of the metric. Furthermore, the ratio of the curvature to the inverse width should then be $\epsilon / W_j \sim \delta f_{N+1}/\delta f_j \sim (\delta t / R)^{N+1-j}$, where δt is the spacing of time points at which the model is sampled and R is the time scale over which the model changes appreciably (see the argument in section 2.4 following Eq. (2.13)).

Since we estimate $\epsilon = \delta f_{N+1}$ to be the most narrow width if the model had an additional parameter, we can find the overall scale of the extrinsic curvature to be given by the narrowest width

$$K_N \approx \frac{1}{W_N}.$$

Additionally, we can find the scale set by the parameter effects curvature by recalling that parameter effects curvature is the curvature of the coordinate mesh. If we ignore all parameter combinations except the stiffest, then motion in this direction traces out a one-dimensional model manifold. The parameter-effects curvature of the full model manifold in the stiffest direction now corresponds to the extrinsic curvature of this one-dimensional manifold [4], and as such is set by the smallest width (which in this case in the only width), i.e. the longest width of the full model manifold. The similar structure of parameter-effects curvature and extrinsic curvature, Eqs. (2.27) and (2.31), suggest that the parameter-effects curvature also be proportional to the inverse eigenvalues (squares of the widths) along the several cross sections. Combining these result, we see that in general the ratio of extrinsic to parameter-effects curvature to be given by ratio of the widest to the most narrow width,

$$\frac{K}{K^P} \approx \frac{W_N}{W_1} \approx \sqrt{\frac{\lambda_N}{\lambda_1}}. \tag{2.33}$$

In our experience the ratio of extrinsic to parameter-effects curvature in Eq. (2.33) is always very small. When Bates and Watts introduced parameter-effects curvature, they considered its magnitude on twenty four models and found it universally larger than the extrinsic curvature, often much larger [9]. We have here offered an explanation of this effect based on the assumption that the deviation from flatness is given by Eq. (2.32).

We explicitly check the assumption of Eq. (2.32) by calculating cross sections for a model of several exponentials and for an artificial neural network. We have already seen in section 2.4 in figure 2.6 that these widths span several orders of magnitude as predicted by the singular values of the Jacobian. In Fig. 2.18 we view the data space image of these widths (projected into the plane spanned by the local velocity and acceleration vectors), where we see explicitly that the

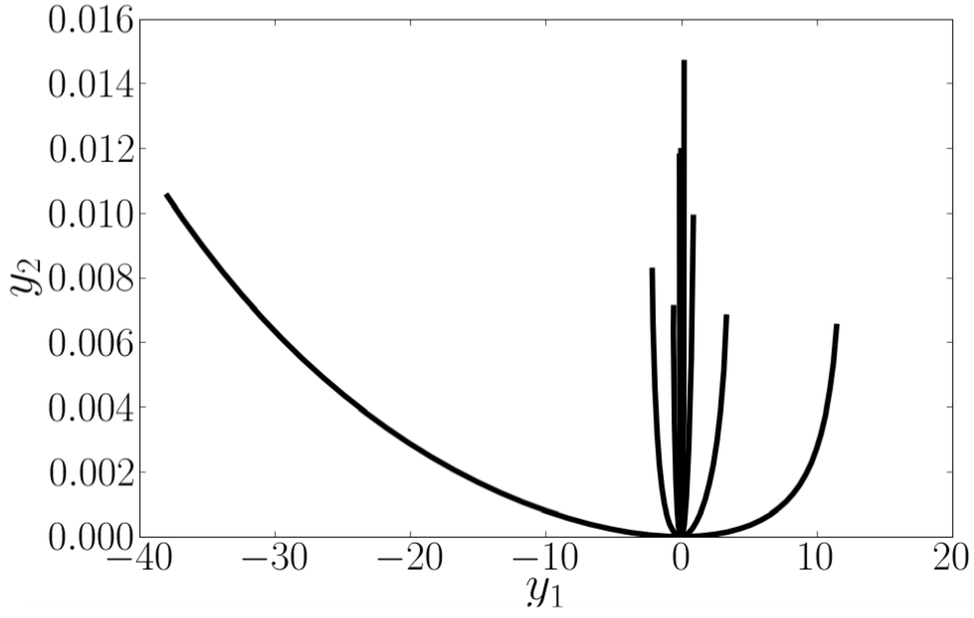
[4]This is strictly only true if the parameter-effects curvature has no compression component. Bates and Watts observe that typically, the compression is a large part of the parameter-effects curvature [9]. As long as the compression is not significantly larger than the rotation (i.e. is within an order of magnitude), the parameter-effects curvature will be the same order of magnitude as the extrinsic curvature of the one-dimensional model.

deviation from flatness is similar for all the cross sections. In Fig. 2.19 we see that that the extrinsic curvature is comparable to the narrowest cross section and the parameter-effects curvature is comparable to the widest cross section as we argued above, both for fitting exponentials and for the neural network model.

We further illustrate the above analysis by explicitly calculating the curvatures for the sloppy model formed by summing several exponential terms with amplitudes. Fig. 2.20 is a log-plot illustrating the eigenvalues of the inverse metric, the geodesic curvatures in each of those eigendirections, as well as the parameter-effects geodesic curvature in each of those directions. We see the same picture whether we consider the eigenvalues of the shape operator or the geodesic curvature. Both measures of curvature are strongly anisotropic with both extrinsic curvature and parameter-effects curvature covering as many orders of magnitude as the eigenvalues of the (inverse) metric. However, the extrinsic curvature is smaller by a factor roughly given by Eq. (2.33). We will use this large discrepancy between extrinsic and parameter-effects curvature when we improve the standard algorithms in section 2.9.

We have seen that manifolds of sloppy models possess a number of universal characteristics. We saw in section 2.4 that they are bounded with a hierarchy of widths which we describe as a hyper-ribbon. In this section we have seen that the extrinsic and parameter-effects curvature also possess a universal structure summarized in Figs. 2.18-2.21. A remarkable thing about the parameter-invariant, global structure of a sloppy model manifold is that is typically well-described by the singular values of the parameter-dependent, local Jacobian matrix. We saw in section 2.4 that the singular values correspond to the widths. We have now argued that the largest and smallest singular values set the scale of the parameter-effects

a)



b)

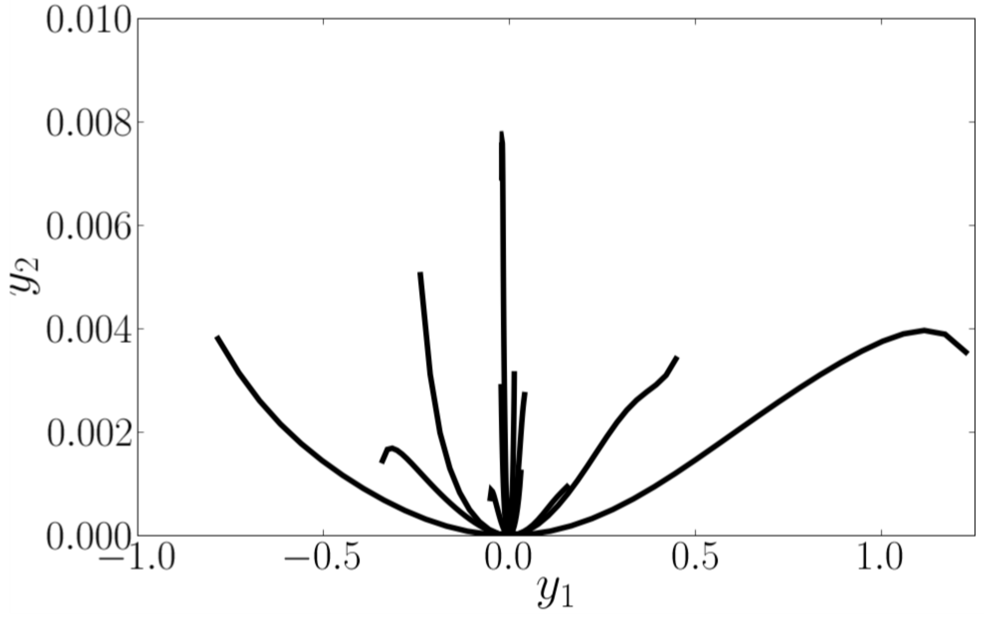
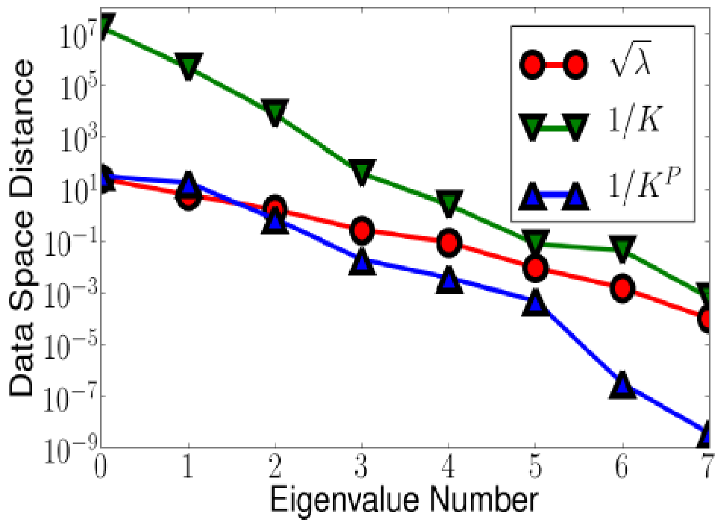


Figure 2.18: Model manifold cross sections
a) **Cross sections of a summing exponential model** projected into the plane spanned by the two principle components in data space. Notice the widths of successive cross sections are progressively more narrow, while the deviations from flatness are uniformly spread across the width. The magnitude of the deviation from flatness is approximately the same for each width, giving rise to the hierarchy of curvatures. b) **Cross sections of a feed forward neural network** has many of the same properties as the exponential model. In both cases, the curvature is much smaller than it appears due to the relative scale of the two axes. In fact, the sloppiest directions (narrowest widths) have an aspect ratio of about one.

a)


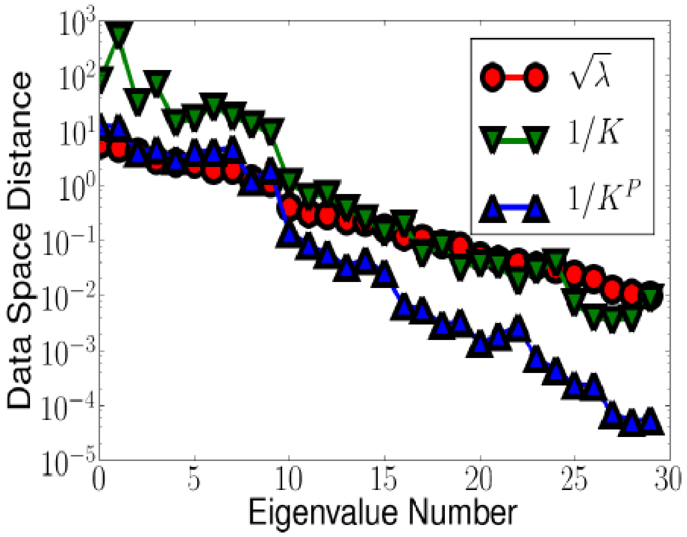
b)


Figure 2.19: Curvatures and widths on the model manifold
(Color online) The extrinsic and parameter-effects curvature on the model manifold are strongly anisotropic, with the largest curvatures along the shortest widths (see Figs. 2.6, 2.18). The slopes of the (inverse) curvature vs. eigenvalue lines are roughly twice that of the singular values (which are equivalent to the widths). The magnitude of the extrinsic curvature is set by the most narrow cross sections, while the magnitude of the parameter-effects curvature is set by the widest cross-section. Consequently the parameter-effect curvature is much larger than the extrinsic curvature. Here we plot the widths and curvatures for a model of four exponentials (above) from reference [94] and a feed forward artificial neural network (below)

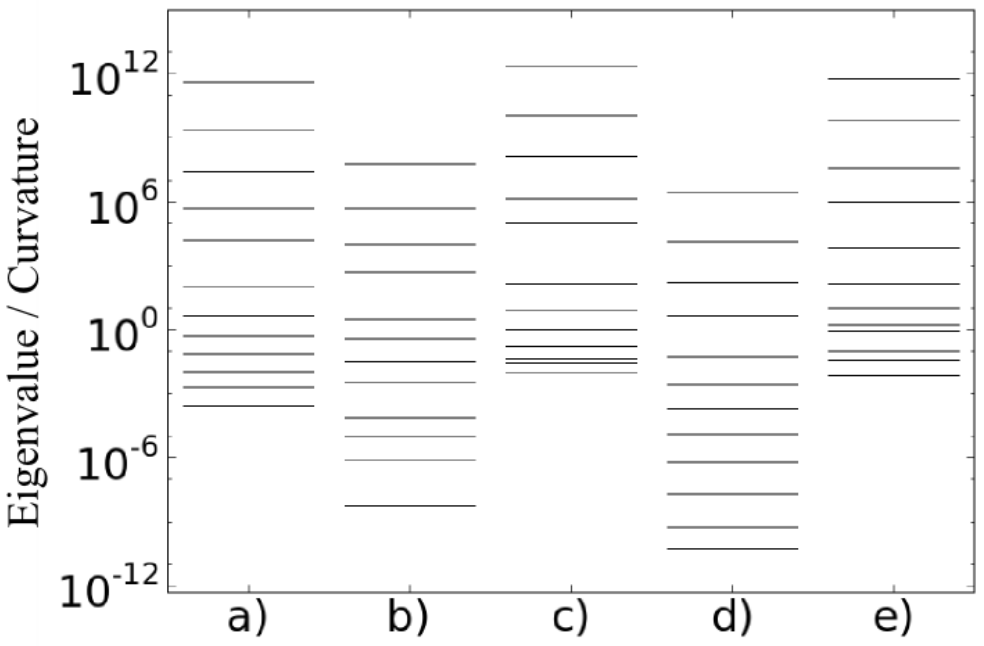


Figure 2.20: Curvature anisotropy on the model manifold
Curvature Anisotropy. a) **Inverse Metric eigenvalues.** The (inverse) metric has eigenvalues spread over several orders of magnitude, producing a strong anisotropy in the way distances are measured on the model manifold. b) **Geodesic Curvature in eigendirections of the metric.** The geodesic curvatures also cover many decades. The shortened distance measurements from the metric eigenvalues magnify the anharmonicities in the sloppy directions. c) **Parameter-Effects Geodesic Curvature.** The parameter-effects curvature is much larger than the extrinsic curvature, but shares the anisotropy. d) **The eigenvalues of the Shape Operator.** The strong curvature anisotropy described by the geodesic curvature is also illustrated in the eigenvalue spectrum of the shape operator. e) **Parameter-Effects Shape Operator eigenvalues.** Two measures (geodesic and shape operator curvatures) span similar ranges, but in both cases the parameter-effects curvature is a factor of about 10^5 larger than the extrinsic curvature equivalent.

and extrinsic curvatures respectively. This entire structure is a consequence of the observation that most models are a multi-dimensional interpolation scheme.

Let us summarize our conclusions about the geometry of sloppy models. We argued in section 2.4 using interpolation theorems that multiparameter nonlinear least-squares models should have model manifolds with a hierarchy of widths, forming a hyper-ribbon with the n^{th} width of order $W_n \sim W_0\Delta^n$ with Δ given by the spacing between data points divided by a radius of convergence (in some multidimensional sense) and W_0 the widest cross section. We discovered in some cases that the eigenvalues of the Hessian about the best fit agreed well with the squares of these widths (so $\lambda_n \sim \Delta^{2n}$, see Fig. 2.6). This depends on the choice of parameters and the placement of the best fit; we conjecture that this will usually occur if the 'bare' parameters are physically or biologically natural descriptions of the model and have natural units (i.e., dimensionless), and if the best fit is not near the boundary of the model manifold. The parameter Δ will depend on the model and the data being fit; it varies (for example) from 0.1 to 0.9 among seventeen systems biology models [48]. We argued using interpolation theory that the extrinsic curvatures should scale as $K_n \sim \epsilon/W_n^2$, where the total variation $\epsilon \sim W_N$, implying $K_n \sim \Delta^N/(W_0\Delta^{2n})$ (Fig. 18c). We find this hierarchy both measured along the eigenvectors of the (parameter-independent) shape operator (Fig. 2.20) or the geodesic curvatures measured along the (parameter-dependent) eigenpredictions at the best fit. Finally, we note that the parameter effects curvature also scales as $1/\Delta^{2n}$ by inspecting the similarity in the two formulae, Eqs. (2.27) and (2.31). We argue that the parameter-effects curvature should be roughly given by the extrinsic curvature of a one-dimensional model moving in a stiff direction, which sets the scale of the parameter effects as $K_n^P \sim W_0/W_n^2 \sim 1/(W_0\Delta^{2n})$, again either measured along the eigendirections of the parameter-effects shape operator or along eigenpredictions. Thus the entire structure of the manifold can be summarized by three numbers, W_0 the stiffest width, Δ the typical spacing between widths, and N the number of parameters. We summarize our conclusions in Fig. 2.21.

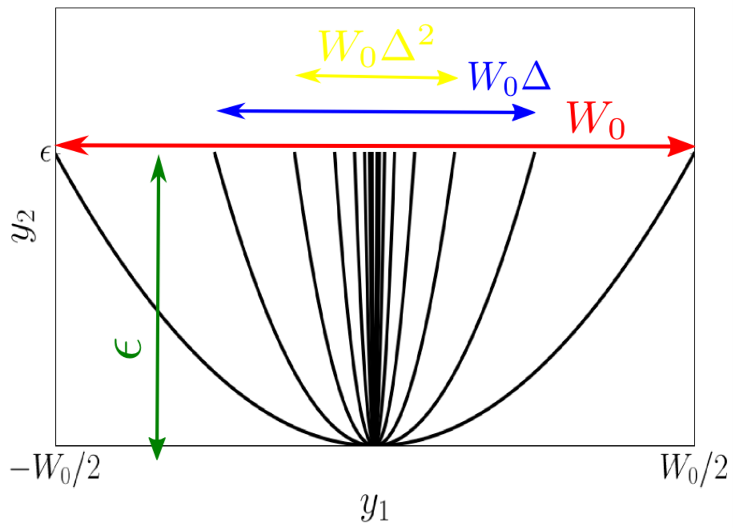
2.8.5 Curvature on the Model Graph

Most of the non-linearities of sloppy models appear as parameter-effects curvature on the model manifold. On the model graph, however, these non-linearities become extrinsic curvature because the model graph emphasizes the parameter dependence. An extreme version of this effect can be seen explicitly in Fig. 2.8, where the model manifold, which had been folded in half, is unfolded in the model graph, producing a region of high curvature around the fold line.

If the Levenberg-Marquardt parameter is sufficiently large, the graph can be made arbitrarily flat (assuming the metric chosen for parameter space is flat, such as for the Levenberg metric). This effect is also visible in Fig. 2.8 in the regions that stretch toward the boundaries. In these regions, the Levenberg-Marquardt parameter is much larger than the eigenvalues of the metric, making the parameter space metric the dominant contribution, and creating an extrinsically flat region on the model graph.

To illustrate how the curvature on the model graph is affected by the Levenberg-Marquardt parameter, we consider how the geodesic curvatures in the eigendirections of the metric change as the parameter is increased for a model involving several exponentials with amplitudes and rates. The results are plotted in Fig. 2.22. As the Levenberg-Marquardt parameter is raised, the widely ranging values of the geodesic curvatures may either increase or decrease. The largest curvature direc-

a)



b)

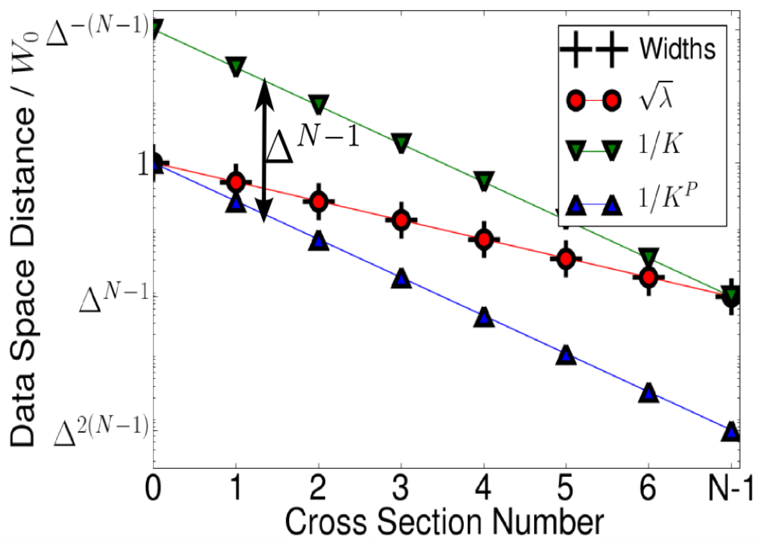


Figure 2.21: Caricature of the model manifold
(Color online) A caricature of the widths and curvatures of a typical sloppy model. a) The manifold deviates by an amount Δ^N from a linear model for each width. As each width is smaller than the last by a factor of Δ the curvature is largest along the narrow widths. This summary agrees well with the two real models in Fig. 2.18. b) The scales of the extrinsic and parameter-effects curvature are set by the narrowest and widest widths respectively. The parameter-effects curvature is therefore smaller than the extrinsic curvature by a factor of Δ^N. Both are strongly anisotropic. Compare this figure to with the corresponding result for the two real models in Fig. 2.19.

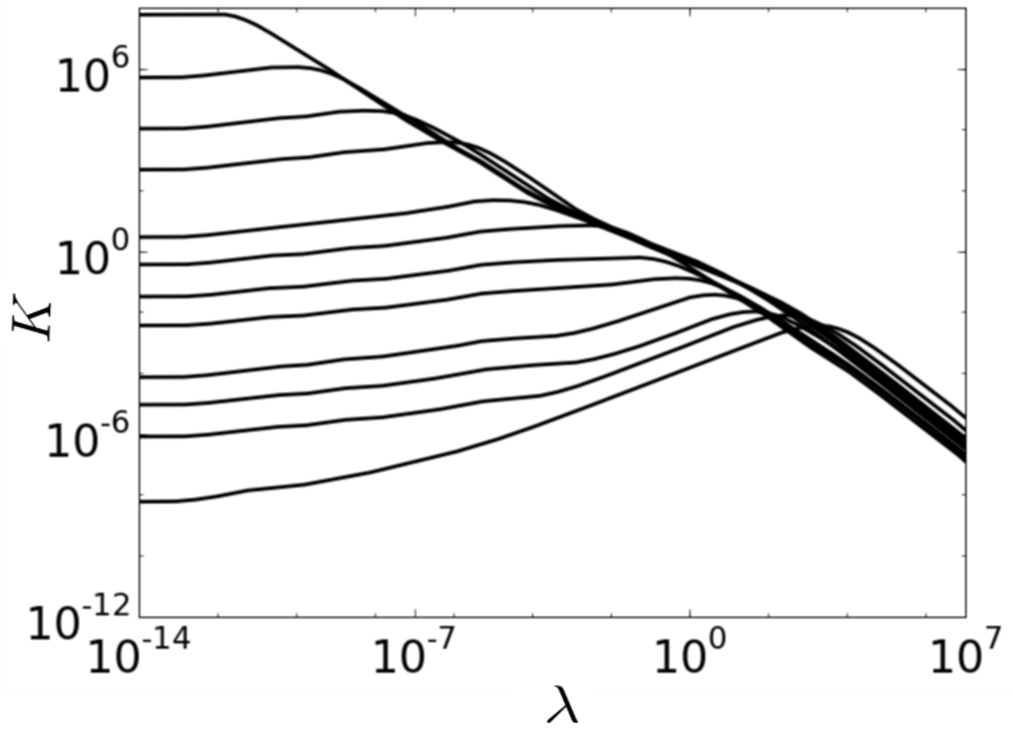


Figure 2.22: Curvature on the model graph
Model Graph Curvature. As the Levenberg-Marquardt parameter, λ, is increased, directions with highest curvature become less curved. For stiff directions with less extrinsic curvature, the parameter effects curvature may be transformed into extrinsic curvature. The damping term reduces the large anisotropy in the curvature. For sufficiently large values of the Levenberg-Marquardt parameters, all curvatures vanish.

tions (the sloppy directions) tend to flatten, but the directions with the lowest curvature (the stiff directions) direction become more curved. The main effect of the the Levenberg-Marquardt parameter is to decrease the anisotropy in the curvature.

The behavior of the extrinsic curvature as the Levenberg-Marquardt parameter is varied can best be understood in terms of the interplay between parameter-effects curvature and extrinsic curvature. Curvatures decrease as more weight is given to the Euclidean, parameter-space metric. However, as long as the parameter-space metric is not completely dominant, the graph will inherit curvatures from the model manifold. Since the graph considers model output versus the parameters, curvature that had previously been parameter-effects become extrinsic curvature. Therefore, directions that had previously been extrinsically flat will be more curved, while the directions with the most curvature will become less curved.

The largest curvatures typically correspond to the sloppy directions. Most algorithms will try to step in sloppy directions in order to follow the canyon. The benefit of the model graph is that it reduces the curvature in the sloppy directions, which allows algorithms to take larger steps. The fact that previously flat directions become extrinsically curved on the model graph does not hinder an algorithm that does not step in these extrinsically flat directions anyway. The role that curvatures play in determining an algorithm's maximal step size is looked at more closely in the next section.

2.8.6 Optimization Curvature

The distinction between extrinsic and parameter-effects curvature is not particularly useful in understanding the limitations of an algorithm. An iterative algorithm taking steps based on a local linearization will ultimately be limited by all non-linearities, both extrinsic and parameter-effects. We would like a measure of non-linearity, analogous to curvature, that explains the limitations of stepping in a given direction.

Suppose an algorithm proposes a step in some direction, v^μ, then the natural measure of non-linearity should include the directional second derivative,

$v^\mu v^\nu \partial_\mu \partial_\nu \vec{r} / v^\alpha v_\alpha$, where we included the normalization in order to remove the scale dependence of v. This expression is very similar to the geodesic curvature without the projection operator.

Simply using the magnitude of this expression is not particularly useful because it doesn't indicate whether curvature of the path is improving or hindering the convergence of the algorithm. This crucial bit of information is given by the (negative) dot product with the unit residual vector,

$$\kappa(v) = -\frac{v^\mu v^\nu \partial_\mu \partial_\nu \vec{r}}{v^\alpha v_\alpha} \cdot \frac{\vec{r}}{|\vec{r}|}, \tag{2.34}$$

which we refer to as the *Optimization Curvature*. Since the goal is to reduce the size of the current residual, the negative sign is to produce the convention that for $\kappa > 0$, the curvature is helping the algorithm while when $\kappa < 0$ the curvature is slowing the algorithm's convergence.

This expression for κ has many of the properties of the curvatures discussed in this section. It has the same units as the curvatures we have discussed. It requires the specification of both a direction on the manifold (the proposed step direction, v) and a direction in data space (the desired destination, $\vec{r}$), making it a combination of both the geodesic and shape operator measures of curvature. Furthermore, without the projection operators, it combines both extrinsic and parameter effects curvature into a single measure of non-linearity, although in practice, it is dominated by the parameter-effects curvature. We now consider how κ is related to the allowed step size of an iterative algorithm.

Consider the scaled Levenberg step given by

$$\delta\theta^\mu = -\left(g^0 + \lambda D\right)^{\mu\nu} \partial_\nu C \ \delta\tau.$$

Each λ specifies a direction for a proposed step. For a given λ, we vary $\delta\tau$ to find how far an algorithm could step in the proposed direction. We determine $\delta\tau$ by performing a line search to minimize the cost in the given direction. While minimizing the cost at each step may seem like a natural stepping criterion, it is actually a poor choice, as we discuss in section 2.9.3; however, this simple criteria is useful for illustrating the limitations on step size.

We measure the step size by the motion it causes in the residuals, $\|\delta\vec{r}\|$. This is a convenient choice because each direction also determines a value for the geodesic curvature (K), parameter-effects curvature (K^p), and an optimization curvature (κ), each of which are measured in units of inverse distance in data space. We compare the step size with the inverse curvature in each direction in Fig. 2.23.

One might assume that the size of the non-linearities always limits the step size, since the direction was determined based on a linearization of the residuals. This is clearly the case for the summing exponentials model in Fig. 2.23a, where $\kappa < 0$; the step size closely follows the largest of the curvatures, the parameter effects curvature $K^P \approx |\kappa|$.

However, the non-linearities on occasion may inadvertently be helpful to an algorithm, as in Fig. 2.23b where $\kappa > 0$. If the value of κ changes sign as we vary λ, then the distinction becomes clear: steps can be several orders of magnitude larger than expected if $\kappa > 0$, otherwise they are limited by the magnitude of κ. The sign of the parameter κ is illustrating something that can be easily understood by considering the cost contours in parameter space, as in Fig. 2.23d. If the canyon is curving “into” the proposed step direction, then the step runs up the canyon wall and must be shortened. However, if the canyon is curving “away” from the proposed step direction, then step runs down the canyon and eventually up the opposite wall, resulting in a much larger step size.

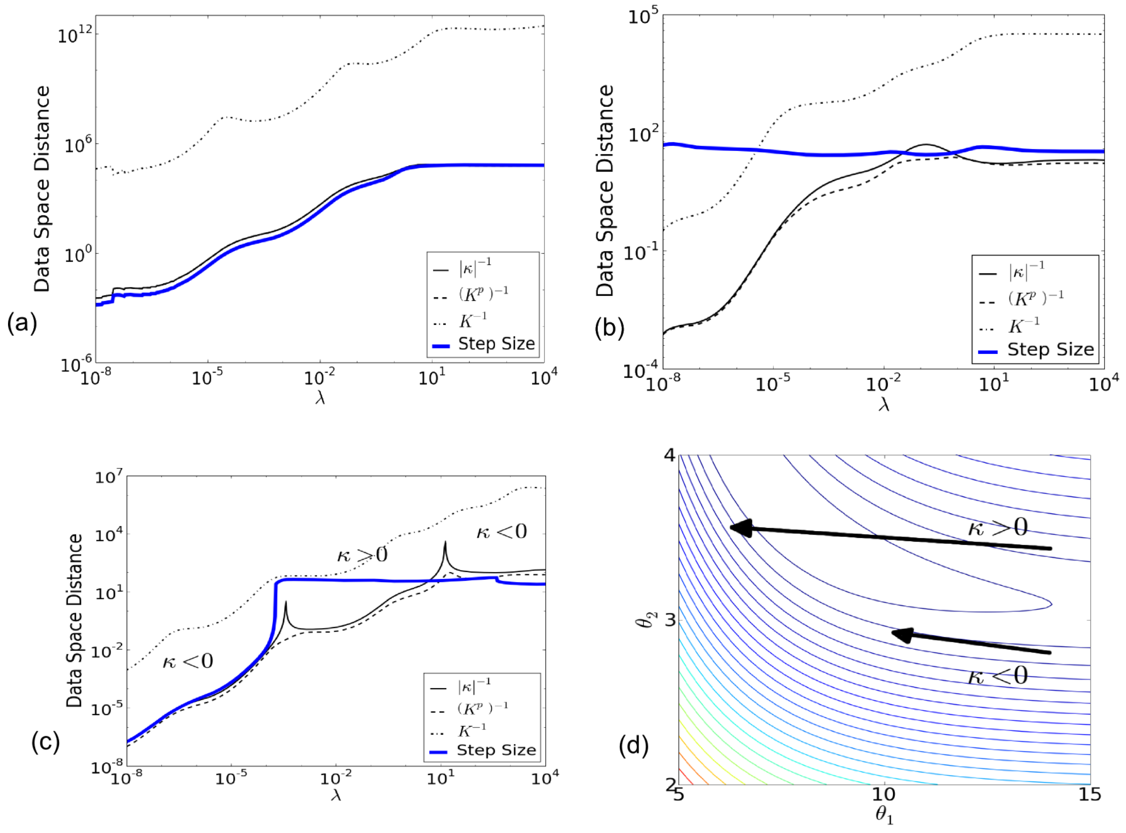


Figure 2.23: Curvature and algorithm step size
(Color online) a) **Curvature and Step Size for $\kappa < 0$.** If $\kappa < 0$, then the non-linearities in the proposed direction are diverting the algorithm away from the desired path. Distances are limited by the size of the curvature. b) **Curvature and Step Size for $\kappa > 0$.** The non-linearities may be helpful to an algorithm, allowing for larger than expected step sizes when $\kappa > 0$. c) **Curvature and Step Size for κ with alternating sign.** For small λ, $\kappa < 0$ and the non-linearities are restricting the step size. However, if κ becomes positive (the cusp indicates the change of sign), the possible step size suddenly increases. d) **Cost contours for positive and negative values of κ.** One can understand the two different signs of κ in terms of which side of the canyon the given point resides. The upper point has positive κ and can step much larger distances in the Gauss-Newton direction than can the lower point with negative κ, which quickly runs up the canyon wall.

2.8.7 Curvature and parameter evaporation

We have stressed the the boundaries of the model manifold are the major obstacle to optimization algorithms. Because a typical sloppy model has many very narrow widths, it is reasonable to expect the best fit parameters to have several evaporated parameter values when fit to noisy data. In order estimate the expected number of evaporated parameters, however, it is necessary to account for the extrinsic curvature of a model.

In Fig. 2.24 we illustrate how the curvature effects which regions of data space correspond to a best fit with either evaporated or finite parameters. A first approximation is a cross-sectional width with no extrinsic curvature, as in Fig. 2.24a. If the component of the data parallel to the cross-section does not lie outside the range of the width, the parameter will not evaporate. If the cross-section has curvature, however, the situation is more complicated, with the best fit depending on the component of the data perpendicular to the cross-section as well. Figs. 2.24(b) and (c) highlight the regions of data space for which the best fit will not evaporate parameters (gray).

Knowing both the regions of data space corresponding to non-evaporated parameters and the relative probabilities of the possible data (Eq. (2.2)), we can estimate the expected number of evaporated parameters for a given a model at the best fit. Using Gaussian data of width σ centered on the middle of a cross-section for a problem of fitting exponentials, we find the best fit and count the number zero-eigenvalues of the metric, corresponding to the number of non-evaporated parameters at the fit.

We can derive analytic estimates for the number of evaporated parameters using

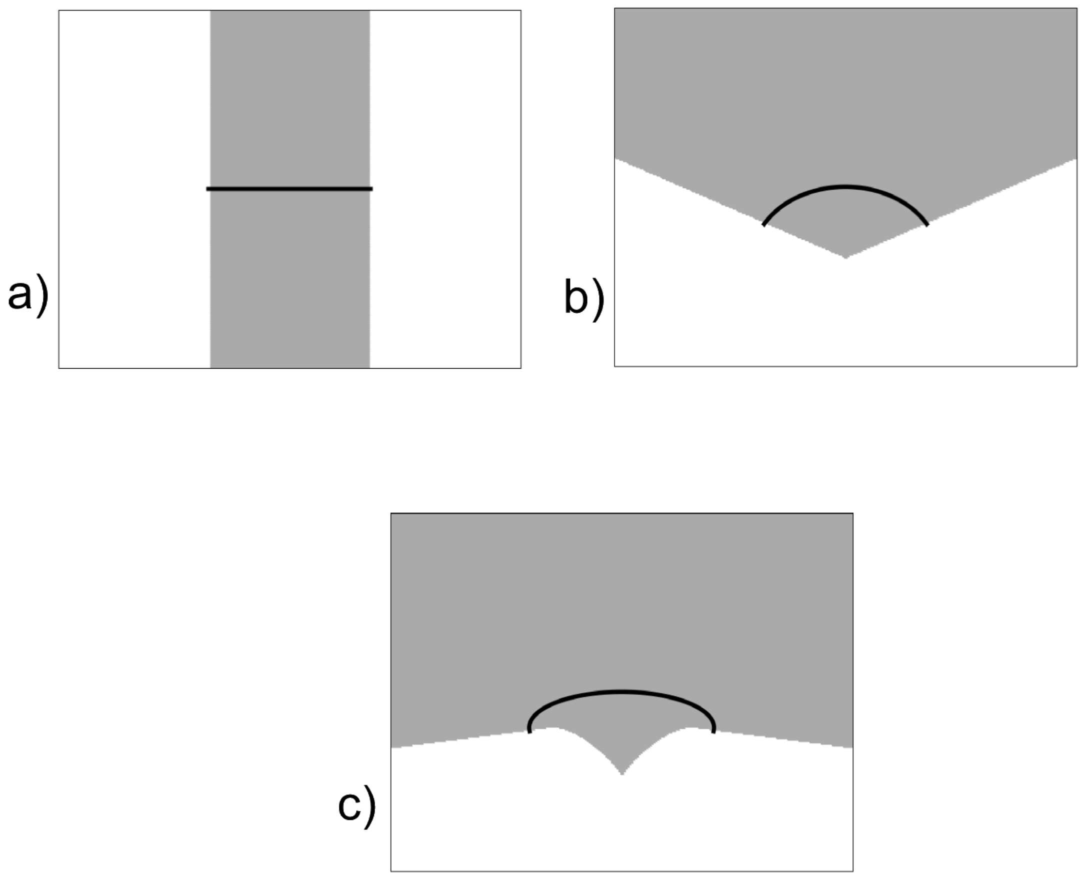


Figure 2.24: Probability of best fit lying on the boundary
The curvature along the width of a manifold effects if the best fit lies on the boundary or on the interior. For a cross-sectional width (thick black line), consider three possibilities: a) extrinsically flat, b) constant curvature along width, and c) curvature proportional to distance from the boundary. Grey regions correspond to data points with best fits on the interior of the manifold, while white regions correspond to data with evaporated parameters. If the curvature is larger near the boundaries, there is less data space available for evaporated best fit parameters.

the approximation that the cross section is either flat or has constant curvature as in Fig. 2.24a and b. If the cross-section is extrinsically flat then the probability of the corresponding parameter combination not evaporating is given in terms of the error function

$$P_n^{\text{flat}} = 2 \text{ erf}\left(\frac{W_n}{2\sigma}\right), \tag{2.35}$$

where W_n is the n^{th} width given by $W_n = W_0\Delta^n$.

A similar formula for the constant curvature approximation is a little more complicated. It involves integrating the Gaussian centered on the cross section in Fig. 2.24 over the gray region. Since the apex of the gray cone is offset from the center of the Gaussian, we evaluate the integral treating the offset as a perturbation. We recognize that there are several cases to be considered. If the noise is smaller than any of the widths, then the probability is approximately one. However, if the noise is larger than the width but smaller than inverse curvature, the probability is given by W_n/σ. Finally, if the noise is larger than any of the widths the probability is W_nK_n. Recall that we characterize a sloppy model manifold by three numbers, W_0, Δ, and N, the largest width, the average spacing between widths and the number of parameters respectively. The final result in each of the three cases in terms of these three numbers is given by

$$P_n^{\text{curved}} = \begin{cases} 1 & \text{if } \sigma < W_n, \\ \frac{W_0\Delta^n}{\sigma} & \text{if } W_n < \sigma < 1/K_n, \\ \Delta^{N-n} & \text{if } 1/K_n < \sigma. \end{cases} \tag{2.36}$$

From our caricature of a typical sloppy model summarized in Fig. 2.21, we estimate how many widths should belong in each category for a given σ. Summing the probabilities for the several widths in Eq. (2.36) we find the expected number of non-evaporated parameters to be given by

$$\langle N_{\text{approx}} \rangle = \frac{2}{1-\Delta} + \frac{\log \sigma / W_0}{\log \Delta} - 1. \tag{2.37}$$

In Table 2.1 we compare the fraction of non-evaporated parameters with the estimates from Eqs. (2.35) and (2.36). We find a large discrepancy when the noise in the data is very large. In this case there is often a large fraction of non-evaporated parameters even if the noise is much larger than any cross-sectional width. We attribute this discrepancy to larger curvatures near the corners of the manifold that increase the fraction of data space that can be fit without evaporating parameters. Since the metric is nearly singular close to a boundary, we expect the extrinsic curvature to become singular also by inspecting Eq. (2.27). We explicitly calculate the curvature near the boundary and we find that this is in fact the case.

The calculation in Table 2.1 can be interpreted in several ways. If one is developing a model to describe some data with known error bars, the calculation can be used to estimate the number of parameters the model could reasonably have without evaporating any at the best fit. Alternatively for a fixed model, the calculation indicates what level of accuracy is necessary in the data to confidently predict which parameters are not infinite. Qualitatively, for a given model, the errors must be smaller than the narrowest width for there to be no evaporated parameters.

Similarly, for experimental data with noise less than any of the (inverse) parameter-effects curvatures the parameter uncertainties estimated by the inverse Fisher information matrix will be accurate since the parameterization is constant over the range of uncertainty. It is important to note, that for models with large numbers of parameters either of these conditions require extremely small, often unrealistically small, error bars. In general, it is more practical to focus on pre-

σ	$\langle N \rangle / N$	$\langle N_{\text{flat}} \rangle / N$	$\langle N_{\text{integral}} \rangle / N$	$\langle N_{\text{approx}} \rangle / N$
$10W_0$	0.61	.0006	0.028	0.025
W_0	0.73	0.05	0.076	0.16
$\sqrt{W_0 W_N}$	0.87	0.50	0.52	0.60
W_N	0.95	0.92	0.93	1.00
$W_N/10$	0.98	1.00	1.00	1.00

Table 2.1: Fraction of nonevaporated parameters
The number of non-evaporated parameters $\langle N \rangle$ per total number of parameters N at the best fit, for an 8 parameter model of exponentials and amplitudes. As the noise of the data ensemble grows, the number of non-evaporated parameters at the best fit decreases (i.e. more parameters are evaporated by a good fit). Even if the noise is much larger than any of the widths, there are still several non-evaporated parameters, due to the curvature (see Fig. 2.24). We estimate the expected number of non-evaporated parameters from both a flat manifold approximation (Eq. (2.35)) and a constant curvature approximation. For the constant curvature approximation we show the result of the exact integral of the gaussian over the grey region of Fig. 2.24b as well as our perturbative approximation, Eq. (2.37), using the parameters $W_0 = 6.1$, $\Delta = .11$ and $N = 8$. These approximations agree with the numerical results when the noise is small, but for very noisy data there are still several non-evaporated parameters even if the noise is much larger than any of the widths. Therefore, although our general caricature of the model manifold as a hyper-cylinder of constant curvatures and widths seems to describe the geometry of the sloppy directions, it does not capture the features of the stiff directions. This discrepancy could be due, for example, by an increase in the curvature near the boundary as in Fig. 2.24c.

dictions made by ensembles of parameters with good fits rather than parameter values at the best fit as the latter will depend strongly the noise in the data.

2.9 Applications to Algorithmics

We now consider how the results derived in previous sections can be applied to algorithms. We have stressed that fitting sloppy models to data consist of two difficult steps. The first step is to explore the large, flat plateau to find the canyon. The second step is to follow the canyon to the best fit.

We begin by deriving two common algorithms, the modified Gauss-Newton method and the Levenberg-Marquardt algorithm from the geometric picture in sections 2.9.1 and 2.9.2. We then suggest how it may be improved by applying what we call delayed gratification and an acceleration term in sections 2.9.3 and 2.9.4.

We demonstrate that the suggested modifications can offer improvements to the algorithm by applying them to a few test problems in section 2.9.5. In comparing the effectiveness of algorithms we make an important observation, that the majority of the computer time for most problems with many parameters is occupied by Jacobian evaluations. As the number of parameters grows, this becomes increasingly the case. Models with many parameters are more likely to be sloppy, so this assumption does not greatly reduce the applicability of the algorithms discussed.

If an algorithm estimates the Jacobian from finite differences of the residuals, then most of the function (residual) evaluations will be spent estimating the Jacobian. (Our function evaluation counts in Table 2.2 do not include function evaluations used to estimate Jacobians.) If this is the case, then for any given problem, comparing function evaluations automatically integrates the relative expense of calculating residuals and Jacobians. However, many of the problems we use for comparison are designed to have only a few parameters for quick evaluation, while capturing the essence of larger problems. We then extrapolate results from small problems to similar, but larger problems. Our primary objective is to reduce the number of Jacobian evaluations necessary for an algorithm to converge. We do not ignore the number of function evaluations, but we but consider reducing the number of function calls to be a lower priority. As we consider possible improvements to algorithms, we will usually be willing to accept a few more function calls if it can significantly reduce the number of Jacobian evaluations that an algorithm requires.

In the next few sections, we discuss the geometric meaning of the Gauss-Newton method (section 2.9.1) and other similar algorithms, such as the Levenberg-Marquardt algorithm (section 2.9.2). We then discuss how ideas from differential geometry can lead to ways of improving convergence rates. First, we suggest a method of updating the Levenberg-Marquardt parameter, which we call delayed gratification, in section 2.9.3. Second, we suggest the inclusion of a geodesic acceleration term in section 2.9.4. We end the discussion by comparing the efficiency of standard versions of algorithms to those with the suggested improvements in section 2.9.5.

2.9.1 Modified Gauss-Newton Method

The result presented in this paper that appears to be the most likely to lead to a useful algorithm is that cost contours are nearly perfect circles in extended geodesic coordinates as described in section 2.7. The coordinates illustrated in Fig. 2.7 transformed a long, narrow, curved valley into concentric circles. Searching for the best fit in these coordinates would be a straightforward task! This suggests that an algorithm that begins at an unoptimized point need only follow a geodesic to the best fit. We have thus transformed an optimization problem into a differential equation integration problem.

The initial direction of the geodesic tangent vector (velocity vector) should be the Gauss-Newton direction

$$\frac{d\theta^\mu}{d\tau}(\tau = 0) = -g^{\mu\nu}\partial_\nu C. \tag{2.38}$$

If we assume that the manifold is extrinsically flat (the necessary and sufficient condition to produce concentric circles in extended geodesic coordinates), then Eq. (2.26) tells us that the cost will be purely quadratic,

$$\frac{d^2C}{d\tau^2} = g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau} = \text{constant}, \tag{2.39}$$

which implies that the first derivative of the cost will be linear in τ:

$$\frac{dC}{d\tau} = \left(g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}\right)\tau + \dot{C}(\tau = 0). \tag{2.40}$$

A knowledge of $\dot{C}(\tau = 0)$ will then tell us how far the geodesic needs to be integrated:

$$\tau_{max} = -\frac{\dot{C}(\tau = 0)}{g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}}. \tag{2.41}$$

We can calculate the missing piece of Eq. (2.41) from the chain rule and Eq. (2.38),

$$\begin{aligned} \dot{C} &= \frac{d\theta^\mu}{d\tau}\partial_\mu C \\ &= -g^{\mu\nu}\partial_\nu C\ \partial_\mu C, \end{aligned}$$

which gives us

$$\tau_{max} = 1.$$

The simplest method one could apply to solve the geodesic equation would be to apply a single Euler step, which moves the initial parameter guess by

$$\begin{aligned} \delta\theta^\mu &= \dot{\theta}^\mu \delta\tau \\ &= -g^{\mu\nu}\partial_\nu C, \end{aligned} \tag{2.42}$$

since $\delta\tau = 1$. Iteratively updating the parameters according to Eq. (2.42) is known as the Gauss-Newton method. It can be derived without geometric considerations by simply assuming a linear approximation to the residuals. Unless the initial guess is very good, however, the appearance of the inverse Hessian in Eq. (2.42) (with its enormous eigenvalues along sloppy directions) will result in large, unreliable steps and prevent the algorithm from converging.

The Gauss-Newton method needs some way to shorten its steps. Motivated by the idea of integrating a differential equation, one could imagine taking several Euler steps instead of one. If one chooses a time step to minimize the cost along the line given by the local Gauss-Newton direction, then the algorithm is known as the modified Gauss-Newton method, which is a much more stable algorithm than the simple Gauss-Newton method [51].

One could also imagine performing some more sophisticated method, such as a Runge-Kutta method. The problem with these approaches is that the sloppy eigenvalues of the inverse metric require the Euler or Runge-Kutta steps to be far too small be competitive with other algorithms. In practice, these techniques are not as effective as the Levenberg-Marquardt algorithm, discussed in the next section.

2.9.2 Levenberg-Marquardt Algorithm

The algorithm that steps according to Eq. (2.42) using the metric of the model graph, Eq. (2.17), is known as the Levenberg-Marquardt step:

$$\delta\theta^\mu = -\left(g^0 + \lambda D\right)^{\mu\nu} \partial_\nu C.$$

If D is chosen to be the identity, then the algorithm is the Levenberg algorithm [64]. The Levenberg algorithm is simply the Gauss-Newton method on the model graph instead of the model manifold.

If D is chosen to be a diagonal matrix with entries equal to the diagonal elements of g^0, then the algorithm is the Levenberg-Marquardt algorithm [69]. As we mentioned in section 2.5, the Levenberg-Marquardt algorithm, using the Marquardt metric, is invariant to rescaling the parameters. We find this property to often be counterproductive to the optimization process since it prevents the modeler from imposing the proper scale for the parameter values. In addition we observe that the resulting algorithm is more prone to parameter evaporation. The purpose for adding D to the metric is to *introduce* parameter dependence to the step direction.

The Levenberg-Marquardt algorithm adjusts λ at each step. Typically, when the algorithm has just begun, the Levenberg-Marquardt term will be very large, which will force the algorithm to take small steps in the gradient direction. Later, once the algorithm has descended into a canyon, λ will be lowered, allowing the algorithm to step in the Gauss-Newton direction and follow the length of the canyon. The Levenberg-Marquardt parameter, therefore, serves the dual function of rotating the step direction from the Gauss-Newton direction to the gradient direction, as well as shortening the step.

As we mentioned in section 2.5, when using the Levenberg metric, λ will essentially wash out all the sloppy eigenvalues of the original metric and leave the large ones unaffected. The relatively large multiplicative factor separating eigenvalues means that λ does not need to be finely tuned in order to achieve convergence. Nevertheless, an efficient method for choosing λ is the primary way that the Levenberg-Marquardt algorithm can be optimized. We discuss two common updating schemes here.

A typical method of choosing λ at each step is described in Numerical Recipes [83]. One picks an initial value, say $\lambda = .001$, and tries the proposed step. If the step moves to a point of larger cost, by default, the step is rejected and λ is increased by some factor, 10. If the step has decreased the cost, the step is accepted and λ is decreased by a factor of 10. This method is guaranteed to eventually produce an acceptable step, since for extremely large values of λ, the method will take an arbitrarily small step in the gradient direction. We refer to this as the traditional scheme for updating λ.

A more complicated method of choosing λ is based on a trust region approach and is described in [72]. As in the previous updating scheme, at each step λ is increased until the step goes downhill (all uphill steps are rejected). However, after an accepted step, the algorithm compares the decrease in cost at the new position with the decrease predicted by the linear approximation of the residuals

$$\frac{\|\vec{r}\,(\theta_{old})\| - \|\vec{r}\,(\theta_{new})\|}{\|\vec{r}\,(\theta_{old})\| - \left\|\vec{r}(\theta_{old}) + \vec{J}_\mu \delta\theta^\mu\right\|}.$$

If this value is very far from unity, then the algorithm has stepped beyond the region for which it trusts the linear approximation and will increase λ by some factor even though the cost has decreased; otherwise, λ is decreased. This method tunes λ so that most steps are accepted, reducing the number of extra function evaluations. As a result, it often needs a few more steps, and therefore, a few more Jacobian evaluations. This algorithm works well for small problems where the computational complexity of the function and the Jacobian are comparable. It is not as competitive using the number of Jacobian evaluations as a measure of success.

These are certainly not the only update schemes available. Both of these criteria reject any move that increases the cost, which is a natural method to ensure that the algorithm does not drift to large costs and never converges. One could imagine devising an update scheme that allows some uphill steps in a controlled way such that the algorithm remains well-behaved. We consider such a scheme elsewhere [96] and note that it was a key inspiration for the Delayed Gratification update scheme that we describe below in section 2.9.3.

As we observed in section 2.6, the metric formed by the model graph acts similarly to the effect of adding linear Bayesian priors as residuals. The Levenberg-Marquardt algorithm therefore chooses a Gauss-Newton step as though there were such a prior, but then ignores the prior in calculating the cost at the new point. A similar algorithm, known as the iteratively updated Gauss-Newton algorithm, includes the contribution from the prior when calculating the new cost, although the strength of the prior may be updated at each step [6].

2.9.3 Delayed Gratification

We have seen that parameter-effects curvatures are typically several orders of magnitude larger than extrinsic curvatures for sloppy models, which means that the model manifold is much more flat than the non-linearities alone suggest and produce the concentric circles in Fig. 2.7. When considering only a single step on even a highly curved manifold, if the parameter-effects curvature dominates, the step size will be less than the (inverse) extrinsic curvature and approximating the manifold by a flat surface is a good approximation. Furthermore, we have seen that when the manifold is flat, geodesics are the paths that we hope to follow.

The Rosenbrock function is a well known test function for which the extended geodesic coordinates can be expressed analytically. It has a long, parabolic shaped canyon and is given by

$$\begin{aligned} r_1 &= 1 - \theta_1 \\ r_2 &= A\left(\theta_2 - \theta_1^2\right), \end{aligned}$$

where A is a parameter that controls the narrowness of the canyon. The Rosenbrock function has a single minimum at $(\theta_1, \theta_2) = (1, 1)$. Since there are two residuals and two parameters, the model manifold is flat and the extended geodesic coordinates are the residuals. It is straightforward to solve

$$\begin{aligned} \theta_1 &= 1 - r_1 \\ \theta_2 &= \frac{r_2}{A} + (1 - r_1)^2 . \end{aligned}$$

If we change to polar coordinates,

$$\begin{aligned} r_1 &= \rho \sin\phi \\ r_2 &= \rho \cos\phi, \end{aligned}$$

then lines of constant ϕ are the geodesic paths that we would like an algorithm to follow toward the best fit, and lines of constant ρ are cost contours. We plot both sets of curves in Fig. 2.25.

Inspecting the geodesic paths that lead to the best fit in Fig. 2.25 reveals that most of the path is spent following the canyon while decreasing the cost only slightly. This behavior is common to all geodesics in canyons such as this. We would like to devise an update scheme for λ in the Levenberg-Marquardt algorithm that will imitate this behavior. The results of section 2.8.6 suggest that we will often be able to step further than a trust region would allow, so we start from the traditional update scheme.

The primary feature of the geodesic path that we wish to imitate is that radial geodesics are nearly parallel to cost contours. In the usual update scheme, if a

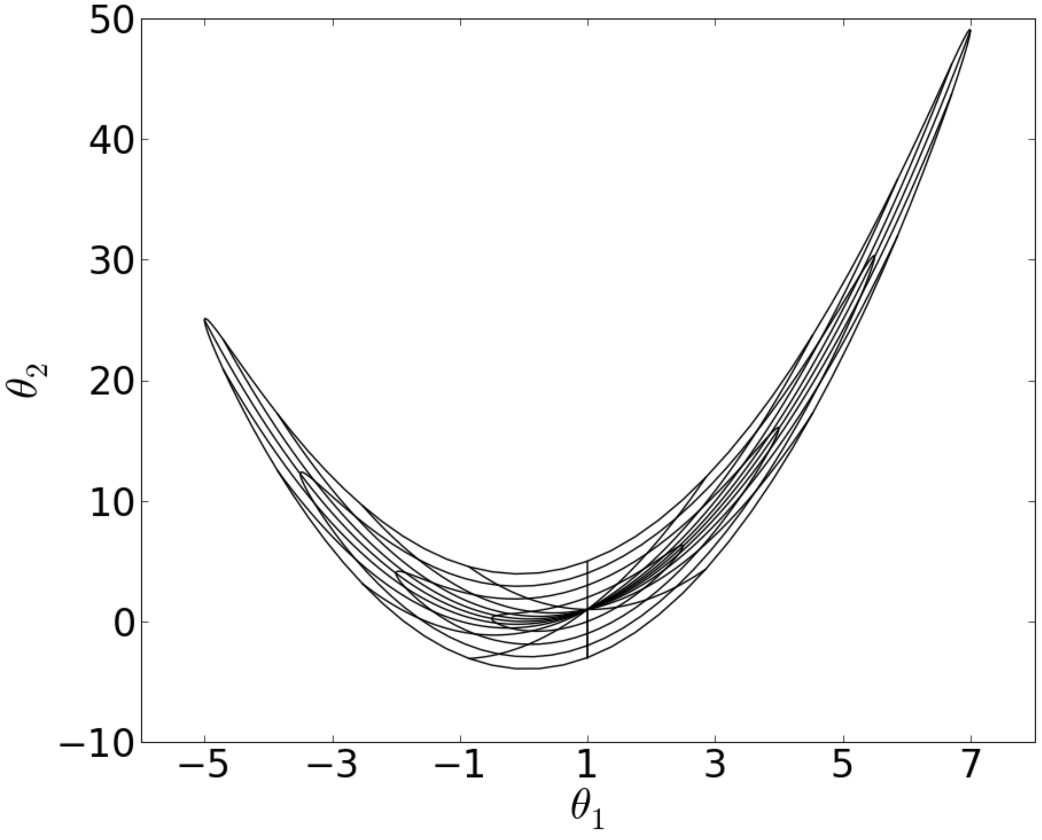


Figure 2.25: Geodesics of the Rosenbrock function

Extended Geodesic Coordinates for Rosenbrock Function. The residuals are one choice of extended geodesic coordinates if the number of parameters equal the number of data points, as is the case for the Rosenbrock function. Because the Rosenbrock function is a simple quadratic, the coordinate transformation can be expressed analytically. Lines of constant ρ are equi-cost lines, while lines of constant ϕ are the paths a geodesic algorithm should follow to the best fit. Because the geodesics follow the path of the narrow canyon, the radial geodesics are nearly parallel to the equi-cost lines in parameter space. This effect is actually much more extreme than it appears in this figure because of the relative scales of the two axes.

proposed step moves uphill, then λ is increased. In the spirit of following a cost contour, one could slowly increase the Levenberg-Marquardt parameter just until the cost no longer increases. If λ is fine-tuned until the cost is the same, we call this the equi-cost update scheme. Such a scheme would naturally require many function evaluations for each step, but as we said before, we are primarily interested in problems for which function calls are cheap compared to Jacobian evaluations. Even so, determining λ to this precision is usually overkill, and the desired effect can be had by a much simpler method.

Instead of precisely tuning λ, we modify the traditional scheme to raise and lower the parameter by different amounts. Increasing λ by very small amounts when a proposed step is uphill and then decreasing it by a large amount when a downhill step is finally found will mimic the desired behavior. We have found that increasing by a factor of 2 and decreasing by a factor of 10 works well, consistent with Lampton's results [63]. We call this method, the *delayed gratification* update scheme.

The reason that this update scheme is effective is due to the restriction that we do not allow uphill steps. If we move downhill as much as possible in the first few steps, we greatly restrict the steps that will be allowed as successive iterations, slowing down the convergence rate, as illustrated in Fig. 2.26.

By using the delayed gratification update scheme, we are using the smallest value of λ that does not produce an uphill step. If we choose a trust-region method, instead, each step will choose a much larger value of λ. The problem with using larger values of λ at each step, is that they drive the algorithm downhill prematurely. Even if the trust region only cuts each possible step in half compared to the delayed gratification scheme, the cumulative effect will be much more damaging

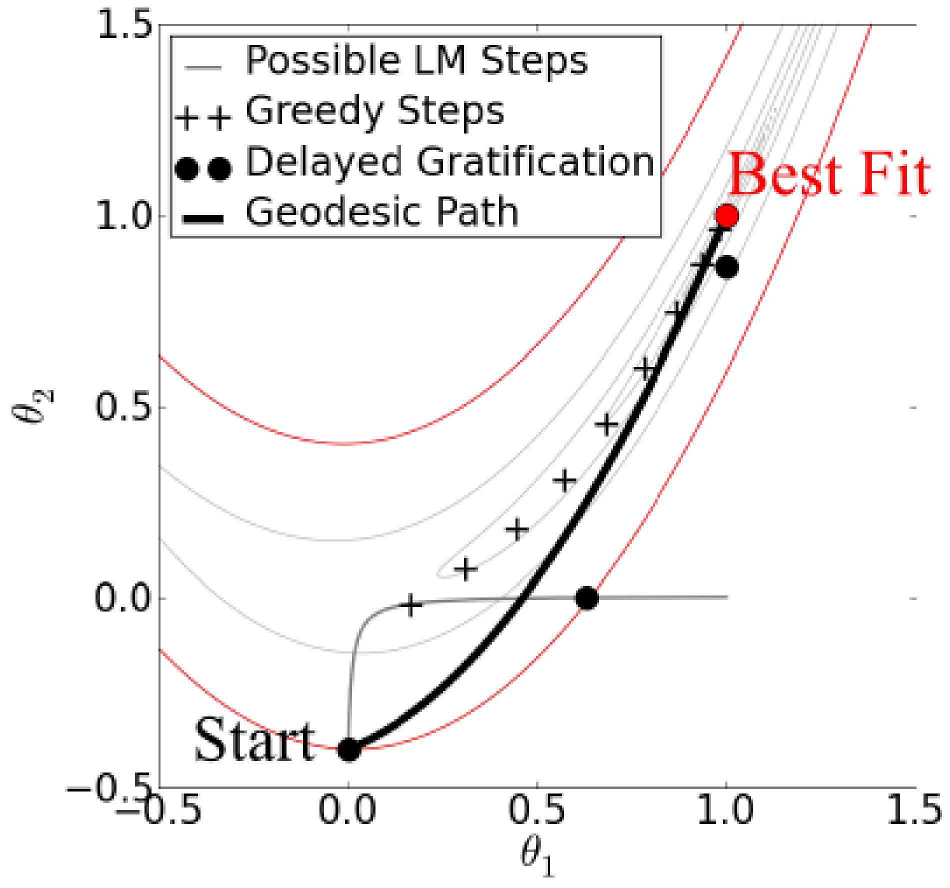


Figure 2.26: Delayed gratification and greedy steps
(Color online) **Greedy Step and Delayed Gratification Step Criterion.** In optimization problems for which there is a long narrow canyon, such as for the Rosenbrock function, choosing a delayed gratification step is important to optimize convergence. By varying the damping term, the algorithm may choose from several possible steps. A greedy step will lower the cost as much as possible, but by so doing will limit the size of future steps. An algorithm that takes the largest allowable step size (without moving uphill) will not decrease the cost much initially, but will arrive at the best fit in fewer steps and more closely approximate the true geodesic path. What constitutes the largest tolerable step size should be optimized for specific problems so as to guarantee convergence.

because of how this strategy reduces the possibility of future steps.

2.9.4 Geodesic Acceleration

We have seen that a geodesic is a natural path that an algorithm should follow in its search for the best fit. The application of geodesics to optimization algorithms is not new. It has been applied, for example to the problem of nonlinear programming with constraints [65, 81], to neural network training [54], and to the general problem of optimization on manifolds [78, 1]. Here we apply it as a second order correction to the Levenberg-Marquardt step.

The geodesic equation is a second order differential equation, whose solution we have attempted to mimic by only calculating first derivatives of the residuals (Jacobians) and following a delayed gratification stepping scheme. From a single residual and Jacobian evaluation, an algorithm can calculate the gradient of the cost as well as the metric, which determines a direction. We would like to add a second order correction to the step, but one would expect its evaluation to require a knowledge of the second derivative matrix, which would be even more expensive to calculate than the Jacobian. We have already noted that most of the computer time is spent on Jacobian evaluations, so second order steps would have even more overhead. Fortunately, the second order correction to the geodesic path can be calculated relatively cheaply in comparison to a Jacobian evaluation.

The second order correction, or acceleration, to the geodesic path is given by

$$a^\mu = -\Gamma^\mu_{\alpha\beta} v^\alpha v^\beta, \tag{2.43}$$

as one can see by inspecting Eq. (2.24). In the expression for the acceleration, the velocity contracts with the two lower indices of the connection. Recall from the definition,

$$\Gamma^{\mu}_{\alpha\beta} = g^{\mu\nu} \partial_\nu r_m \partial_\alpha \partial_\beta r_m,$$

that the lowered indices correspond to the second derivatives of the residuals. This means that the acceleration only requires a directional second derivative in the direction of the velocity. This directional derivative can be estimated with two residual evaluations in addition to the Jacobian evaluation. Since each step will always call at least one residual evaluation, we can estimate the acceleration with only one additional residuals call, which is very cheap computationally compared to a Jacobian evaluation.

With an easily evaluated approximation for the acceleration, we can then consider the trajectory given by

$$\delta\theta^\mu = \dot{\theta}^\mu \delta\tau + \frac{1}{2}\ddot{\theta}^\mu \delta\tau^2. \tag{2.44}$$

By following the winding canyon with a parabolic path instead of a linear path, we expect to require fewer steps to arrive at the best fit. The parabola can more naturally curve around the corners of the canyon than the straight line path. This is illustrated for the Rosenbrock function in Fig. 2.27. Because the canyon of the Rosenbrock function is parabolic, it can be traversed exactly to the best fit by the acceleration in a single step.

The relationship between the velocity and the acceleration depicted in Fig. 2.27 for the Rosenbrock function is overly idealized. In general the velocity and the acceleration will not be perpendicular; in fact, it is much more common for them to be nearly parallel or anti-parallel. Notice that the expression for the connection coefficient involves a factor of the inverse metric, which will tend to bias the acceleration to align parallel to the sloppy directions, just as it does for the velocity. It is much more common for the acceleration to point in the direction opposite to

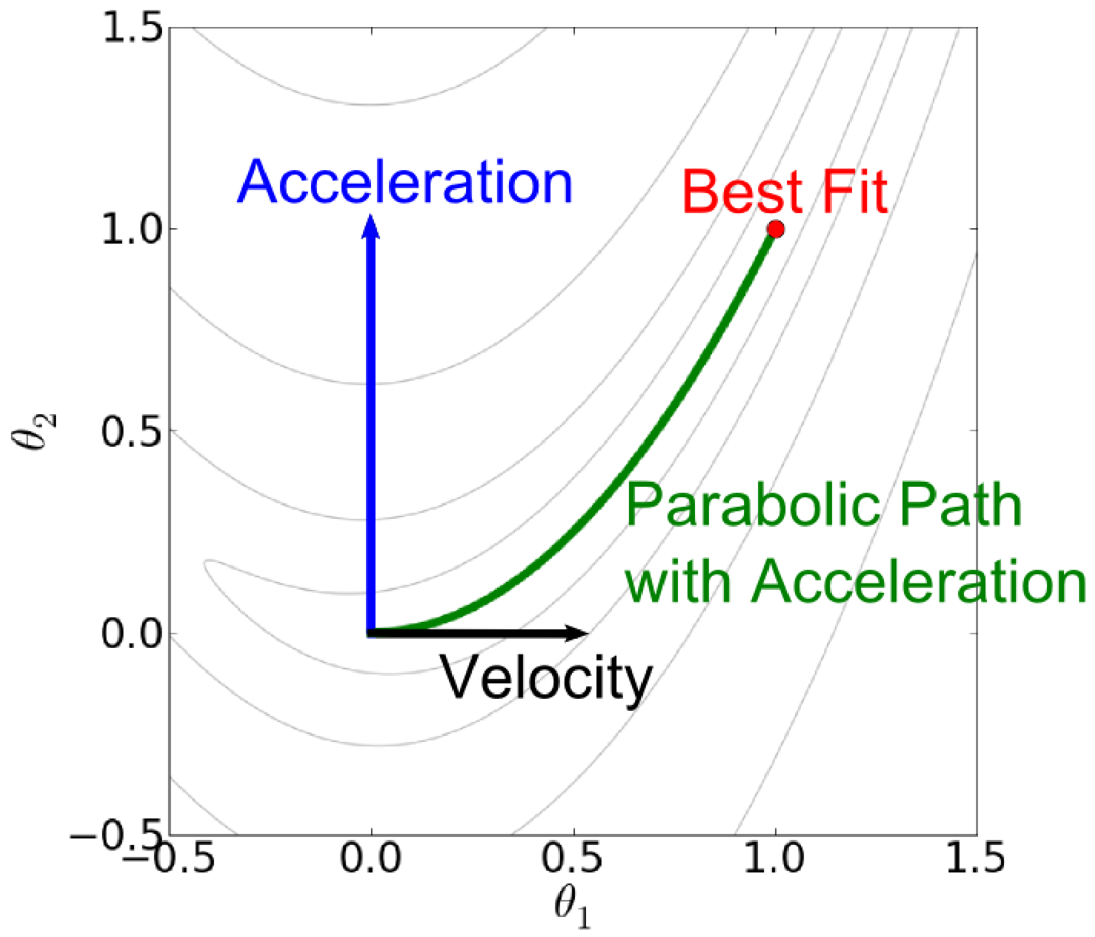
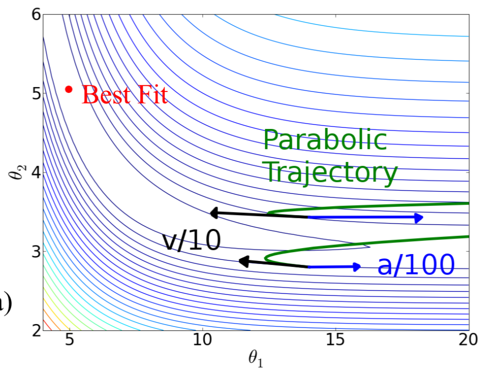


Figure 2.27: Geodesic acceleration in the Rosenbrock Valley
(Color online) **Geodesic Acceleration in the Rosenbrock Valley.** The Gauss-Newton direction, or velocity vector, gives the correct direction that one should move to approach the best fit while navigating a canyon. However, that direction quickly rotates, requiring an algorithm to take very small steps in order to avoid uphill moves. The geodesic acceleration indicates the direction in which the velocity rotates. The geodesic acceleration determines a parabolic trajectory that can efficiently navigate the valley without running up the wall. The linear trajectory quickly runs up the side of the canyon wall.

a)



b)

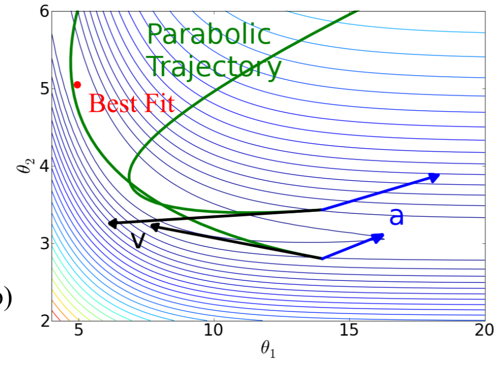


Figure 2.28: Effect of damping on geodesic acceleration
(Color online) a) **De-acceleration when overstepping.** Typically the velocity vector greatly overestimates the proper step size. (We have rescaled both velocity and acceleration to fit in the figure.) Algebraically, this is due to the factor of the inverse metric in the velocity, which has very large eigenvalues. The acceleration compensates for this by pointing anti-parallel to the velocity. However, the acceleration vector is also very large, as it is multiplied twice by the velocity vector and once by the inverse metric.To make effective use of the acceleration, it is necessary to regularize the metric by including a damping term. b) **Acceleration indicating the direction of the canyon.** As the Levenberg-Marquardt parameter is raised, the velocity vector shortens and rotates from the natural gradient into the downhill direction. The acceleration vector also shortens, although much more rapidly, and also rotates. In this two dimensional cross section, although the two velocity vectors rotate in opposite directions, the accelerations both rotate to indicate the direction that the canyon is turning. By considering the path that one would optimally like to take (along the canyon), it is clear that the acceleration vector is properly indicating the correction to the desired trajectory.

the velocity, as for a summing exponentials model in Fig. 2.28a.

Although an acceleration that is anti-parallel to the velocity may seem worthless, it is actually telling us something useful: our proposed step was too large. As we regulate the velocity by increasing the Levenberg-Marquardt parameter, we also regulate the acceleration. Once our velocity term is comparable to the distance over which the canyon begins to curve, the acceleration indicates into which direction the canyon is curving, as in Fig. 2.28b.

If the damping term is too small, the acceleration points in the opposite direction to and is much larger than the velocity. This scenario is dangerous because it may cause the algorithm to move in precisely the opposite direction to the Gauss-Newton direction, causing parameter evaporation. To fix this problem, we add another criterion for an acceptable step. We want the contribution from the acceleration to be smaller than the contribution from the velocity; therefore, we typically reject proposed steps, increasing the Leveberg-Marquardt parameter until

$$\frac{\sqrt{\sum (a^\mu)^2}}{\sqrt{\sum (v^\mu)^2}} < \alpha, \tag{2.45}$$

where α is a chosen parameter, typically unity, although for some problems a smaller value is required.

The acceleration is likely to be most useful when the canyon is very narrow. As the canyon narrows, the allowed steps become smaller. In essence, the narrowness of the canyon is determining to what accuracy we are solving the geodesic equation. If the canyon requires a very high accuracy, then a second order algorithm is likely to converge much more quickly than a first order algorithm. We will see this explicitly in the next section when we compare algorithms.

We have argued repeatedly that for sloppy models whose parameter-effects curvature are dominant, a geodesic is the path that an algorithm should follow. One could object to this assertion on the grounds that, apart from choosing the initial direction of the geodesic to be the Gauss-Newton direction, there is no reference to the cost gradient in the geodesic equation. If a manifold is curved, then the geodesic will not lead directly to the best fit. In particular, the acceleration is independent of the data.

Instead of a geodesic, one could argue that the path that one should follow is given by the first order differential equation

$$v^\mu = \frac{-g^{\mu\nu}\nabla_\nu C}{\sqrt{g^{\alpha\beta}\nabla_\alpha C \ \nabla_\beta C}}, \tag{2.46}$$

where we have introduced the denominator to preserve the norm of the tangent vector. Each Levenberg-Marquardt step chooses a direction in the Gauss-Newton direction on the model graph, which seems to be better described by Eq. (2.46) than by the geodesic equation, Eq. (2.24). In fact Eq. (2.46) has been proposed as a Neural Network training algorithm by Amari et al. [2].

The second order differential equation corresponding to Eq. (2.46) which can be found by taking the second derivative of the parameters, is a very complicated expression. However, if one then applies the approximation that all non-linearities are parameter-effects curvature, the resulting differential equation is exactly the geodesic equation. By comparing step sizes with inverse curvatures in Fig. 2.23, we can see that over a distance of several steps, the approximation that all non-linearities are parameter-effects curvature should be very good. In such a case, the deviation of Eq. (2.46) from Eq. (2.24) will not be significant over a few steps.

While the tensor analysis behind this result is long and tedious, the geometric meaning is simple and intuitive: if steps are much smaller than the extrinsic curvature on the surface, then the vector (in data space) corresponding to the Gauss-Newton direction can parallel transport itself to find the Gauss-Newton direction at the next point. That is to say the direction of the tangent vector of a geodesic does not change if the manifold is extrinsically flat.

Including second derivative information in an algorithm is not new. Newton's method, for example replaces the approximate Hessian of the Gauss-Newton method in Eq. (2.5), with the full Hessian in Eq. (2.4). Many standard algorithms seek to efficiently find the actual Hessian, either by calculating it directly or by estimation [43, 83]. One such algorithm, which we use for comparison in the next section, is a quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shannon (BFGS) [79], which estimates the second derivative from an accumulation of Jacobian evaluations at each step.

In contrast to these Newton-like algorithms, the geodesic acceleration is not an attempt to better approximate the Hessian. The results of section 2.7 suggest that the approximate Hessian is very good. Instead of correcting the error in the size and direction of the ellipses around the best fit, it is more productive to account for how they are bent by non-linearities, which is the role of the geodesic acceleration. The geodesic acceleration is a *cubic* correction to the Levenberg-Marquardt step.

There are certainly problems for which a quasi-Newton algorithm will make important corrections to the approximate Hessian. However, we have argued that sloppy models represent a large class of problems for which the Newton correction is negligible compared to that of the geodesic acceleration. We demonstrate this numerically with several examples in the next section.

2.9.5 Algorithm Comparisons

To demonstrate the effectiveness of an algorithm that uses delayed gratification and the geodesic acceleration, we apply it to a few test problems that highlight the typical difficulties associated with fitting by least squares.

First, consider a generalized Rosenbrock function,

$$C = \frac{1}{2}\left(\theta_1^2 + A^2\left(\theta_2 - \frac{\theta_1^n}{n}\right)^2\right),$$

where A and n are not optimizable parameters but set to control the difficulty of

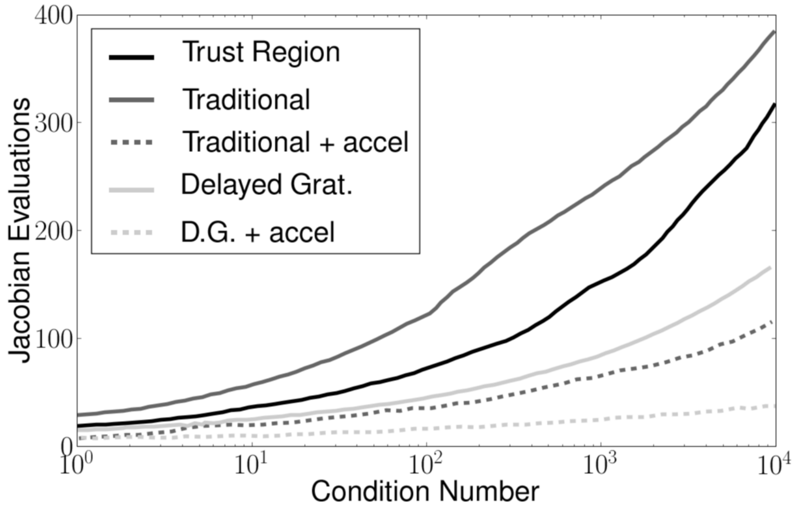
Algorithm	Success Rate	Mean NJEV	Mean NFEV
Trust Region LM	12%	1517	1649
Traditional LM	33%	2002	4003
Traditional LM + accel	65%	258	1494
Delayed Gratification	26%	1998	8625
Delayed Gratification + accel	65%	163	1913
BFGS	8%	5363	5365

Table 2.2: Algorithm performance on an exponential model
The results of several algorithms applied to a test problem of fitting a sum of four exponential terms (varying both rates and amplitudes – 8 parameters) in log-parameters (to enforce positivity). Initial conditions are chosen near a manifold boundary with a best fit of zero cost near the center of the manifold. Among successful attempts, we further compare the average number of Jacobian and function evaluations needed to arrive at the fit. Success rate indicates an algorithm's ability to avoid the manifold boundaries (find the canyon from the plateau), while the number of Jacobian and function evaluations indicate how efficiently it can follow the canyon to the best fit. BFGS is a quasi newton scalar minimizer of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) [79, 57]. The traditional [69, 83] and trust region [72] implementations of Levenberg-Marquardt consistently outperform this and other general optimization routines on least squares problems, such as Powell, simplex, and conjugate gradient. Including the geodesic acceleration on a standard variant of Levenberg-Marquardt dramatically increases the success rate while decreasing the computation time.

the problem. This problem has a global minimum of zero cost at the origin, with a canyon following the polynomial path θ_1^n/n whose width is determined by A. To compare algorithms we draw initial points from a Gaussian distribution centered at $(1, 1/n)$ with standard deviation of unity, and compare the average number of Jacobian evaluations an algorithm requires in order to decrease the cost to 10^{-4}. The results for the cubic and quartic versions of the problem are given in Fig. 2.29 for several version of the the Levenberg-Marquardt algorithm.

We next consider a summing exponential problem; a summary of these results can be found in [94]. Here we expand it to include the delayed gratification algorithm outlined above in section 2.9.3.

a)



b)

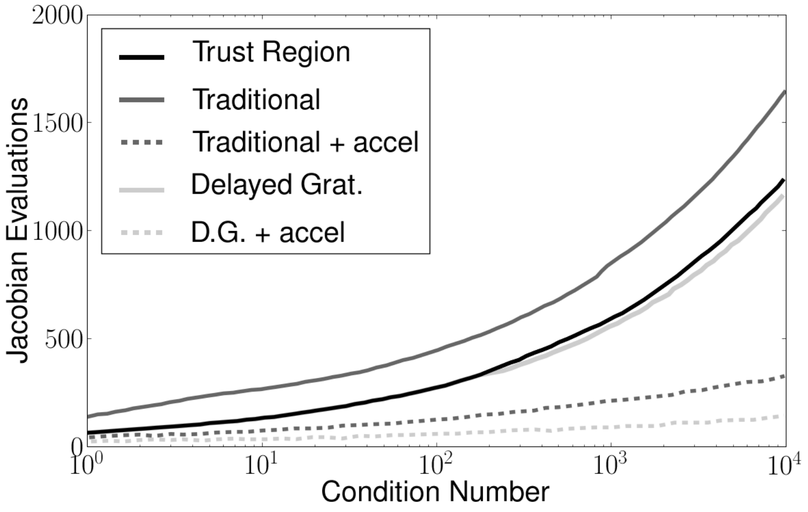


Figure 2.29: Algorithm performance on the Rosenbrock function
Generalized Rosenbrock results for Levenberg-Marquardt variants. If the canyon that an algorithm must follow is very narrow (measured by the condition number of the metric at the best fit) or turns sharply, the algorithm will require more steps to arrive at the best fit. Those that use the geodesic acceleration term converge more quickly as the canyon narrows. As the parameter-effects curvature increases, the canyon becomes more curved and the problem is more difficult. Notice that changing the canyon's path from a cubic function in a) to a quartic function in b) slowed the convergence rate by a factor of 5. We have omitted the quadratic path since including the acceleration allows the algorithm to find the best fit in one step, regardless of how narrow the canyon becomes.

A surprising result from Table 2.2 is that including the geodesic acceleration not only improves the speed of convergence, but improves the likelihood of convergence, that is, the algorithm is less likely to evaporate parameters. This is a consequence of the modified acceptance criterion in Eq. (2.45). As an algorithm evaporates parameters, it approaches a singular point of the metric on the model manifold, causing the velocity vector in parameter space to diverge. The acceleration, however, also diverges, but much more rapidly than the velocity. By requiring the acceleration term to be smaller than the velocity, the algorithm is much more adept at avoiding boundaries. Geodesic acceleration, therefore, helps to improve both the initial search for the canyon from the plateau, as well as the subsequent race along the canyon to the best fit.

Finally, we emphasize that the purpose of this section was to demonstrate that delayed gratification and geodesic acceleration are potentially helpful modifications to existing algorithms. The results presented in this section do not constitute a rigorous comparison, as such a study would require a much broader sampling of test problems. Instead, we have argued that ideas from differential geometry can be helpful to speed up the fitting process if existing algorithms are sluggish. We are in the process of performing a more extensive comparison whose results will appear shortly [96].

2.10 Conclusions

A goal of this paper has been to use a geometric perspective to study nonlinear least squares models, deriving the relevant metric, connection, and measures of curvature, and to show that geometry provides useful insights into the difficulties associated with optimization.

We have presented the model manifold and noted that it typically has boundaries, which explain the phenomenon of parameter evaporation in the optimization process. As algorithms run into the manifold's boundaries, parameters are pushed to infinite or otherwise unphysical values. For sloppy models, the manifold is bounded by a hierarchy of progressively narrow boundaries, corresponding to the less responsive direction of parameter space. The model behavior spans a hyper-ribbon in data space. This phenomenon of geometric sloppiness is one of the key reasons that sloppy models are difficult to optimize. We provide a theoretical caricature of the model manifold characterizing their geometric series of widths, extrinsic curvatures, and parameter-effects curvatures. Using this caricature, we estimate the number of evaporated parameters one might expect to find at the best fit for a given uncertainty in the data.

The model graph removes the boundaries and helps to keep the parameters at reasonable levels. This is not always sufficient, however, and we suggest that in many cases, the addition of thoughtful priors to the cost function can be a significant help to algorithms.

The second difficulty in optimizing sloppy models is that the model parameters are far removed from the model behavior. Because most sloppy models are dominated by parameter-effects curvature, if one could reparametrize the model with extended geodesic coordinates, the long narrow canyons would be transformed to one isotropic quadratic basin. Optimizing a problem in extended geodesic coordinates would be a trivial task!

Inspired by the motion of geodesics in the curved valleys, we developed the delayed gratification update scheme for the traditional Levenberg-Marquardt algorithm and further suggest the addition of a geodesic acceleration term. We have seen that when algorithms must follow long narrow canyons, these can give significant improvement to the optimization algorithm. We believe that the relative cheap computational cost of adding the geodesic acceleration to the Levenberg-Marquardt step gives it the potential to be a robust, general-purpose optimization algorithm, particularly for high dimensional problems. It is necessary to explore the behavior of geodesic acceleration on a larger problem set to justify this conjecture [96].

Acknowledgments

We would like to thank Saul Teukolsky, Eric Siggia, John Guckenheimer, Cyrus Umrigar, Peter Nightingale, Stefanos Papanikolou, Bryan Daniels, and Yoav Kallus for helpful discussions. We thank Dave Schneider for suggesting information geometry to us and acknowledge support from NSF grants number DMR-0705167 and DMR-1005479.

CHAPTER 3

IMPROVEMENTS TO THE LEVENBERG-MARQUARDT ALGORITHM

3.1 Abstract

We introduce several improvements to the Levenberg-Marquardt algorithm for nonlinear least-squares minimization in order to improve both its convergence speed and robustness to initial parameter guesses. We update the usual step to include a geodesic acceleration correction term, introduce a systematic way of accepting steps that increase the residual sum of squares, and employ the Broyden method to update the Jacobian matrix. We test these changes by comparing their performance on a number of test problems with standard implementations of the algorithm.

3.2 Introduction

A common computational problem is that of minimizing a sum of squares

$$C(\theta) = \frac{1}{2}\sum_{m=1}^{M} r_m(\theta)^2, \tag{3.1}$$

where $r : \mathbf{R}^N \to \mathbf{R}^M$ is an M-dimensional nonlinear vector function of N parameters, θ, where $M \geq N$. The Levenberg-Marquardt algorithm is perhaps the most common method for nonlinear least-squares minimization. This paper discusses a number of modifications to the Levenberg-Marquardt algorithm designed to improve both its success rate and convergence speed. These modifications are likely to be most useful on large problems with many parameters, where the usual Levenberg-Marquardt routine often has difficulty.

Least-Squares minimization is most often used in data fitting, in which case the function $r_m(\theta)$ takes the form

$$r_m(\theta) = \frac{y(t_m, \theta) - y_m}{\sigma_m}, \tag{3.2}$$

where $y(t, \theta)$ is a model of the observed data, y_m, that depends on a set of unknown parameter, θ and one or more independent variables t. The deviation of model from observation is weighted by the relative uncertainty in observed data, σ. The terms in Eq. (3.2) are known as the residuals and may be augmented by additional terms representing Bayesian prior information about the expected values of θ. We refer to the function in Eq. (3.1) as the cost function. The cost corresponds to the negative log-likelihood of parameter values given the data assuming gaussian errors. The parameter values that minimize $C(\theta)$ are known as the best fit parameters.

Often, models with many parameters exhibit universal characteristics known as sloppiness. The behavior of sloppy models is determined by only a few stiff (relevant) parameter combinations, while most other parameter combinations are sloppy (irrelevant)[19, 18, 74, 99, 48]. Fitting difficulties arise when algorithms are lost in regions of parameter space where the model behavior is insensitive to changes in the parameters, i.e. a plateau on the cost surface in parameter space as in Fig. 3.1. A common occurance is that while lost on the plateau, algorithms push parameters to infinite values without finding a good fit, a phenomenon known as parameter evaporation[94, 95]. The algorithm must then be guided by hand in order to converge.

In addition to parameter evaporation, algorithms become sluggish when they must follow a narrow canyon to find the best fit, as in Fig. 3.1. It is not uncommon

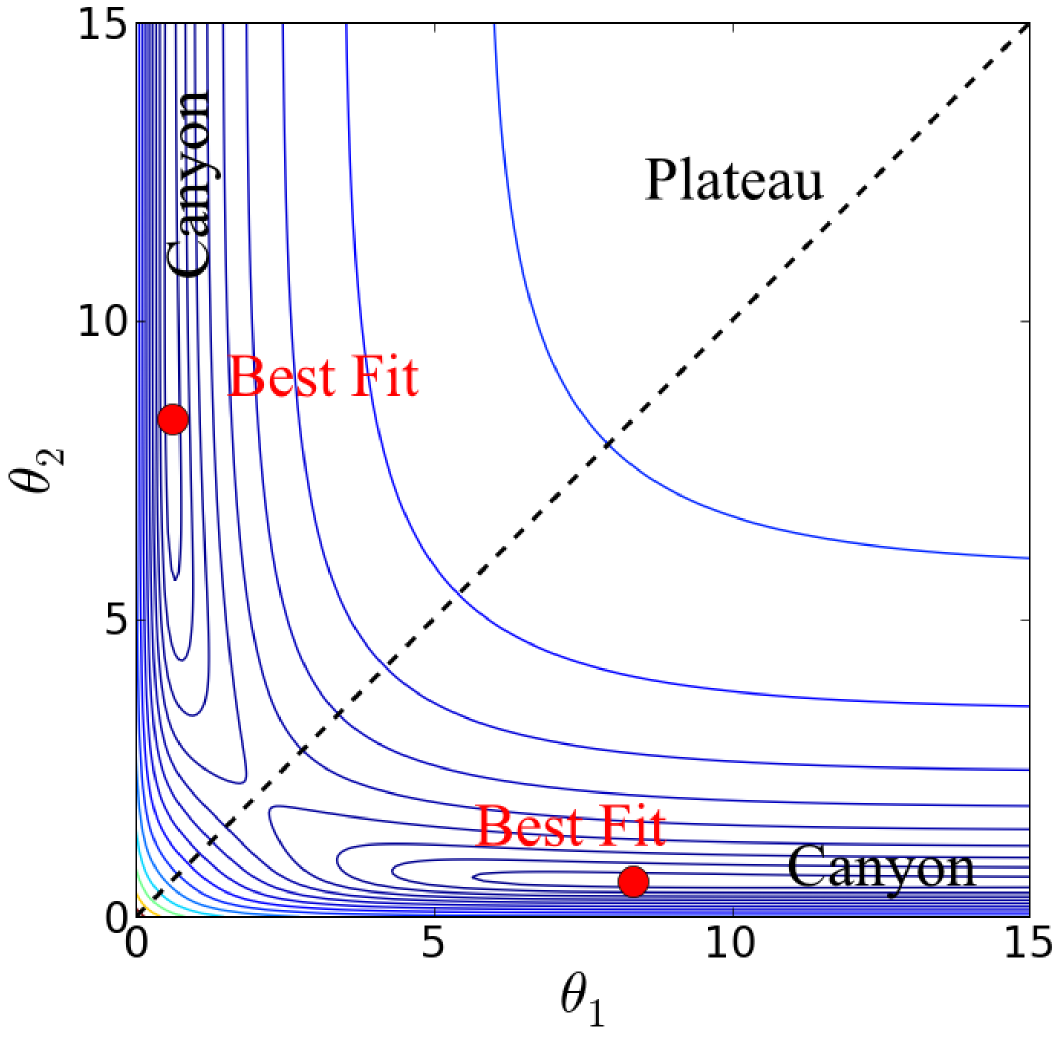


Figure 3.1: The cost surface in parameter space for least-squares problems often forms a hierarchy of narrow, winding canyons surrounded by plateaus. Algorithms are easily lost on the plateaus, often evaporating parameters (pushing them to infinity) while searching for a canyon. Having found the canyon, algorithms can become sluggish while following it to the best fit. This simple model, $y = e^{-\theta_1 t} + e^{-\theta_2 t}$, fit to three data points has a plateau when the parameters become very large. It also exhibits a symmetry when parameter are permuted.

for the aspect ratio of the canyon be 1000 : 1 or more for problems with ten or more parameters[99, 48], which requires the algorithm to take very small steps as it inches along the bottom of the trough.

The difficulty in data fitting is exacerbated by the fact that solutions to the two principal problems (parameter evaporation and slow convergence) are often at odds with one another. Methods which speed convergence in the canyon usually increase the probability of parameter evaporation and vice versa.

The Levenberg-Marquardt algorithm[64, 69, 72, 83] is particularly well-suited to deal with the challenges of least-squares minimization. The algorithm is a modification of the Gauss-Newton method. The Gauss-Newton method is based on a local linearization of the residuals

$$r_m(\theta + \delta\theta) \approx r_m(\theta) + J_{m\mu}\delta\theta^{\mu}, \tag{3.3}$$

where J is the Jacobian matrix $J_{m\mu} = \partial r_m / \partial \theta_\mu$. The Gauss-Newton method then iterates according to

$$\delta\theta = -(J^T J)^{-1}\nabla C = -(J^T J)^{-1} J^T r. \tag{3.4}$$

The Gauss-Newton method will usually converge quickly if it begins sufficiently near a minimum of C. However, the matrix $J^T J$ is often ill-conditioned, with eigenvalues often spanning a range of six orders of magnitude or more. Therefore, unless the initial guess is very good, the Gauss-Newton method takes large, uncontrolled steps and will fail to converge.

To remedy the shortcomings of the Gauss-Newton method, Levenberg and Marquardt each suggested damping the $J^T J$ matrix by a diagonal cuttoff[64, 69]. The Levenberg-Marquardt algorithm therefore steps according to

$$\delta\theta = -\left(J^T J + \lambda D^T D\right)^{-1}\nabla C. \tag{3.5}$$

where $D^T D$ is a positive-definite, diagonal matrix representing the relative scaling of the parameters and λ is a damping parameter adjusted by the algorithm. When

λ is large, the method takes a small step in the gradient direction, well-suited for avoiding parameter evaporation. As the method nears a solution, λ is chosen to be small and the method converges quickly via the Gauss-Newton method.

Because it is can tune λ as needed, the Levenberg-Marquardt method is well-suited for dealing with the difficulties in nonlinear least-squares minimization. By properly adjusting the damping term, the method can interpolate between gradient descent, for avoiding parameter evaporation, and the Gauss-Newton algorithm for quickly converging along a canyon. One of the challenges for the Levenberg-Marquardt method is in choosing a suitable scheme for updating the damping parameter λ that successfully interpolates between the two regimes. Many such schemes exist and although some are more suited for avoiding parameter evaporation and others are more adept at navigating the canyon, the Levenberg-Marquardt method is generally robust to the specific method used.

Its relative success notwithstanding, the Levenberg-Marquardt algorithm may still fail to converge if it begins far from a minimum and will converge slowly if it must inch along the bottom of a canyon. Given the ubiquitous role of nonlinear least squares in mathematical modeling, and considering the trend to use increasingly large and computationally expensive models in all areas of science and engineering, any improvements that could be made to the Levenberg-Marquardt algorithm would be welcome. In this paper we discuss several such improvements.

In section 3.3 we discuss an alternative interpretation of the Levenberg-Marquardt algorithm in terms of an approximation to gradient flow on a manifold and summarize existing versions of the algorithm in section 3.4. We then explore how the existing methods can be improved by including the so-called geodesic acceleration in section 3.5 and a modified acceptance criterion in section 3.6. We then discuss how a rank-deficient update to the Jacobian matrix can reduce the number of times it must be evaluated in section 3.7. An open source implementation of the Levenberg-Marquardt algorithm with our proposed improvements is available in the FORTRAN and C languages.

In each of the following sections we compare the performance of the algorithm with the suggested improvements on a set of several test problems drawn from the Minpack-2 project[5] and the NIST statistical reference datasets[70]. Because most of these problems are of small or moderate size, it is unclear how performance on these problems extrapolates to performance on larger problems. We therefore also explore the algorithms' performance on several large test problems drawn from recent research in section. These problems and the comparison methods are summarized in B.1. We find that our proposed improvements consistently improve the performance of the algorithm on these problems.

3.3 Geometric Motivations

Because the function C is a sum of squares, it can be interpreted as a Euclidean distance which leads naturally to a geometric interpretation for the model. The key results of the geometric interpretation can be found in reference [95], which we summarize in the following paragraphs. As we will see, this geoemtric approach leads to an interpretation of the Levenberg-Marquardt algorithm as an approximation to geodesic flow on a manifold.

Consider the image of parameter space under the mapping $r : \mathbf{R}^N \to \mathbf{R}^M$. The mapping describes an N-dimensional manifold embedded in $\mathbf{R}^M$, known as the model manifold, $\mathcal{M}$ and has metric $J^T J$. The parameters θ act as coordinates

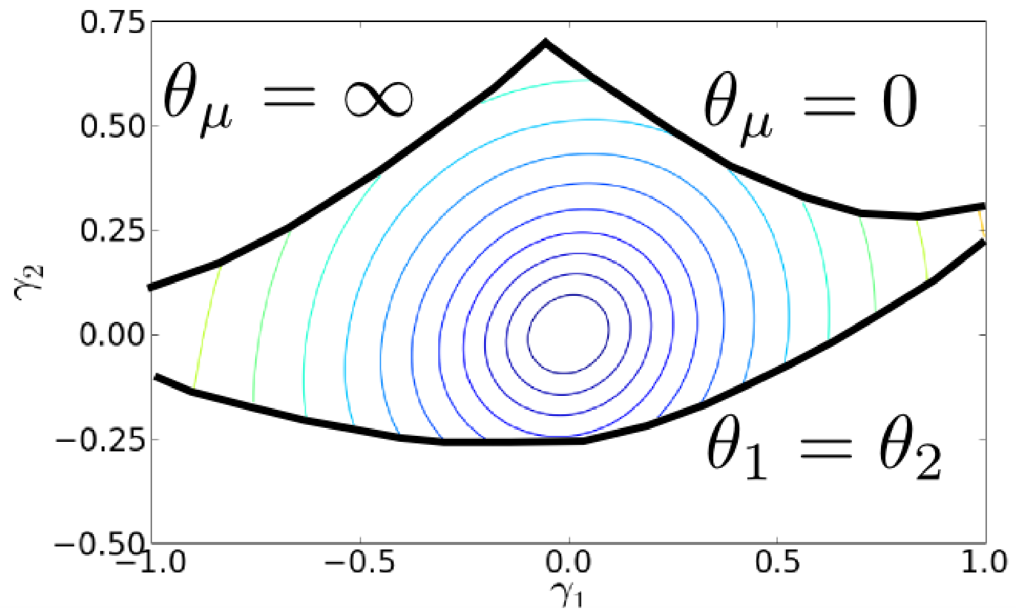


Figure 3.2: The cost surface in geodesic coordinates does not display the same hierarchy of canyons and plateaus typical of most parameterizations. Because of the relatively small extrinsic curvature on the model manifold, geodesics transform the surface into a single isotrpic, quadratic basin as it has here for the model in Fig. 3.1. A fitting problem in these coordinates would be relatively straightforward.

on $\mathcal{M}$ It has been shown that the problem of parameter evaporation is a result of algorithms running into the boundaries on $\mathcal{M}$, while sluggish performance is often the result of an awkward parameterization on the manifold.

Perhaps surprisingly, the model manifold usually has a small extrinsic curvature, which leads to the use of geodesics in the optimization process. In figure 3.1 we show contours of constant cost versus the model parameters. Figure 3.2 shows contours of constant cost for the same model in a new parameterization based on geodesic motion on the manifold. By using geodesics, the canyons and plateaus characteristic of the original parameterization have been transformed into a single isotropic, quadratic basin. In geodesic coordinates the Gauss-Newton method would have a much larger radius of convergence.

In general one cannot simply follow geodesics on $\mathcal{M}$ to minimize C. Because the boundaries on the model manifold often form very narrow widths, unless the initial guess is very close to the minimum, the geodesic will hit the boundary before converging. It is therefore natural to consider the manifold formed by plotting the output of r versus the parameters θ. The resulting manifold, known as the model graph, $\mathcal{G}$ is an N-dimensional manifold embedded in an $M + N$ dimensional space. The metric on the model graph is an interpolation between data space and parameter space metrics, and is given by $J^T J + \lambda D^T D$, where λ is a parameter expressing the relative weight of the parameter space portion of the embedding. The model graph, unlike the model manifold, has no boundaries at infinite parameter values and has a metric that is nowhere singular. By following geodesics on the model graph, an algorithm will be able to avoid the boundaries on $\mathcal{M}$ and be more robust to the initial parameter guess.

Motivated by these geometric ideas we posit that an algorithm should try to follow the path given by

$$\dot{\theta}^\mu = -\frac{g^{\mu\nu}\nabla_\nu C}{\sqrt{g^{\alpha\beta}\nabla_\alpha C \nabla_\beta C}}, \tag{3.6}$$

where $g^{\mu\nu} = (g^{-1})^{\mu\nu}$ is the inverse metric of the model graph $\mathcal{G}$. The path given by Eq. (3.6) corresponds to gradient descent on $\mathcal{G}$. The normalization is chosen such that the velocity in the embedding space has constant, unit norm; in this way the parameterization τ corresponds to the arc-length of the path in the embedding space. As long as the initial parameter guess does not correspond to an infinite cost, following the path given by Eq. (3.6) will lead to the best fit in a finite τ.

By assuming no extrinsic curvature on the model manifold, i.e. for $\lambda = 0$, the distance to the best fit is given by

$$\delta\tau = \sqrt{g^{\alpha\beta}\nabla_\alpha C \nabla_\beta C}. \tag{3.7}$$

Integrating Eq. (3.6) with a single Euler step will therefore, reproduce the Gauss-Newton method:

$$\delta\theta^\mu = -g^{\mu\nu}\nabla_\nu C. \tag{3.8}$$

Integrating with multiple Euler steps reproduces the modified Gauss-Newton method.

Euler steps on the model graph are equivalent to the Levenberg-Marquardt algorithm, which iterates according to

$$\delta\theta = -\left(J^T J + \lambda D^T D\right)^{-1} \nabla C. \tag{3.9}$$

Notice that the step size $|\delta\theta|$ decreases continuously to zero as λ tends to ∞. Thus, unlike the modified Gauss-Newton method, it is not necessary to to choose both a time step and a damping parameter in the Levenberg-Marquardt scheme. In the next section we discuss how to efficiently choose the damping term before we discuss how the the basic Levenberg-Marquardt algorithm can be improved.

3.4 The Levenberg-Marquardt algorithm

In this section we describe the basic concepts of the Levenberg-Marquardt algorithm. This primarily consists of a method for choosing the damping parameter, λ, which we discuss in section 3.4.1 and the parameter space metric $D^T D$, which we consider in section 3.4.2. Finally we discuss convergence and stopping criteria in section 3.4.3.

3.4.1 Choosing the damping parameter

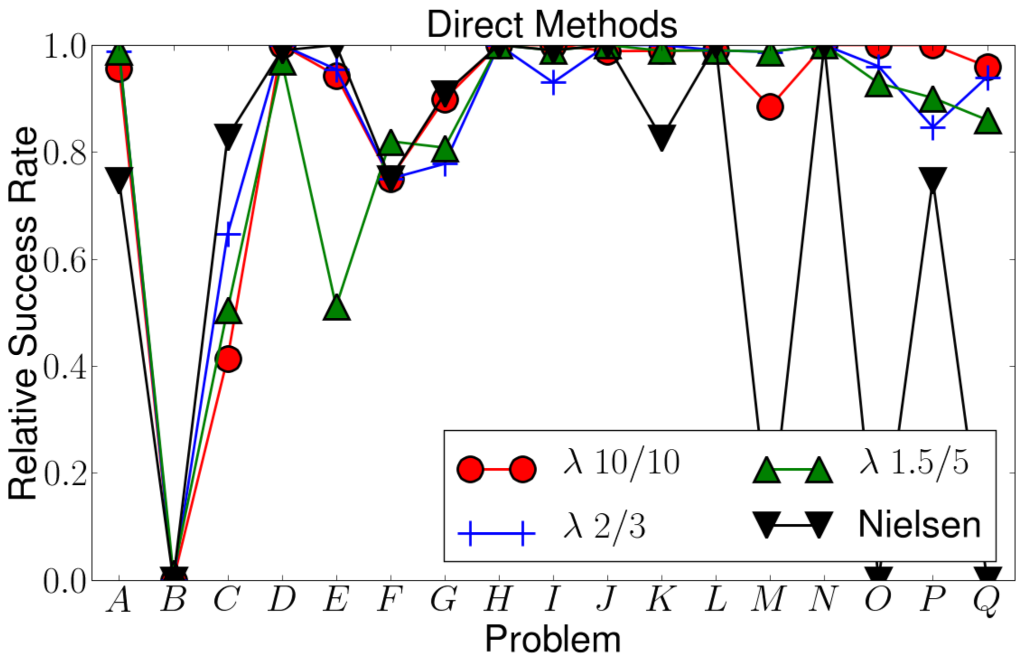
The basic strategy behind choosing the damping term uses the observations that the step size $\Delta^2 = \delta\theta^T D^T D \delta\theta$ is a monotonically decreasing function of λ in Eq. (3.9). Therefore, for a sufficiently large value of λ, the algorithm will take an arbitrarily small step in a descent direction. If a proposed step is unacceptable, one need only increase the damping term until a smaller, more acceptable step has been found. Because choosing λ is equivalent to choosing the step size, the Levenberg-Marquardt method can be considered a trust-region method. There are two general schemes for determining the appropriate damping. This can be done by either adjusting λ directly or by first choosing an acceptable step size Δ and then finding a λ such that $|\delta\theta| \leq \Delta$ (note that reference [72] describes how λ may be efficiently found for a given Δ). We will refer to these two schemes as direct and indirect methods respectively.

Many schemes have been developed to efficiently adjust λ or Δ. In our experience, the simple method originally suggested by Marquardt (with a slight modification described shortly) is usually adequate. In this scheme, if a step is accepted, then λ is decreased by a fixed factor, say 10. If a step is rejected then λ is appropriately raised by a factor of 10. As explained in reference [95], the qualitative effect of the damping term is to wash out the eigenvalues of the model manifold metric $J^T J$, which often are well spaced on a log-scale. It is therefore natural to choose the factor by which λ is either raised/lowered to be comparable to the eigenvalue spacing of $J^T J$. We have much greater success using a factor of 2 or 3 on most problems.

Additionally, we find that lowering λ by a larger factor than it is raised also produces more benficial results. For many moderate sized problems decreasing by a factor of 3 and raising by a factor of 2 is adequate. For larger problems, decreasing by a factor of 5 and raising by a factor of 1.5 is better. This scheme is known as delayed gratification, and its motivation is described in [95]. This basic scheme can also be applied to an indirect method by systematically increasing/decreasing Δ by a multiplicative factor instead of λ. The relative performance of these schemes, together with more complicated schemes describe by Nielson [77] and Moré [72] are summarized in Figs. 3.3 - 3.5. In these figures we have labeled direct algorithms with the prefix λ followed by the relative factors by which λ is lowered and raised. Similarly we have labeled indirect methods by the prefix Δ followed by the factor by which Δ is tuned.

There are several observations to be made from inspecting Figs. 3.3- 3.5. First, there is no algorithm that is clearly superior to any other. The relative success of any algorithm seems to depend very strongly on the problem. The most notable trend arises when comparing the direct methods (taken together) and the indirect methods (taken together). While the performance may vary among members within this group, there are clearly problems for which any indirect method is superior to any direct method and vice versa. Consider, for example, problems B and D. For these problems, indirect methods are both more successful and more efficient at finding good fits. On the other hand, direct methods seem superior for Problems J, K, and L.

The relative success of indirect methods on problems B and C can be understood by considering the eigenvalues of the of the Hessian matrix $J^T J$ near the best fit, as in figure 3.6. Notice that although they eigenvalue span many order of magnitude, they tend to collect near 10^3 and 10^1. Recall that the effect of the damping is to effectively wash out the eigenvalues smaller than λ. In order to be

a)


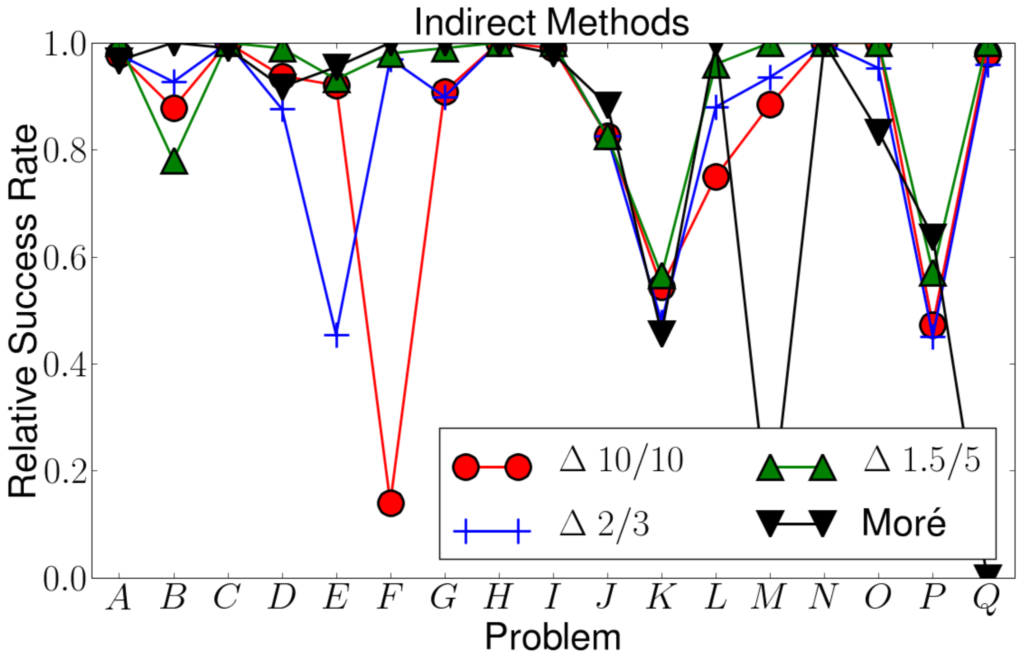
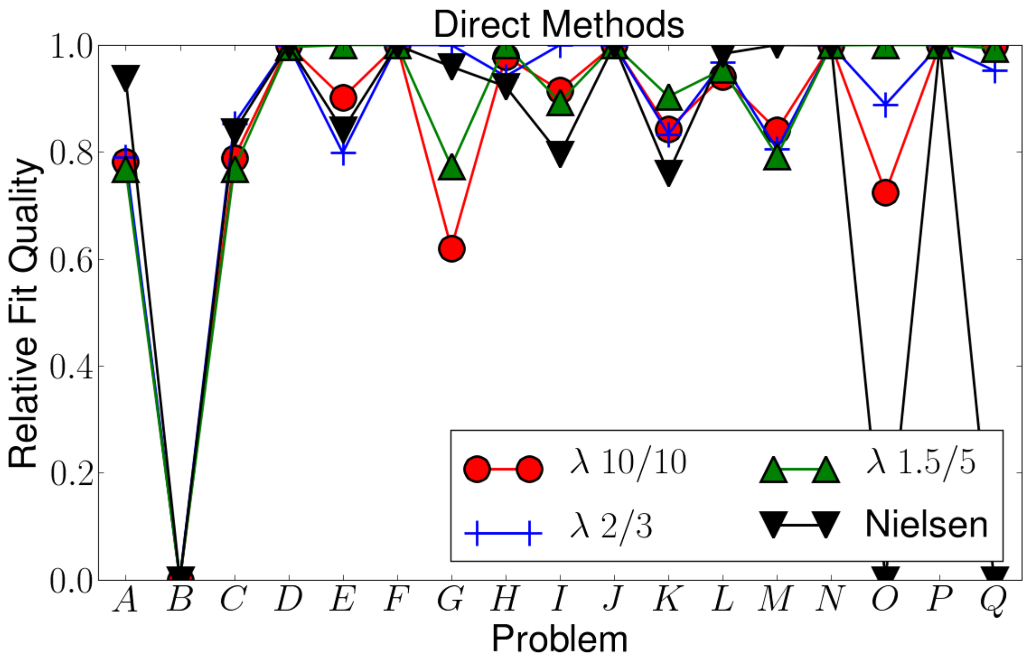
b)


Figure 3.3: **Relative Success Rate** of direct (a) and indirect (b) methods of choosing λ. The relative convergence rate is found by dividing each algorithm's convergence rate, as described in B.1, by the largest convergence rate of any method. Notice the direct methods' relative failure on problems B and C while the indirect methods struggled on J, K, and P.

a)


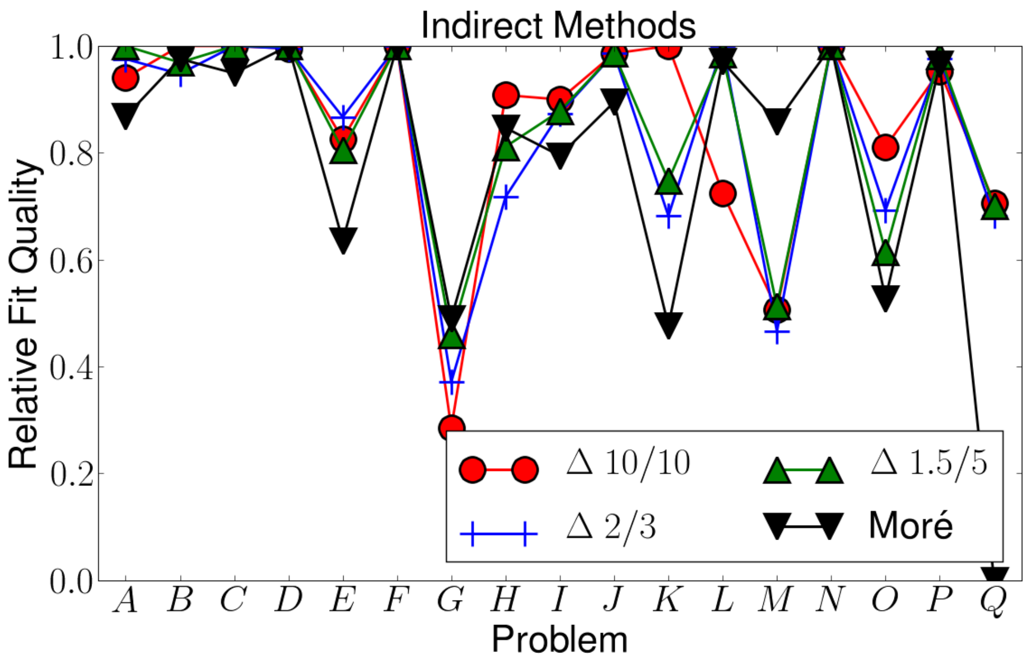
b)


Figure 3.4: **Relative Fit Quality** of direct (a) and indirect (b) methods of choosnig λ. The relative fit quality is found by dividing each algorithm's quality factor Q, as described in B.1, by the largest quality factor of any method. Again note the low success rate of direct methods on problems B and C.

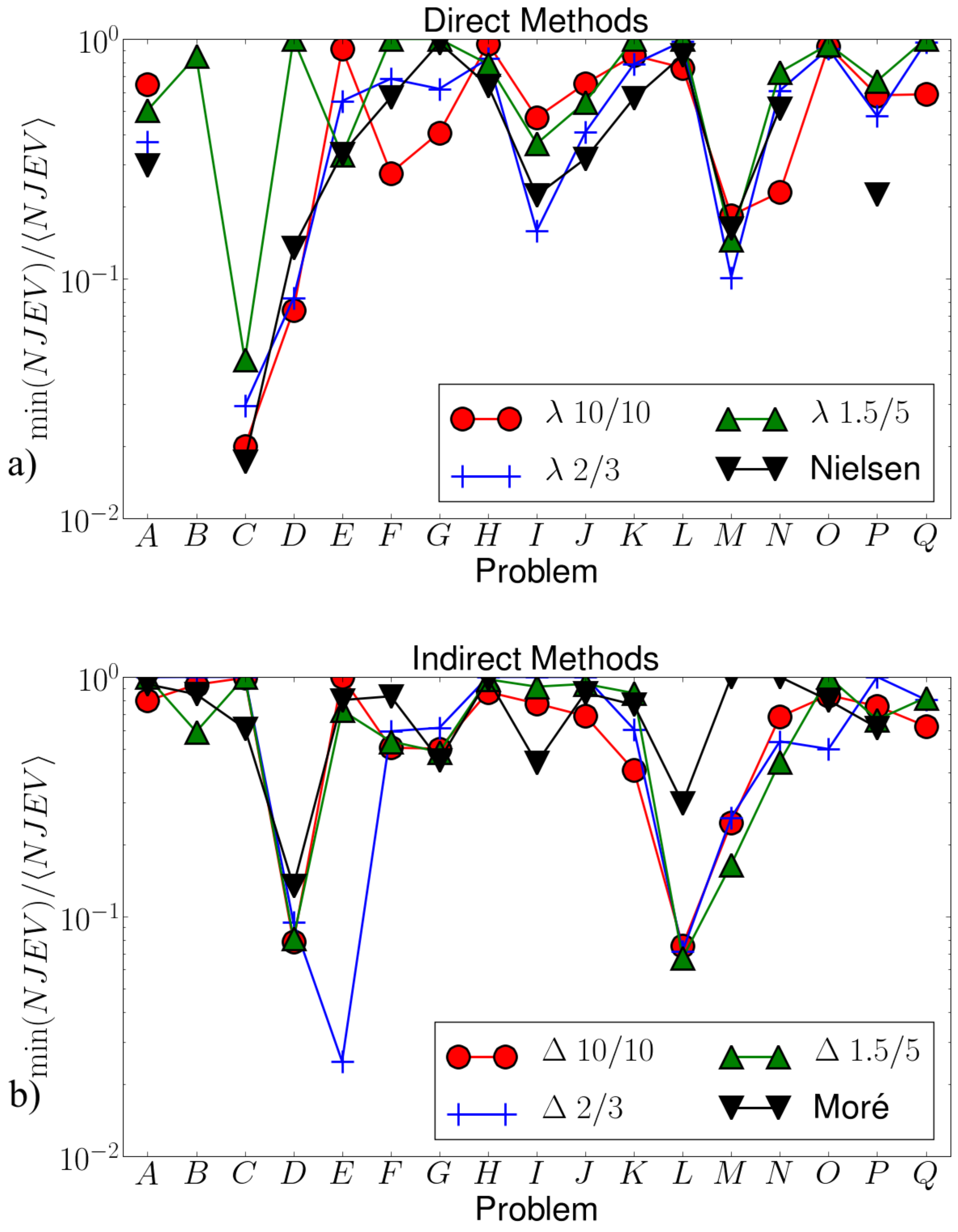


Figure 3.5: **Relative Inverse NJEV** of direct (a) and indirect (b) methods of choosing λ. The relative inverse NJEV is found by dividing an algorithm's average inverse NJEV by the fewest average inverse NJEV of any algorithm. Since we plot the inverse relative NJEV, larger numbers imply a more efficient algorithm. Notice how the indirect methods are sluggish on problems D and L compared to the direct methods.

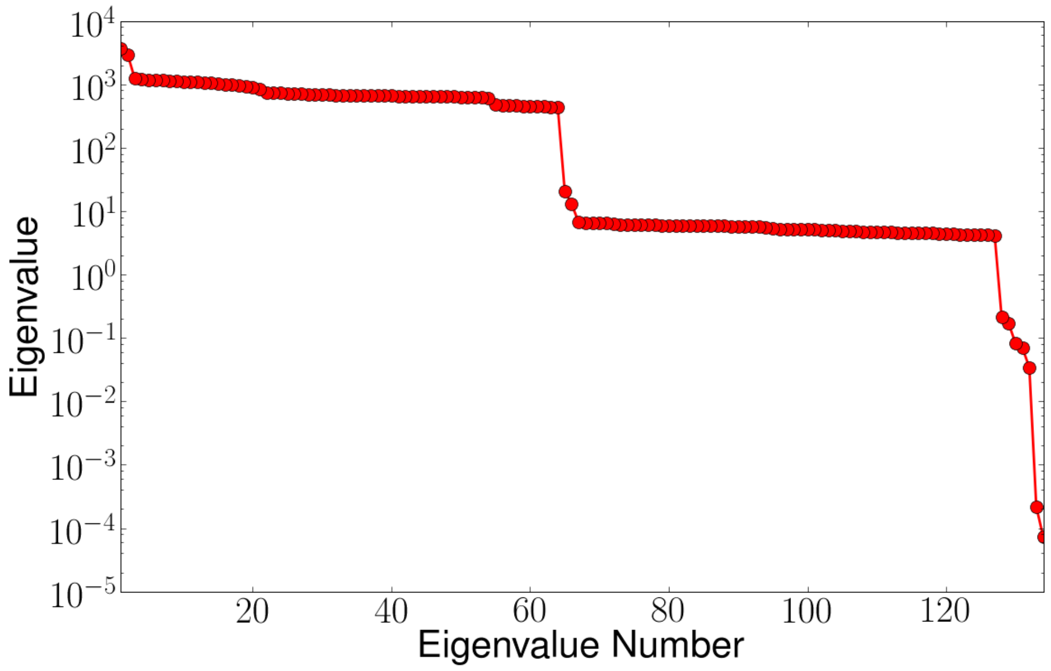


Figure 3.6: The eigenvalues of of the Hessian matrix for problem C. Although the eigenvalues span nearly eight order of magnitude, they are not evenly spaced over this range. There are two large collections are eigenvalues near 1000 and between 1 and 10. This clumping helps explain why indirect update methods are superior for this problem.

effective, an algorithm would therefore need to carefully tune λ while it was near 10^3, but then change very quickly while between 10^1 and 10^3. The direct methods described above cannot do this, however, the indirect methods that update Δ rather than λ seem to accomplish this naturally.

The relative success of direct methods on problems such as problem J, K, and L can be understood by a similar argument. In these problems there are regions of parameter space in which the step size must be finely tuned and other regions in which must change by large amounts. Most of the problems for which indirect methods struggled had exponential terms and initial guesses far form the best fit. The method initially had to move the parameters by large amount, however, once the algorithm descended into a canyon, the step size must be quickly reduced. It seems for these problems the direct methods that tune λ rather than the step size are more efficient.

In principle the improvements we suggest in subsequent sections could be applied to any of the eight methods summarized in Figs. 3.3- 3.5, or to any other variant of Levenberg-Marquardt. However, in practice we will restrict our attention to just two versions, one from each of the direct and indirect categories. Specifically, we will apply our improvements to the methods λ 2/3 and Δ 2/3 methods which lower/raise λ (raise/lower Δ) by factors of 2 and 3 respectively. Inspecting the performance of these two methods, we find that one of the two was either the best performer or nearly the best performer on almost all of the test problems. The effect of the improvements described in subsequent sections are similar for other update schemes.

3.4.2 Parameter space metric

We now describe how to choose an effective parameter space metric $D^T D$. The parameter space metric ideally should represent the relative scaling of the several parameters. Levenberg originally suggested an additive damping strategy, corresponding to $D^T D = \delta$, the identity. It has since been suggested that a multiplicative damping strategy in which $D^T D$ is a diagonal matrix with entries equal to the diagonal entries of $J^T J$ would more accurately capture the scaling of the several parameters. This method for choosing parameter scaling has the property that the algorithm is invariant to rescaling parameters, which is to say if the parameters of the model were replaced by the parameters $\tilde{\theta} = \Sigma\theta$ for some matrix Σ, then the sequence of iterates produced by the algorithm would be left unchanged.

The problem with a method that preserves scale invariance is that it greatly increases the susceptibility to parameter evaporation. In particular, if a parameter begins to evaporate, the model becomes less sensitive to the parameter, so it's corresponding entry in on the diagonal of $J^T J$ becomes small, in turn decreasing the damping of this parameter. This, however, is exactly the wrong behavior for dealing with parameter evaporation. Recall that the role of the parameter space metric is to introduce parameter dependence to the step, so a choice that is scale invariant is somewhat counter productive. On the other hand, we find that using the Marquardt scaling can greatly speed up the algorithm when it is in the region of a canyon, when parameter independence is crucial.

The popular implementation of Levenberg-Marquardt found in MINPACK uses a similar but superior method described in ref[72]. It chooses $D^T D$ to be diagonal with entries given by the largest diagonal entries of $J^T J$ yet encountered. This method also preserves invariance under rescaling but is more robust at avoiding parameter evaporation, however, it is still more prone to parameter evaporation than Levenberg scaling. This is because initial parameter guesses may lie in regions that do not produce enough damping.

A good compromise is to specify a minimum value of the damping terms in $D^T D$. This prevents the damping from being too small, either as the parameters evaporate or begin far from the canyon, but allows the algorithm to fine tune the scaling as it follows the canyon. We find that this method is both robust to parameter evaporation and efficient at finding good fits.

3.4.3 Convergence Criteria

Finally, we discuss criteria for the algorithm to stop searching for a best fit. It is important to distinguish between convergence criteria and stopping criteria. The former are criteria indicating that the algorithm has indeed found a local minimum of the cost, while the latter are criteria indicating that the algorithm is, in effect, giving up. In our comparison of the several algorithms, we list the convergence rate as a measure of the algorithms performance. This rate is the fraction of times the algorithm claimed to have successfully found a minima.

An elegant convergence criteria proposed by Bates and Watts also originated in the geometric interpretation of the least squares problem[11]. This method monitors the angle between the residual vector and the tangent plane, which we denote by ϕ. In particular, the cosine of the angle in data space is given by

$$\cos\phi = \frac{|P^T r|}{|r|}, \tag{3.10}$$

where P^T is a projection operator that projects into the tangent plane of the model manifold. Given a singular value decomposition of the Jacobian matrix $J = U\Sigma V^\dagger$, then $P^T = UU^T$. The algorithm can then be stopped when $\cos\phi$ is less than some quantity, say 10^{-2} or 10^{-3}.

This method provides a dimensionless convergence criterion that indicates how near one is to the minimum. It also has a statistical interpretation in terms of the accuracy of the solution in terms of the statistical uncertainty in the parameters. This method has a serious deficiency, however, when the model manifold has narrow boundaries as described in [95]. If the best fit happens to have evaporated parameters, a likely scenario for large models fit to noisy data, then $\cos\phi$ may be large although the algorithm has in fact converged.

As parameters evaporate, the model becomes less sensitive to that particular parameter combination and the Jacobian matrix has a singular value that becomes vanishingly small. (Note that the singular values correspond to the square root of the eignenvalues of the model manifold metric $J^T J$.) When it is sufficiently small we should consider these parameter directions to lie in the null space of J. Although the singular value may be formally nonzero, for computational purposes we understand that the algorithm will not make any more progress by moving the parameters in these directions and they should not contribute to the tangential component of the residuals.

To remedy this situation, we replace the projection operator $P^T = UU^T$ in Eq. (3.10) with $P^T = \tilde{U}\tilde{U}^T$ where $\tilde{U}$ is a matrix of left singular vectors of J for which the corresponding singular value is larger than some threshold. If the function is evaluated to precision ϵ, then we find that ignoring the directions with singular values less than $\sqrt{\epsilon} \max \Sigma$, where $\max \Sigma$ is the largest singular value, works well. An alternative solution is to use a convergence criteria when the gradient of the cost falls below a certain threshold.

In addition to the convergence criterion described above, the algorithm should have a number of stopping criteria. In our implementation we provide stopping criteria for when a maximum number of residual and Jacobian evaluation have been reached, in addition to a maximum number of iterations of the algorithm. We also provide stopping criteria for when the gradient of the cost has fallen to some threshold, when the change in parameter values becomes sufficiently small, and when cost itself has reached some acceptable value.

3.5 Geodesic Acceleration

In order to improve the efficiency of the Levenberg-Marquardt method, we return to the interpretation presented in section 3.3 that the Levenberg-Marquardt method is an Euler approximation to geodesic flow on the model graph. To improve upon this method, we propose iterating with the parabolic step given by

$$\delta\theta = \dot{\theta}\delta\tau + \frac{1}{2}\ddot{\theta}\delta\tau^2 \tag{3.11}$$

rather than the first-order, linear Euler step $\delta\theta = \dot{\theta}\delta\tau$.

The acceleration term in Eq. (3.11) can be calculating by differentating Eq. (3.6) once with respect to τ. The result after some simplification is

$$\begin{aligned} \ddot{\theta}^\mu &= -g^{\mu\nu} J_{m\nu} A_{m\alpha\beta}\dot{\theta}^\alpha\dot{\theta}^\beta + \\ &\quad \frac{1}{\sqrt{g_{\epsilon\delta}\dot{\theta}^\epsilon\dot{\theta}^\delta}}\left(\dot{\theta}^\mu\dot{\theta}^\gamma\dot{\theta}^\nu r_m P^N_{mn} A_{n\nu\gamma} - \dot{\theta}^\gamma g^{\mu\nu} r_m P^N_{mn} A_{n\nu\gamma}\right), \end{aligned} \tag{3.12}$$

where $P^N = 1 - J(J^TJ)^{-1}J^T$ is an operator projecting normal to the tangent plane of the manifold. At this point a major simplification arises when we make the assumption that the extrinsic curvature is very small. In this case, all terms in which a second derivative is contracted with P^N are negligible, leaving

$$\ddot{\theta}^\mu = -g^{\mu\nu} J_{m\nu} A_{m\alpha\beta}\dot{\theta}^\alpha\dot{\theta}^\beta. \tag{3.13}$$

It is surprising how good the small-curvature approximation turns out to be. As shown in reference [95], the extrinsic curvature of the model manifold should under many circumstances be very small compared the largest step size an algorithm can take, which is limited by the so-called parameter-effect curvature. When Bates and Watts first introduced measures of parmeter-effects curvature, they noted that for every problem they considered the parameter-effects curvature was larger (often much larger) than the extrinsic curvature[9]; explicit examples in [95] show parameter-effects curvature with up to six order of magnitude larger extrinsic radii of curvature compared to the allowed step sizes. Although there are assuredly counter examples, it is reasonable to expect that for most problems of interest, this approximation will hold.

The second order differential equation given in Eq. (3.13) is the geodesic equation on the manifold graph. We refer to this as the geodesic acceleration correction to the path and iterating according to Eq. (3.11) is the geodesic acceleration algorithm. As expected, the geodesic acceleration depends on the second derivative of the model, but perhaps surprisingly, the only dependence is on the directional second derivative oriented along the velocity, $\dot{\theta}$. This result is significant, as calculating the full array of second derivatives would likely be prohibitively expensive for large models; a directional second derivative has a small computational cost comparable to a single evaluation of $r(\theta)$, and in fact a finite-difference estimate can be found with only one additional evaluation of r. In contrast, for large models, most of the computational cost of minimizing least-squares problems comes from evaluating the Jacobian matrix. In these cases, the additional cost of evaluating the second order correction is negligible.

The benefit of including the geodesic acceleration comes when the algorithm is navigating a narrow canyon towards the best fit as illustrated in Figure 3.7. By approximating our path with a parabola, the geodesic acceleration method can more accurately follow the path of a winding canyon toward the best fit.

In order to utilize the geodesic acceleration as an addition to the Levenberg-Marquardt algorithm, it is necessary to make one other small addition. The usual Levenberg-Marquardt algorithm accepts all steps that decrease the cost function;

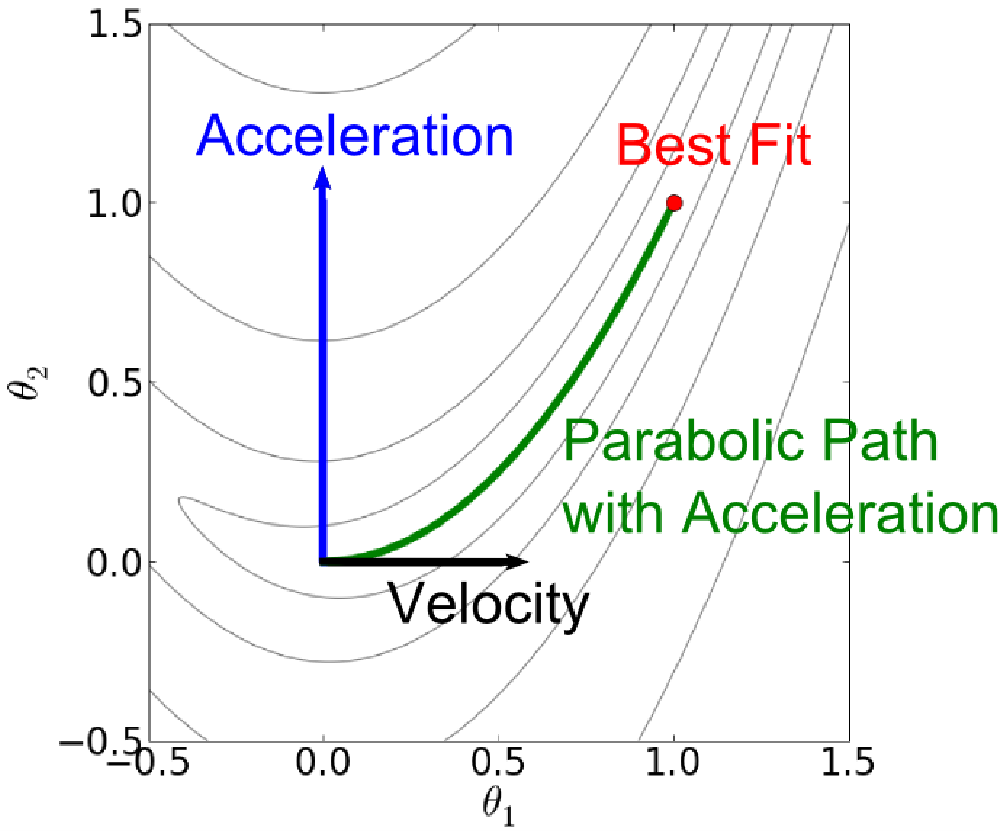


Figure 3.7: When navigating a canyon towards the best fit, the geodesic acceleration indicates in which direction the canyon is curving. By approximating the path with a parabola, the best fit can be found in fewer iterations.

however, in order to ensure that the geodesic acceleration algorithm converges, it is also necessary to monitor the relative magnitude of contribution from the second order term. We require acceptable steps to satisfy the condition

$$\frac{|\ddot{\theta}|\delta\tau}{|\dot{\theta}|} < \alpha, \tag{3.14}$$

where α is some number of order 1 that is set by the user and whose optimal value may vary from problem to problem. The motivation for this requirement is that the proposed step represents a truncated Taylor series and so the terms ought to be decreasing in magnitude to gurantee convergence. For most problem we find that $\alpha = 0.75$ is a good guess. Problems for which convergence is difficult, $\alpha = 0.1$ is an effective choice.

Without the requirement in Eq. (3.14), the resulting algorithm is very unpredictable and will often take large, uncontrolled steps and become lost. This phenomenon is closely related to parameter evaporation that gives so much trouble to other algorithms. The additional requirement in Eq. (3.14) helps the geodesic algorithm avoid parameter evaporation, increasing its likelihood of convergence.

We note that for a given value of α one can always find a suitable value of λ such that Eq. (3.14) is satisfied as long as the second derivative is reasonably well behaved. In particular, if λ is very large, then $g^{-1}, \approx \frac{1}{\lambda}(D^T D)^{-1}$, and

$$\dot{\theta} \approx \left(\frac{1}{\sqrt{\lambda}}\right)\left(\frac{(D^T D)^{-1}\nabla C}{\sqrt{(\nabla C)^T (D^T D)^{-1} \nabla C}}\right) \tag{3.15}$$

so that $u = \sqrt{\lambda}\dot{\theta}$ is independent of λ. Then it follows that

$$\ddot{\theta} \approx \frac{1}{\lambda^2}\left(D^T D\right)^{-1} J^T A u u, \tag{3.16}$$

where $Auu = u^\mu u^\nu \partial_\mu \partial_\nu r$ is the directional second derivative along u, and

$$\frac{|a|\delta\tau}{|v|} \approx \left(\frac{1}{\lambda^2}\right)\left((\nabla C)^T (D^T D)^{-1} \nabla C\right)\left(\frac{|\left(D^T D\right)^{-1} J^T A u u|}{|\left(D^T D\right)^{-1} \nabla C|}\right). \tag{3.17}$$

As long as $\nabla C \neq 0$ (in which case the algorithm is converged), the only complication can come from the directional second derivative, Auu. As long as this is not singular, which is a reasonable expectation for most problems, a suitable λ can always be found.

In section 3.4.2 it was noted that for a suitable choice of $D^T D$ the iterates produced by the Levenberg-Marquardt algorithm are invariant to a change of scale of the parameters. It is interesting to note that this result is unchanged by including the geodesic acceleration. It is also worth noting that by including the geodesic acceleration, the step size is no longer a monotonically decreasing function of λ. Whereas the bare Levenberg-Marquardt algorithm is considered a trust region method since it chooses a step size rather than a search direction, the geodesic acceleration algorithm is a combination of trust region and linesearch methods. In particular, the choice of damping, λ can be considered a trust region in velocities, $\dot{\theta}$. Having selected a velocity, the acceleration explores the model behavior in the chosen direction and diverts and dampens the step as appropriate.

Before discussing the performance of the geodesic acceleration, we offer a few remarks about evaluating the directional second derivative necessary for the geodesic acceleration. As is always the case, analytic derivatives are preferable to finite difference estimates. In principle, if an analytic expression can be found for the first derivatives, in principle one could also find an expression for the directional second derivative, although the resulting expression may be very complicated, especially for large models. (With code often being generated by computer algebra systems, it may not be unreasonable in some cases.) Automatic differentiation also might make exact evaluations of the second derivative feasible.

In cases where a directional second derivative cannot be evaluated analytically, one can always use a finite difference approximation. The usual formula for the directional second derivative is

$$A_{m\mu\nu}\dot{\theta}^{\mu}\dot{\theta}^{\nu} \approx \frac{r_m(\theta + h\dot{\theta}) - 2r_m(\theta) + r_m(\theta - h\dot{\theta})}{h^2}, \tag{3.18}$$

for some finite-difference step size, h. When evaluating Eq. (3.18), note that the algorithm will already have evaluated $r(\theta)$, leaving two additional function evaluations necessary for the estimate. The algorithm has also previously evaluated the Jacobian matrix. Using this information one can provide a finite difference estimate with just a single additional function evaluation:

$$A_{m\mu\nu}\dot{\theta}^\mu\dot{\theta}^\nu \approx \frac{2}{h}\left(\frac{r_m(\theta + h\dot{\theta}) - r_m(\theta)}{h} - J_{m\mu}\dot{\theta}^\mu\right). \tag{3.19}$$

In practice, the finite difference estiamte may be sensitive to h, particularly if the problem is poorly scaled. We find in practice that choosing a large finite-difference step size, giving something analogous to a secant estimation, is less dangerous than a step that is too small. In general, choosing $h = 0.1$, so the step is about 10% of the velocity seems to work reasonably well.

We now consider how the geodesic acceleration effects performance summarized in figure 3.8, notice we have appended the letter A after the algorithm to indicate the it now includes acceleration. We see that with a few exceptions the acceleration improves the algorithm's performance in each of the three measures we have used. The most dramatic improvement is in the number of Jacobian evaluations necessary for convergence, where we saw speed ups as large as a factor of 70, with most improvements between a factor of 2 and 10.

Perhaps the most significant benefit gained from geodesic acceleration is in the improved success rate and fit quality. We attribute this improvement to the modified acceptance criterion given in Eq. (3.14). For many cases the algorithm could be improved further by a smaller choice of α (the results in figure 3.8 are for $\alpha = 0.75$), although this comes at a cost in convergence speed (more Jacobian evaluations).

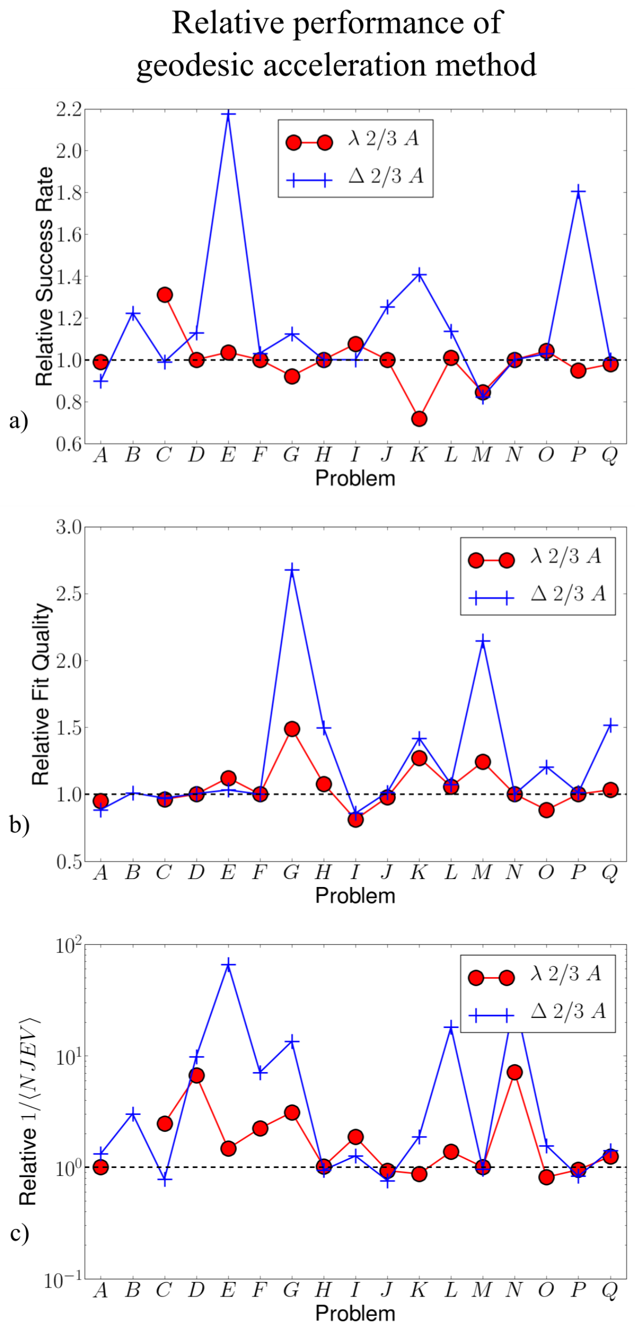


Figure 3.8: **Performance of geodesic acceleration** The relative success rate (a), quality factor (b) and inverse NJEV (c) of two algorithms using geodesic acceleration. The rates are each relative to each algorithm's performance without acceleration. One each plot, points larger than 1 (dashed black line) represent an improvement. Notice that by including the acceleration the algorithm typically finds better fits more often in less time. Perhaps most dramatically, in some cases the NJEV have been reduced by a factor of 70!.

3.6 Uphill steps

It is necessary for the algorithm to have some way to distinguish whether a proposed step should be accepted or rejected. The standard method of the Levenberg-Marquardt algorithm is to accept any step that decreases the cost while rejecting all steps that increase the cost. In the geodesic flow interpretation of the algorithm, this is a natural choice as the cost is a Lyapunov function for the dynamical system defined by Eq. (3.6). Such a function is locally minimum at the fixed points of the flow and decreases monotonically as the system approaches the equilibrium. The algorithm then accepts all steps which decrease the Lyapunov function and reject those that increase it. It is straightforward to see that the cost is a Lyapunov function for Eq. (3.6), and all implementations of the Levenberg-Marquardt algorithm of which the authors are aware accept or reject steps solely on whether or not the cost decreases or increases respectively. While this choice of Lyapunov function is natural and leads to a stable algorithm, we argue in this section that often it is not the most efficient choice.

When an algorithm must follow a narrow canyon to the best fit, if the aspect ratio of the canyon is very large, then there will be only a small sliver of steps that decrease the cost. It has been shown previously, that if an algorithm decreases the cost as little as possible in each step, it will be able to take larger steps and will traverse the desired path much more quickly[95]. This observation suggests that while in the canyon a decrease in the cost function is not necessarily the best measure of progress towards the best fit. Furthermore, if decreasing the cost as little as possible is beneficial, perhaps allowing increases in the cost could also be beneficial. In this section we elaborate on this idea to construct a method for accepting uphill steps as long as progress towards the best fit is still being made.

Consider the function

$$L(\theta) = \int_0^{\tau_{bf}} |\dot{\theta}(\tau')|^2 d\tau', \tag{3.20}$$

with $\theta(\tau)$ given by Eq. (3.6). It is straightforward to show that Eq. (3.20) is also a Lyapunov function for Eq. (3.6), since

$$\frac{d}{d\tau} L(\theta) = -|\dot{\theta}(\tau)|^2 \leq 0. \tag{3.21}$$

The physical meaning of $L(\theta)$ is the arc-length in parameter space from the point θ to the best fit along the path given by Eq. (3.6). In practice, calculating $L(\theta)$ would be more computationally intensive than minimizing the sum of squares function; however, the spirit of Eq. (3.20) is that steps may be safely accepted if they move us closer to the best fit as measured by the parameter space distance, even if the cost temporarily increases. We now construct an acceptance scheme motivated by this parameter arc-length criterion, Eq. (3.20).

In our new acceptance scheme, we accept steps if

$$\beta C_{\text{new}} \leq C_{\text{old}}, \tag{3.22}$$

where $\beta \leq 1$ is some measure of progress made towards the best fit. We define β by comparing the cosine of the angle in parameter space between the old and new vectors tangent to the path:

$$\beta = 1 - \cos\left(\dot{\theta}_{i+1}, \dot{\theta}_i\right), \tag{3.23}$$

where $\dot{\theta}_{i+1}$ is the velocity of the proposed step and $\dot{\theta}_i$ is the velocity of the last accepted step. Qualitatively, if the two vectors are nearly aligned, then β will be very small. In this case we accept uphill moves as they are more likely to be in the direction of the best fit, i.e. decreasing the arc-length function. If the vectors are nearly orthogonal or point in opposite directions, then we accept only downhill moves since the path is changing direction and we have no other way to know if progress is being made. If $\beta > 1$, then we accept only downhill steps.

There are a few variations of Eq. (3.22) that one may want to consider:

$$\beta^2 C_{\text{new}} \leq C_{\text{old}} \tag{3.24}$$

will accept moves with larger increase in the cost while the conditions

$$\beta C_{\text{new}} \leq C_{\text{best}} \tag{3.25}$$

$$\beta^2 C_{\text{new}} \leq C_{\text{best}} \tag{3.26}$$

will be less accepting of uphill moves (C_{best} is the lowest cost yet found). The ideal choice is likely to depend on the particular problem. We refer to this acceptance criterion as the "bold method,". The results we show below correspond to Eq. (3.26).

Although in general, calculating $L(\theta)$ would be a daunting task, for simple test problems for which the minimum is already known, such as the Rosenbrock function, it is not difficult. Consider the Rosenbrock function with residuals

$$r(\theta) = \begin{pmatrix} 1 - \theta_1 \\ 100(\theta_2 - \theta_1^2) \end{pmatrix} \tag{3.27}$$

and a minimum cost at $\theta = (1, 1)$. This function is characterized by a narrow parabolic valley around the best fit. An algorithm beginning from $\theta = (-1, 1)$ must follow the narrow parabola to reach the global minimum. In table 3.1 we compare the residual sum of squares and the arc-length function evaluated at each iterate of the Levenberg-Marquardt algorithm when uphill moves were accepted according to the "bold method" described above. Although the cost appears to grow in a seemingly uncontrolled manner, the arc-length function decreases steadily and the

Iterate	$C(\theta)$	$L(\theta)$	Iterate	$C(\theta)$	$L(\theta)$
1	2.00	2.96	6	1789.68	0.39
2	1.95	2.90	7	32.83	0.14
3	1.95	2.75	8	0.26	0.02
4	11.34	2.34	9	5.1e-5	8.7e-4
5	417.71	1.51	10	1.8e-10	1.2e-5

Table 3.1: Table comparing sum of squares, $C(\theta)$ and the Lyapunov function $L(\theta)$ defined in Eq. (3.20) for a series of iterates on the Rosenbrock function for which uphill moves were accepted. Note that although the residual sum of squares increased temporarily by several orders of magnitude, the arc-length function L decreased at a steady rate. The same algorithm would have taken nearly five times as many steps to reach the minimum if uphill moves had been rejected.

algorithm ultimately converges. Although in general the algorithm cannot monitor the value of $L(\theta)$ at each iterate, table 3.1 suggests that the acceptance scheme described above is a reasonable substitute. We emphasize that the bold acceptance scheme does not guarantee that $L(\theta)$ decreases, and there is no guarantee that accepting uphill moves will always lead to convergence. In practice, however, we find this method to be fairly robust.

We now investigate how the bold acceptance criterion affects the algorithms' performance. The results are summarized in figure 3.9. Notice that we have appended the letter B to the algorithm name to indicate that the bold method was used. While the results vary from problem to problem, it appears that for many cases the bold acceptance may reduce the success rate of the algorithm. We note that this is due to the relatively difficult starting points of many of the test problems. If we had used the standard starting points supplied by the Minpack-2 or NIST versions of problems A - N, the success rate and fit quality would have been much higher. This result is a reflection of that fact that there is no guarantee for convergence when uphill moves are accepted.

On the other hand, the bold acceptance criterion speeds up the algorithm, in some cases by a factor of 30! In our experience if a problem is chosen such that there is little chance of the algorithm becoming lost or not converging, then accepting uphill moves can greatly reduce the time to find good fits. This is most likely to be useful for fitting problems that either start in a narrow canyon or can easily find the canyon but become sluggish en route to the best fit. If the problem is difficult because it is hard to find a canyon, then accepting uphill moves are not likely to improve the search.

In figure 3.10 we show results for combining geodesic acceleration with the bold acceptance method, which we denote by the letters AB. It is clear that the geodesic acceleration helps to control the method in the regions where a bold acceptance may have otherwise become lost. We remind that the reader that by lowering the parameter α from Eq. (3.14) the algorithm can be made increasingly stable at the cost of more iterations. Which variation of the algorithm is the most efficient in terms of the number of Jacobian evaluations varies from problem to problem.

3.7 Updating the Jacobian Matrix

In comparing algorithm performance, we have assumed that the most computationally intensive part of the Levenberg-Marquardt algorithm is an evaluation of the Jacobian matrix of first derivatives. If this is done using finite difference then for a model of N parameters the Jacobian matrix is N times as expensive as a residual evaluation. If analytic formulas are available, it may be more efficient than a finite difference approximation, but for large N Jacobians will still occupy the bulk of the computer time. Much of the discussion of this paper has revolved around reducing the number of Jacobian evaluations necessary for convergence.

Relative performance of
bold acceptance method

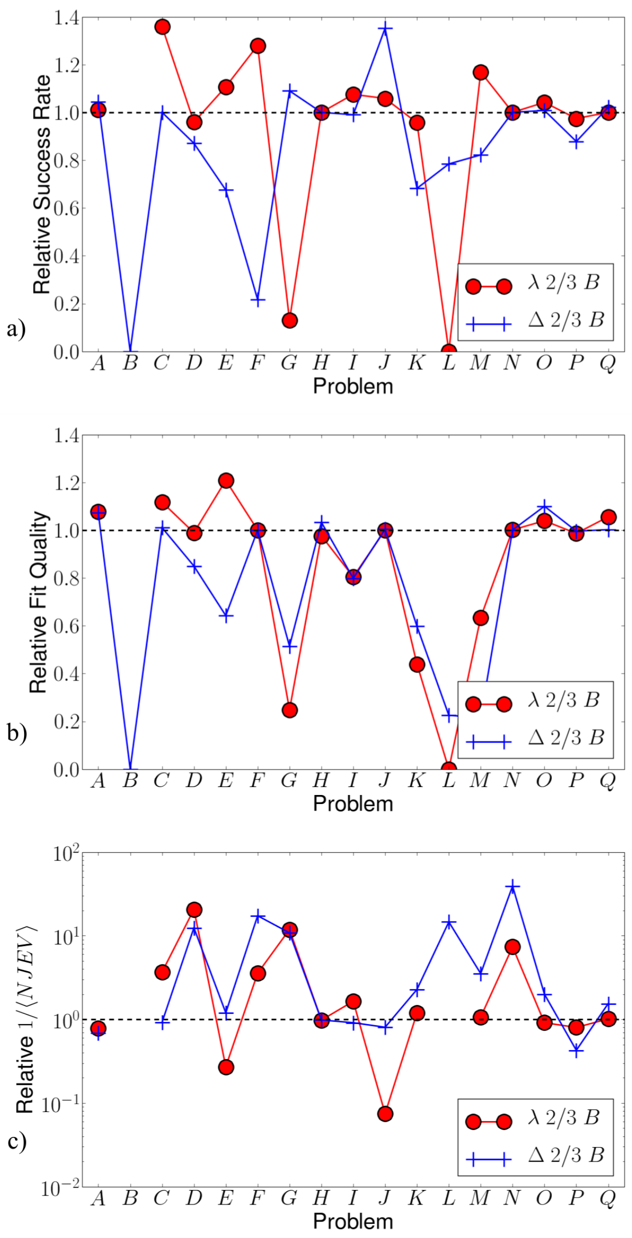


Figure 3.9: **Performance of bold acceptance**. The relative success rate (a), fit quality (b) and inverse NJEV (c) of two algorithms using bold acceptance criterion. The rates are each relative to each algorithm's performance while accepting only downhill moves. On each plot, points larger than 1 (dashed black line) represent an improvement. For many of these problems, accepting uphill moves increases the probability that the algorithm will become lost. However, when the algorithm does succeed, it may be much faster!

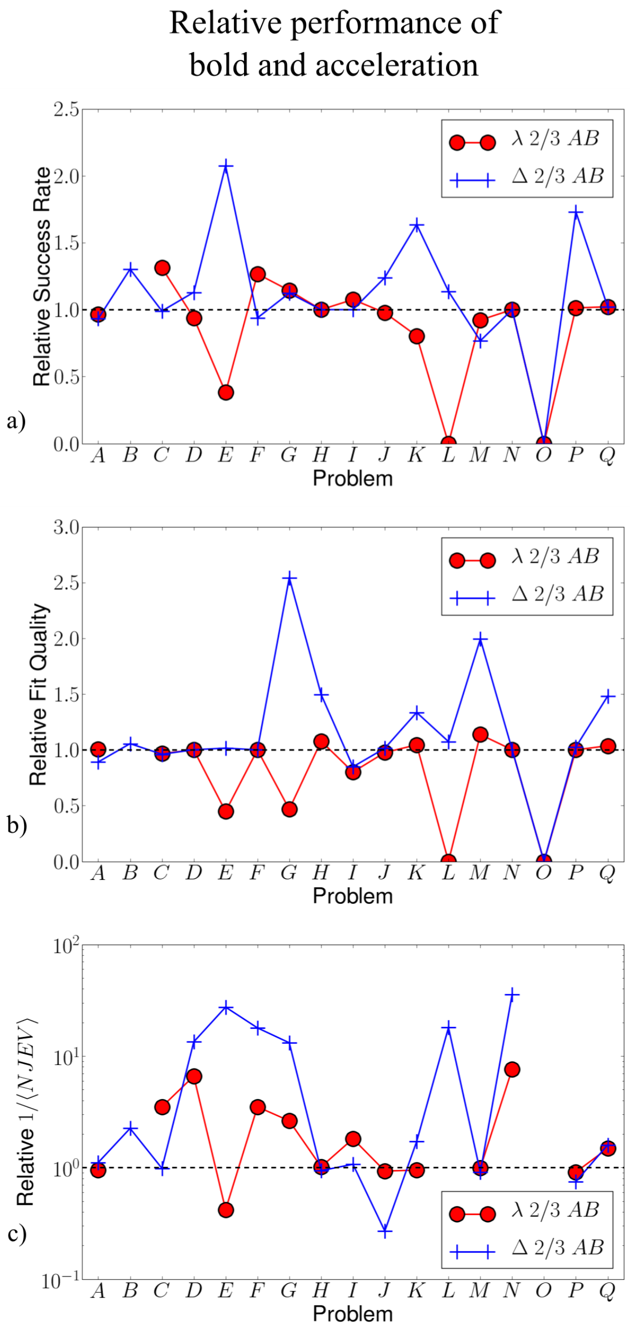


Figure 3.10: **Performance of the bold and acceleration** The relative success rate (a), fit quality (b) and inverse NJEV (c) of two algorithms using bold acceptance criterion. The rates are each relative to each algorithm's performance while accepting only downhill moves. On each plot, points larger than 1 (dashed black line) represent an improvement. Notice that by including the acceleration we are able to prevent the algorithm from becoming lost when using the bold acceptance.

Typically, the algorithm will evaluate the Jacobian after each accepted step in order to calculate the gradient $\nabla C = J^T r$ and the metric $g = J^T J + \lambda D^T D$. Broyden suggested a quasi-Newton root fidning method that evaluates the Jacobian only on the first iteration and subsequently updates the Jacobian with a rank-1 update[20]. Thus, given the Jacobian at the previous iterate, J_{n-1}, the Jacobian at the current parameter values J_n is estimated to be

$$J_n = J_{n-1} + \frac{\Delta r_n - J_{n-1}\Delta\theta_n}{|\Delta\theta_n|^2}\Delta\theta_n^T, \tag{3.28}$$

where $\Delta r_n = r_n - r_{n-1}$ is the change in the residual vector and $\Delta\theta_n = \theta_n - \theta_{n-1}$ is the change in the parameter space vector. In principle this method can be applied to the Levenberg-Marquardt method to eliminate the need to evaluate J at each step.

In practice, after many such updates, the matrix J may become a very poor approximation to the actual Jacobian, resulting in a poor estimate of the gradient direction. If the algorithm's performance suffers as a result, it may be necessary to reevaluate the whole Jacobian matrix and begin the update scheme anew. We therefore reevaluate the Jacobian after a few proposed steps have been rejected by the algorithm. Typically, reevaluating after one or two rejections works well.

Is it possible to improve upon Broyden's update method so as to reduce the frequency that the Jacobian needs a full udpate? An approach one might take is to note that Broyden's method is based upon a secant approximation

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}, \tag{3.29}$$

where for the moment we consider the simple case of a function a scalar argument. It appears that this update is less accurate than it could be. In particular the derivative (or at least an approximation of the derivative) is known at x_{n-1},

$f'(x_{n-1})$ which information has been ignored. We can incorporate this additional information by considering the Taylor series centered at x_{n-1}:

$$f(x) \approx f(x_{n-1}) + f'(x_{n-1})(x - x_{n-1}) + \frac{1}{2} f"(x_{n-1})(x - x_{n-1})^2, \tag{3.30}$$

where $f"(x_{n-1})$ is then estimated from the new function evaluation $f(x_n)$:

$$f"(x_n) \approx 2 \left(\frac{f(x_n) - f(x_{n-1})}{\Delta x^2} - \frac{f'(x_{n-1})}{\Delta x} \right). \tag{3.31}$$

The derivative at $f'(x_n)$ is then estimated by

$$f'(x_n) \approx f'(x_{n-1}) + f"(x_{n-1})(x_n - x_{n-1}). \tag{3.32}$$

Repeating this calculation in multiple dimensions and using the second derivative matrix with least Frobenius norm consistent with the new function evaluation produces the Jacobian upadte formula

$$J_n = J_{n-1} + 2 \frac{\Delta r_n - J_{n-1} \Delta \theta_n}{|\Delta \theta_n|^2} \Delta \theta_n^T, \tag{3.33}$$

multiplying the Broyden correction by a factor of two. Unfortunately, using Eq. (3.33) does not give significantly better results than Eq. (3.28).

When the geodesic acceleration is included, then each iteration has the information of two function evaluations and one can construct a rank-2 update to the Jacobian. We accomplish this as follows: if a step is proposed with velocity v and acceleration a, then we first assume a step was taken corresponding to $\delta\theta = v/2$ and residuals $r_n + \frac{1}{2} J v + \frac{1}{8} v^\mu v^\nu \partial_\mu \partial_\nu r$. Notice this expression for the residuals involves the directional second derivative which has been evaluated to calcualte the geodesic acceleration. A rank-1 Broyden update is then performed for this step. It is then assumed that a second step is taken corresponding to $\delta\theta = \frac{1}{2}(v + a)$ and another rank-1 update is performed corresponding to this step.

In practice there is little performance gain from using the rank-2 update instead of the rank-1 update suggested by Broyden. This is most likely because the velocity and the acceleration are often nearly colinear, as described in [95]. However, as there are cases in which a rank-2 update can be beneficial, we include it in our method.

In figure 3.11 we present the performance results for methods with rank-1 updates after accepted steps. (Note that we distinguish a method using Broyden's update by the letter C in the algorithm's name.) The qualitative effect of this addition is similar to that of the bold acceptance criterion. The algorithm appears to be more likely to get lost, although it can be much faster in terms of the number of Jacobian evaluations. We understand the lower success rates and fit quality as a direct consequence of using an approximate Jacobian matrix, resulting in an inaccurate gradient direction.

In figure 3.12 we present the results of the algorithm that uses a rank-2 update using both geodesic acceleration and the bold acceptance method. These additions do not appear to have much effect on the algorithm's performance. It is unfortunate that the low convergence and success rates could not be salvaged in these cases by including the geodesic acceleration. It is likely that a more sophisticated scheme for deciding when the Jacobian should be reevaluated could improve it's robustness against becoming lost. We recommend using this Jacobian update scheme only when there is very little chance that the algorithm will become lost or when the Jacobian is so expensive that evaluating it multiple times is not possible.

Particularly for large problems, the relative cost of a Jacobian evaluation to a function evaluation reflects the relative information content of the two. For small problems the rank-1 update contains a significant fraction of the infomartion avail-

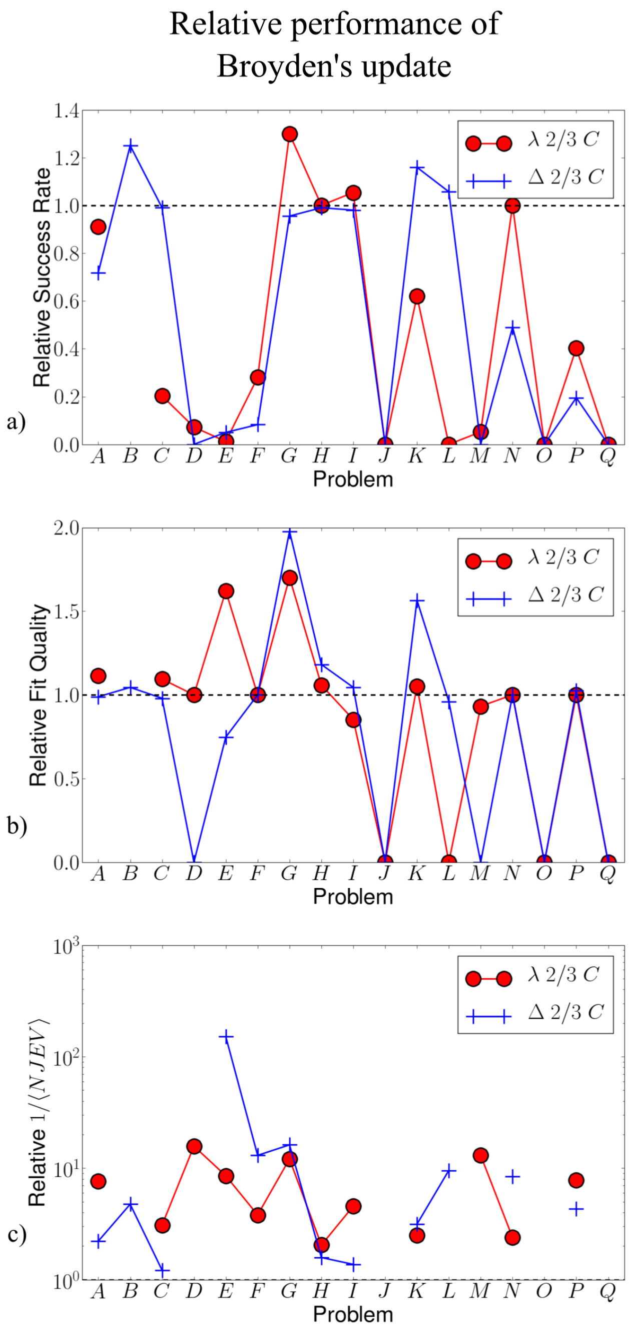


Figure 3.11: **Performance of Broyden's Update** The relative success rate (a), fit quality (b) and inverse NJEV (c) of two algorithms using a rank-1 Broyden's update. The rates are each relative to each algorithm's performance without such an update. On each plot, points larger than 1 (dashed black line) represent an improvement. Including the Broyden update typically causes the algorithm to be less robust to initial conditions, manifest by a lower success rate and average fit quality. Without the need to reevaluate the Jacobian after each accepted step, the best fit can often be found much more quickly. Note that missing points in (c) correspond to points for which the convergence rate and average quality factor was very near zero so that comparisons do not have any merit.

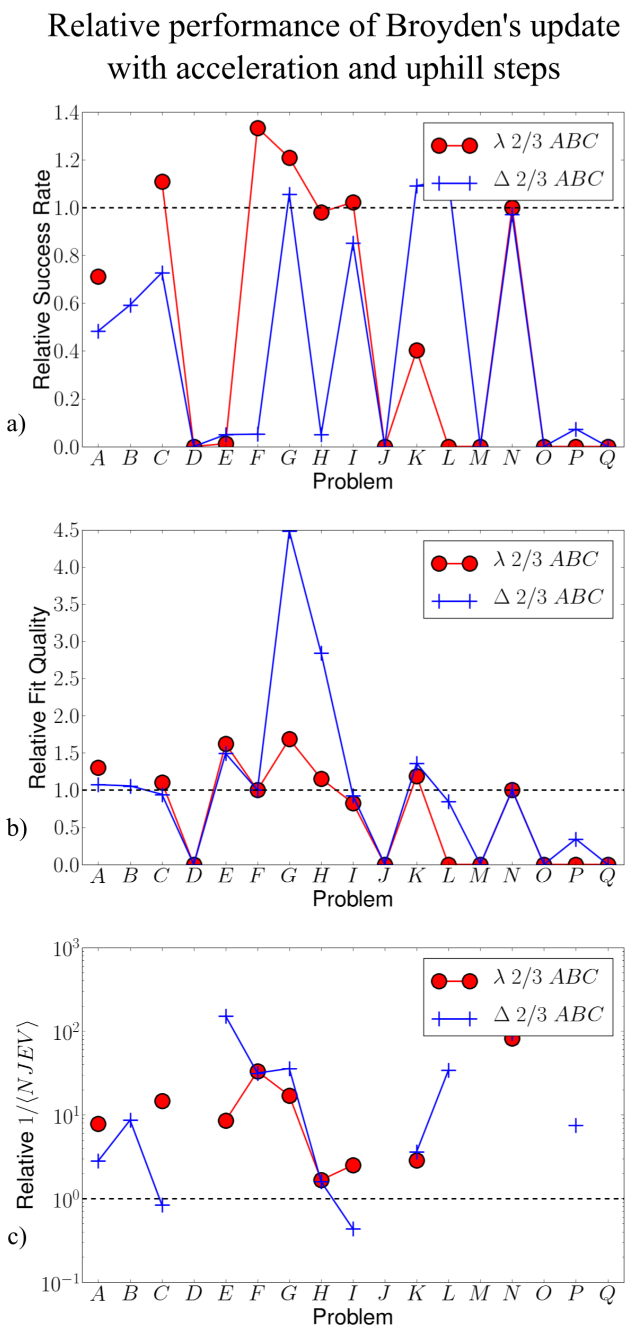


Figure 3.12: **Performance of Broyden's Update with acceleration and bold** The relative success rate (a), fit quality (b) and inverse NJEV (c) of two algorithms using geodesic acceleration. The rates are each relative to each algorithm's performance without acceleration. On each plot, points larger than 1 (dashed black line) represent an improvement. Notice that by including the acceleration and bold does not have a strong effect on the results of the algorithm.

able in the Jacobian. However, for larger problems the Broyden update becomes an increasingly bad approximation. In these cases, the authors speculate significant performance gains could be obtained by updating the Jacobian with a few strategically chosen function evaluations or directional derivatives rather than updating the entire Jacobian.

3.8 Conclusion

The computational problem of minimizing a sum of squares is a common problem, particularly in data fitting, that can be notoriously difficult. The difficulties often fall into one of two categories: algorithms easily become lost on broad flat plateaus or become sluggish as they must follow narrow canyons to the best fit. In this paper we have discussed several modifications to the standard least squares minimizer, the Levenberg-Marquardt algorithm.

By considering a geometric interpretation of the least-squares on a manifold in data space, we have shown that the Levenberg-Marquardt algorithm is an Euler approximation to geodesic flow on this manifold and have further refined the method by including the geodesic acceleration correction. We have shown that this correction helps the algorithm be both more robust to initial conditions, resulting in higher convergence and success rates, as well has decreasing the computational cost of finding the best fit as measured by the number of Jacobian evaluations. Amazingly the computational cost of the geodesic acceleration is small, comparable to a single function evaluation, and becomes negligible for large problems.

We have also suggested that accepting uphill steps in a controlled manner can also speed up the algorithm. When an algorithm is susceptible to becoming lost in parameter space, accepting uphill moves may exacerbate this problem, but when the algorithm must follow a narrow canyon to the best fit, the potential speed up of the bold method can be enormous.

Finally, we have suggested that providing a rank-deficient update to the Jacobian matrix can further reduce the computational cost of the Levenberg-Marquardt algorithm. Although the resulting algorithm has a tendency to become lost, like the bold acceptance method, it can be much more efficient when following a canyon to the best fit.

The performance of our several suggested improvements is summarized in figure 3.13. Notice that including geodesic acceleration has the tendency to improve success, fit quality, and speed. Including the bold acceptance and using Broydens update can be even more effective for speeding up the algorithm, although on some problems, particularly those with difficult start points, these algorithms are more likely to become lost. Which variation of the algorithm is most effective is likely to vary from problem to problem and whether the user is more concerned with fit quality or convergence speed. A good implementation of the Levenberg-Marquardt method should be flexible enough to allow the user to adapt the method to their specific needs.

We have provided an open source (FORTRAN and C versions) of the Levenberg-Marquardt algorithm together with our suggested improvements. Our implementation provides a simple way for each addition to be turned on or off, in addition to choosing among several schemes for updating the damping term λ. In this way, users have the tools to optimize the fitting process to more quickly and robustly find best fits based upon the needs of their individual models.

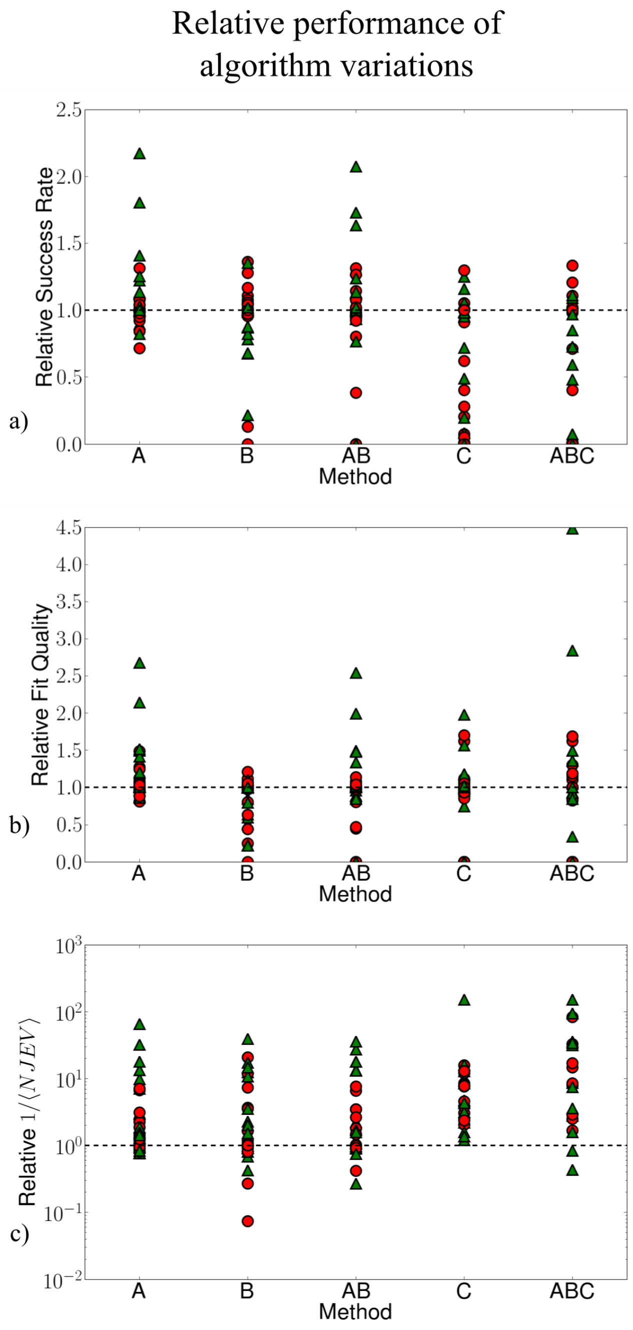


Figure 3.13: **Performance of several algorithms** The relative success rate (a), fit quality (b), and inverse NJEV (c) of several algorithms. The rates are each relative to the same algorithm without geodesic acceleration, bold, or a Broyden update. Columns represent different algorithms, with red dots denoting the $\lambda 2/3$ method while green triangles represent the $\Delta 2/3$ method on each of the 17 test problems. The label A indicates that geodesic acceleration was including, B denotes uphill steps were accepte with the bold acceptance criterion, and C indicates that the Jacobian was updated with Broyden's method.

Acknowledgements

The authors would like to thank S. Teukolsky for helpful conversation. This work was supported by NSF grant number DMR-1005479.

CHAPTER 4
SUPERHEATING FIELD OF SUPERCONDUCTORS WITHIN GINZBURG-LANDAU THEORY

4.1 Abstract[1]

We study the superheating field of a bulk superconductor within Ginzburg-Landau theory, which is valid near the critical temperature. We calculate, as functions of the Ginzburg-Landau parameter κ, the superheating field H_{sh} and the critical momentum k_c characterizing the wavelength of the instability of the Meissner state to flux penetration. By mapping the two-dimensional linear stability theory into a one-dimensional eigenfunction problem for an ordinary differential equation, we solve the problem numerically. We demonstrate agreement between the numerics and analytics, and show convergence to the known results at both small and large κ. We discuss the implications of the results for superconducting RF cavities used in particle accelerators.

4.2 Introduction

One of the primary features of superconductivity is the Meissner effect — the expulsion of a weak magnetic field from a bulk superconducting material [90]. For sufficiently large magnetic fields, the Meissner state becomes unstable, and the system undergoes a phase transition. The exact nature of the transition depends on the so-called Ginzburg-Landau parameter, $\kappa = \lambda/\xi$, where λ is the London

[1]This chapter has been published in *Physical Review B* with couathors Gianluigi Catelani and James P. Sethna.

penetration depth and ξ the superconducting coherence length. Type I superconductors, characterized by small κ, transition from the Meissner state into a normal metal state for magnetic fields above the thermodynamic critical field, H_c. Type II superconductors, with larger κ, instead transition into a superconducting state with vortices above the first critical field H_{c1}. This state is stable up to a second critical field H_{c2}, above which the metal becomes normal. For any superconductor, however, the Meissner superconducting state is metastable, persisting up to the superheating field H_{sh}, well above H_c or H_{c1} (for type I and II superconductors, respectively). The main goal of the present work is the calculation of H_{sh} as function of κ for superconductors near the critical temperature T_c where Ginzburg-Landau theory is applicable (we remind that within Ginzburg-Landau theory, the transition from type I to type II superconductors is at $\kappa = 1/\sqrt{2}$).

The metastability of the Meissner state is of interest in the design of resonance RF cavities in particle accelerators, where H_{sh} places a fundamental limit on the maximum accelerating field [80]. As type II superconducting materials are being considered in cavity designs, a precise calculation of H_{sh} in this regime is of value. One must note, however, that operating temperatures of superconducting RF cavities are well below the critical temperature T_c and that at these low temperatures Ginzburg-Landau theory is not quantitatively valid. The numerical techniques developed here are also being used within the Eilenberger formalism to address these lower temperatures [91]. Using this formalism, the limit $\kappa \to \infty$ was studied in Ref. [23] for arbitrary temperature.

Much work has already been done in calculating the superheating field within Ginzburg-Landau theory [31, 42, 60, 61, 37, 26, 24, 33]. The problem is formulated as follows: the superconductor occupies a half space with a magnetic field applied parallel to the surface. The order parameter and vector potential are functions of the distance from the surface and can be found by solving a boundary value problem of ordinary differential equations. The superheating field is then the largest magnetic field for which the corresponding solution is a local minimum of the free energy. For small values of κ, the superheating field corresponds to the largest magnetic field for which a nontrivial solution to the Ginzburg-Landau equations exist [33], as the instability does not break translational invariance. However, as κ increases the one-dimensional solution is unstable to two-dimensional perturbations, resulting in a lower estimate of $H_{\rm sh}$ as first shown in Ref. [42]. The task at hand is to find which perturbations destroy the Meissner state and at which value of the applied magnetic field they first become unstable.

The present stability analysis is more challenging than many such calculations, as the instability destabilizes an interface with a pre-existing depth-dependence of field and superconducting order parameter. In section 4.3, we map the partial differential equation for the unstable mode into an eigenvalue analysis for a family of one-dimensional ordinary differential equations. The equations for the zero eigenvalues describe the critical fluctuations to which the system is first unstable (as first derived in Ref. [60] and solved in Ref. [37]) This technique could be useful in a variety of other linear stability calculations [16, 15, 14], replacing thin interface approximations with a microscopic depth-dependent treatment of the destabilizing interface.

In this work we present a detailed numerical study of the problem of metastability of superconductors within Ginzburg-Landau theory, both for its intrinsic importance and as a prototypical illustration of the more general method – stripped, for example of the additional complexities of Eilenberger theory. Analytic treatments have been developed for limiting cases, such as very large [26] or small [33] values of the Ginzburg-Landau parameter κ which are far from real experimental cases. In section 4.4, we numerically explore the behavior of H_{sh} over a wide range of values of κ, demonstrating convergence to the known limiting cases. By comparing numerical values of H_{sh} to the analytic approximations, we show that these approximations give reasonably good estimates for H_{sh} even at values of κ which are far from their expected validity regimes. Additionally, we calculate accurately the critical value $\kappa_c \approx 1.1495$ which separates the regimes in which the instability is due to one- and two-dimensional perturbations and describe the behavior of the critical momentum near this transition. For $0.91 < \kappa < \kappa_c$, superconductors go unstable via uniform penetration of magnetic flux which then must dynamically reform into a vortex lattice. For $1/\sqrt{2} < \kappa < 0.91$ we find that $H_{\mathrm{sh}} > H_{c2}$, so at constant temperature the instability leads directly into the normal state (as for type-I superconductors with $\kappa < 1/\sqrt{2}$). Finally, we find that the instability wavelength has no immediate connection with the vortex lattice spacing.

The paper is organized as follows: in the next section we present the Ginzburg-Landau free energy and the differential equations to be studied for the stability analysis. In Sec. 4.4 we give some details about the numerical calculations and our main results. In Sec. 4.5 we discuss the implications of the results for accelerator cavity design and outline future research directions. In Appendices we derive analytic formulas, valid at large κ, which we compare against the numerics.

4.3 Ginzburg-Landau theory and stability analysis

The calculation of H_{sh} is a linear stability analysis of the coupled system of superconducting order parameter and vector potential. For a given configuration, we study the stability to arbitrary two-dimensional perturbations by considering the second variation of the free energy: if the second variation is positive definite for all possible perturbations then the solution is (meta)stable. The second variation can be expressed as a Hermitian operator acting on the perturbations, so it is sufficient to show that the eigenvalues of this operator are all positive. By expanding the perturbations in Fourier modes parallel to the surface, the eigenvalue problem can once again be translated into a boundary value problem of an ordinary differential equation. The eigenvalues now depend upon the wave-number of the Fourier mode, but can otherwise be solved in the same way as the Ginzburg-Landau equations. The superheating field is then the largest applied magnetic field for which the smallest eigenvalue is positive for all Fourier modes. In this section we outline the derivation of the relevant equations for the Ginzburg-Landau free energy using the method described above.

The Ginzburg-Landau free energy for a superconductor occupying the half space $x > 0$ in terms of the magnitude of the superconducting order parameter f and the gauge-invariant vector potential $\mathbf{q}$ is given by

$$\begin{aligned} \mathcal{F}[f, \mathbf{q}] &= \int_{x>0} d^3r \Big\{ \xi^2 (\nabla f)^2 + \frac{1}{2}(1-f^2)^2 \\ &\qquad + f^2 \mathbf{q}^2 + (\mathbf{H}_a - \lambda \nabla \times \mathbf{q})^2 \Big\}, \end{aligned} \tag{4.1}$$

where $\mathbf{H}_a$ is the applied magnetic field (in units of $\sqrt{2}H_c$), ξ is the Ginzburg-Landau coherence length, and λ is the penetration depth. Note that after choosing the unit of length, the only remaining free parameter in the theory is the ratio of these two characteristic length scales, the Ginzburg-Landau parameter $\kappa = \lambda/\xi$. The magnetic field inside of the superconductor is given by $\mathbf{H} = \lambda \nabla \times \mathbf{q}$.

We take the applied field to be oriented along the z-axis $\mathbf{H}_a = (0, 0, H_a)$, and the order parameter $f = f(x)$ to depend only on the distance from the superconductor's surface. We have assumed that the order parameter is real and further parametrize the vector potential as $\mathbf{q} = (0, q(x), 0)$, which fixes the gauge. The Ginzburg-Landau equations that extremize $\mathcal{F}$ with respect to f and $\mathbf{q}$ are

$$\begin{aligned} &\xi^2 f'' - q^2 f + f - f^3 = 0, \\ &\lambda^2 q'' - f^2 q = 0, \end{aligned} \tag{4.2}$$

and with our choices $H = \lambda q'$. Hereafter we use primes to denote derivatives with respect to x.

The boundary conditions at the surface derive from the requirement that the magnetic field be continuous, $q'(0) = H_a/\lambda$, and that no current passes through the boundary, $f'(0) = 0$. We also require that infinitely far from the surface the sample is completely superconducting with no magnetic field, giving us $f(x) \to 1$ and $q(x) \to 0$ as $x \to \infty$. In the limits $\kappa \to 0$ and $\kappa \to \infty$, Eqs. (4.2) can be explicitly solved perturbatively, see Ref. [33] and Appendix C.1, respectively. For arbitrary κ they can be solved numerically via a relaxation method, as we discuss in Sec. 4.4.

For a given solution $(f, \mathbf{q})$ we consider the second variation of $\mathcal{F}$ associated with small perturbations $f \to f + \delta f$ and $\mathbf{q} \to \mathbf{q} + \delta\mathbf{q}$ given by

$$\begin{aligned} \delta^2 \mathcal{F} \;=\; \int_{x>0} & d^3 r \Big\{ \xi^2 (\nabla \delta f)^2 + 4 f \delta f \mathbf{q} \cdot \delta\mathbf{q} + f^2 \delta\mathbf{q}^2 \\ & (3f^2 + \mathbf{q}^2 - 1)\delta f^2 + \lambda^2 (\nabla \times \delta\mathbf{q})^2 \Big\}. \end{aligned} \tag{4.3}$$

If the expression in Eq. (4.3) is positive for all possible perturbations, then the solution is stable. Since our solution $(f, \delta\mathbf{q})$ depends only on the distance from the boundary (and is therefore translationally invariant along the y and z directions), we can expand the perturbation in Fourier modes parallel to the surface. As shown in Ref. [60], we can restrict our attention to perturbations independent of z and write

$$\begin{aligned} \delta f(x,y) &= \delta\tilde{f}(x)\cos ky, \\ \delta\mathbf{q}(x,y) &= (\delta\tilde{q}_x \sin ky, \delta\tilde{q}_y \cos ky, 0), \end{aligned} \tag{4.4}$$

where k is the wave-number of the Fourier mode. The remaining Fourier components (corresponding to replacing cos $\rightarrow$ sin and vice-versa in Eq. 4.4) are redundant as they decouple from those given in Eq. 4.4 and satisfy the same differential equations derived below.

After substituting into the expression (4.3) for the second variation and integrating by parts, we arrive at

$$\begin{aligned} \delta^2\mathcal{F} \;=\; & \int_0^\infty dx \begin{pmatrix} \delta\tilde{f} & \delta\tilde{q}_y & \delta\tilde{q}_x \end{pmatrix} \\ & \begin{pmatrix} -\xi^2\frac{d^2}{dx^2} + q^2 + 3f^2 + \xi^2k^2 - 1 & 2fq & 0 \\ 2fq & -\lambda^2\frac{d^2}{dx^2} + f^2 & -\lambda^2 k\frac{d}{dx} \\ 0 & \lambda^2 k\frac{d}{dx} & f^2 + \lambda^2k^2 \end{pmatrix} \\ & \begin{pmatrix} \delta\tilde{f} \\ \delta\tilde{q}_y \\ \delta\tilde{q}_x \end{pmatrix}. \end{aligned} \tag{4.5}$$

The matrix operator in Eq. (4.5) is self-adjoint, and the second variation will be positive definite if its eigenvalues are all positive. In the eigenvalue equations for this operator, the function $\delta\tilde{q}_x$ can be solved for algebraically. The resulting differential equations for $\delta\tilde{f}$ and $\delta\tilde{q}_y$ are

$$-\xi^2\delta\tilde{f}'' + (3f^2 + q^2 - 1 + \xi^2 k^2)\delta\tilde{f} + 2fq\delta\tilde{q}_y \quad = \quad E\delta\tilde{f}, \tag{4.6}$$

and

$$-\lambda^2\frac{d}{dx}\left[\frac{f^2 - E}{f^2 + \lambda^2 k^2 - E}\delta\tilde{q}_y'\right] + f^2\delta\tilde{q}_y + 2fq\delta\tilde{f} = E\delta\tilde{q}_y, \tag{4.7}$$

where E is the stability eigenvalue. Note that by decomposing in Fourier modes, we have transformed the two-dimensional problem into a one-dimensional eigenvalue problem. Numerically, it can be solved by the same relaxation method as the Ginzburg-Landau equations – see Sec. 4.4. The boundary conditions associated with the eigenvalue equations derive from the same physical requirements previously discussed: we require $\delta\tilde{f}'(0) = 0$, since no current may flow through the boundary, and $\delta\tilde{q}_y'(0) = 0$, since the magnetic field must remain continuous. Additionally, we require $\delta\tilde{f}(x) \to 0$ and $\delta\tilde{q}_y(x) \to 0$ as $x \to \infty$. There is also an arbitrary overall normalization, which we fix by requiring $\delta\tilde{f}(0) = 1$.

The stability eigenvalue will depend on the solution of the Ginzburg-Landau equations, i.e., the applied magnetic field H_a, and the Fourier mode k under consideration. The problem at hand is to find the applied magnetic field and Fourier mode for which the smallest eigenvalue first becomes negative, which is the case if the following two conditions hold:

$$E = 0\,, \qquad \frac{dE}{dk} = 0. \tag{4.8}$$

The value of the magnetic field at which these conditions are met is the superheating field H_{sh}, and the corresponding wave-number is known as the critical momentum k_c. In the next section we discuss in more detail the numerical approach used to calculate these two quantities.

4.4 Numerical Results

As explained in the previous section, the calculation of the superheating field comprises two main steps: (1) solving the Ginzburg-Landau equations (4.2) and (2) solving the eigenvalue problem (4.6)-(4.7) with conditions (4.8). To solve these equations we employ a relaxation method. The basic scheme is to replace the ordinary differential equations with a set of finite difference equations on a grid. From an initial guess to the solution, the method iterates using Newton's method to relax to the true solution [83]. The grid is chosen with a high density of points near the boundary, with the density diminishing approximately as the inverse distance from the boundary. This is similar to the scheme used by Dolgert *et al.* [33] to solve the Ginzburg-Landau equations for type I superconductors.

For κ near the type I/II transition, the relaxation method typically converges without much difficulty. In the limiting cases that κ becomes either very large or small, however, the grid spacing must be chosen with care to achieve convergence. The eigenfunction equations are particularly sensitive to the grid choice. This is not surprising, since in either limit there are two well-separated length scales. For example, using units $\lambda = 1$ and $\xi = 1/\kappa$, we find that a grid with density

$$\rho(x) = \frac{150\kappa}{1 + 25\kappa x} \tag{4.9}$$

leads to convergence for κ as high as 250. The grid points are then be generated recursively $x_{i+1} = x_i + 1/\rho(x_i)$ with $x_0 = 0$. We find that if the grid is not sufficiently sparse at large x, the relaxation method fails, presumably due to rounding errors. On the other hand if it is too sparse, the finite difference equations poorly approximate the true differential equation. Fortunately, the method converges quickly, allowing us to explore the density by trial and error, as we have done to get Eq. 4.9.

In solving Eq. 4.2, if a sufficiently large value for the applied magnetic field is used, there may not be a nonzero solution to the Ginzburg-Landau equations and the relaxation method will often fail to converge, indicating that the proposed H_a is above the actual superheating field. In practice, therefore, it is more convenient to replace the boundary condition $q'(0) = \lambda H_a$ with a condition on the value of the order parameter $f(0) = A$, which then implicitly defines the applied magnetic field as a function of A, $H_a(A)$. This has the advantage that $H_a(A)$ is a differentiable function of A, as is the stability eigenvalue, improving the speed and accuracy of the search for the superheating field. The drawback to this approach is that $H_a(A)$ is not single-valued, with an unstable branch of solutions as illustrated in Fig. 4.1. For the problem at hand, this turns out to be straightforward to address since we determine the stability of each solution in the second step. To achieve the conditions in Eq. (4.8), we vary both the Fourier mode, k, and the value of the order parameter at the surface A.

The results of the procedure described above are summarized in Figs. 4.2-4.5, where we also compare them with analytical estimates which, for large values of κ, are derived in Appendices. In Fig. 4.2 we plot the numerically calculated superheating field as a function of κ (solid line). The vertical line at $\kappa_c \simeq 1.1495$ separates the regimes of one-dimensional (1D, $k = 0$) and two-dimensional (2D, $k \neq 0$) critical perturbations. We have checked the value of κ_c both by assuming 2D perturbations and finding when their critical momentum goes to zero and by assuming 1D perturbation and finding when the coefficient of the term quadratic in momentum in the second variation of the free energy vanishes; these methods lead to the same value within our numerical accuracy of 10^{-4}. Our value of κ_c is higher than previous estimates, which ranged from 0.5 [60] to 1.10 [37] and 1.13($\pm$0.05) [26]. (The estimate in Ref. [37] also comes from a numerical solution of

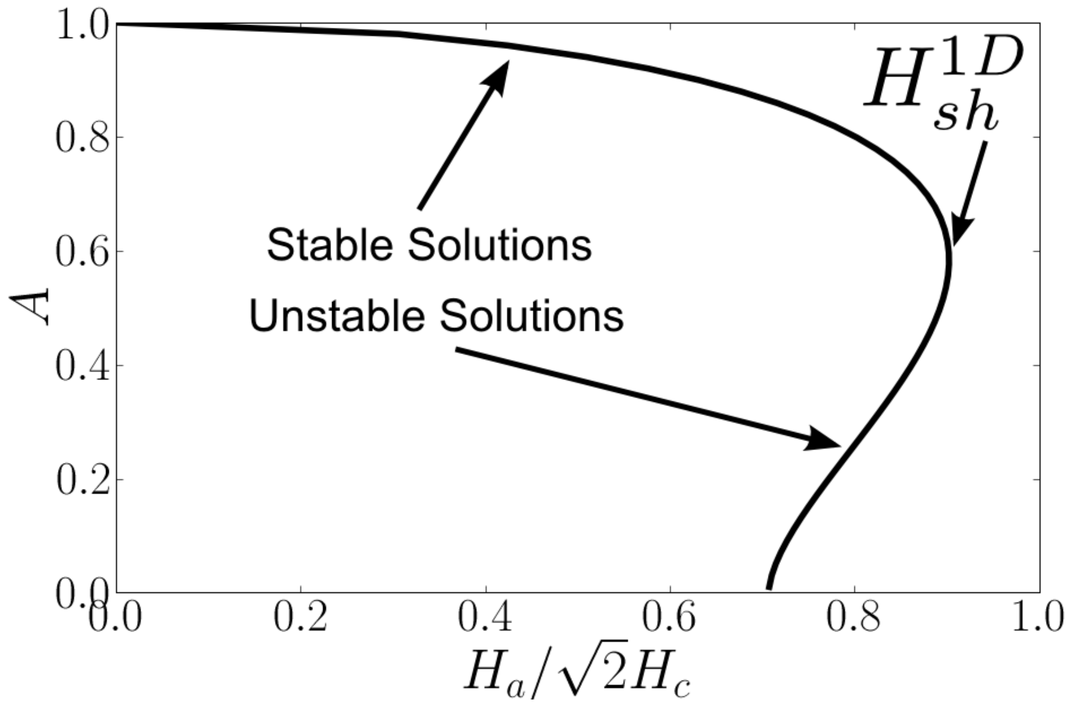


Figure 4.1: Solving for the 1-D Ginzburg-Landau superheating field
Solving for the 1-D Ginzburg-Landau superheating field. By fixing value of the order parameter at the surface, we implicitly define the applied magnetic field $H_a(A)$ at the surface. This definition produces a branch of unstable solutions, but guarantees that our equations will have a solution for all guesses of A. The "nose" of the curve occurs at the H_{sh}^{1D}, the superheating field ignoring two-dimensional fluctuations, and is the largest H_a for which a nontrivial solution to Eq. (4.2) can be found. This example was calculated for $\kappa = 1$, and $H_{\text{sh}}^{1D} \approx 0.9$.

the same differential equations, although the accuracy is there limited, presumably, by using a shooting method to find the solution.) κ_c is larger than the boundary $\kappa = 1/\sqrt{2}$ separating type I from type II superconductors. Type II superconductors for which $\kappa < \kappa_c$ become unstable via a spatially uniform invasion of magnetic flux. Additionally, we find that superheated type II superconductors with $\kappa < 0.9192$ can transition directly into the normal state since the corresponding H_{sh} is larger than the second critical field H_{c2}.

For $\kappa < \kappa_c$ the instability is due to 1D perturbations. In this regime, the Padé

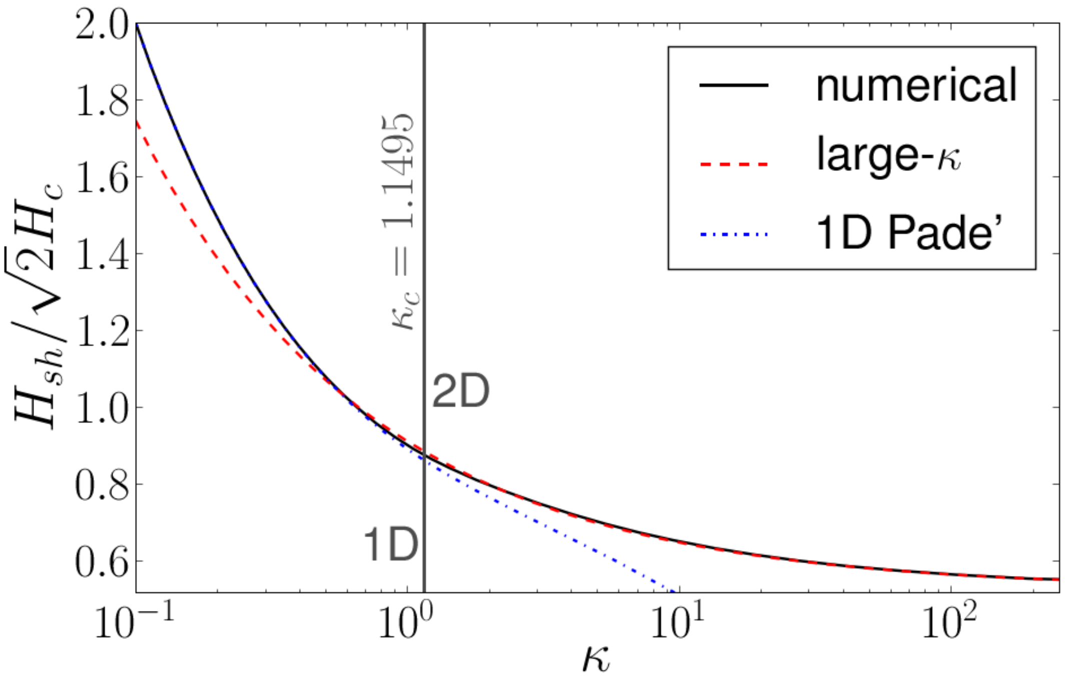


Figure 4.2: Numerical H_{sh} and analytical approximation
(Color online) Numerically calculated H_{sh} and corresponding analytical approximations [Eqs. (4.10) and (4.11)] versus the Ginzburg-Landau parameter κ.

approximate

$$\frac{H_{\mathrm{sh}}(\kappa)}{\sqrt{2}H_c} \approx 2^{-3/4}\kappa^{-1/2}\frac{1 + 4.6825120\kappa + 3.3478315\kappa^2}{1 + 4.0195994\kappa + 1.0005712\kappa^2} \tag{4.10}$$

derived in Ref. [33] (dot-dashed line) gives a good approximation to the actual H_{sh}, with deviation of less than about 1.5 %. In the opposite case $\kappa > \kappa_c$, 2D perturbations are the cause of instability and the superheating field is approximately given by [26] (dashed line)

$$\frac{H_{\mathrm{sh}}(\kappa)}{\sqrt{2}H_c} \approx \frac{\sqrt{10}}{6} + \frac{0.3852}{\sqrt{\kappa}}. \tag{4.11}$$

Equation (4.11), derived in Appendix C.2, is also a good approximation, deviating at most about 1 % from the numerics. Therefore, our numerics show that the simple analytical formulas for H_{sh} in Eqs. (4.10) and (4.11) can be used to accurately estimate the superheating field for arbitrary value of the Ginzburg-Landau param-

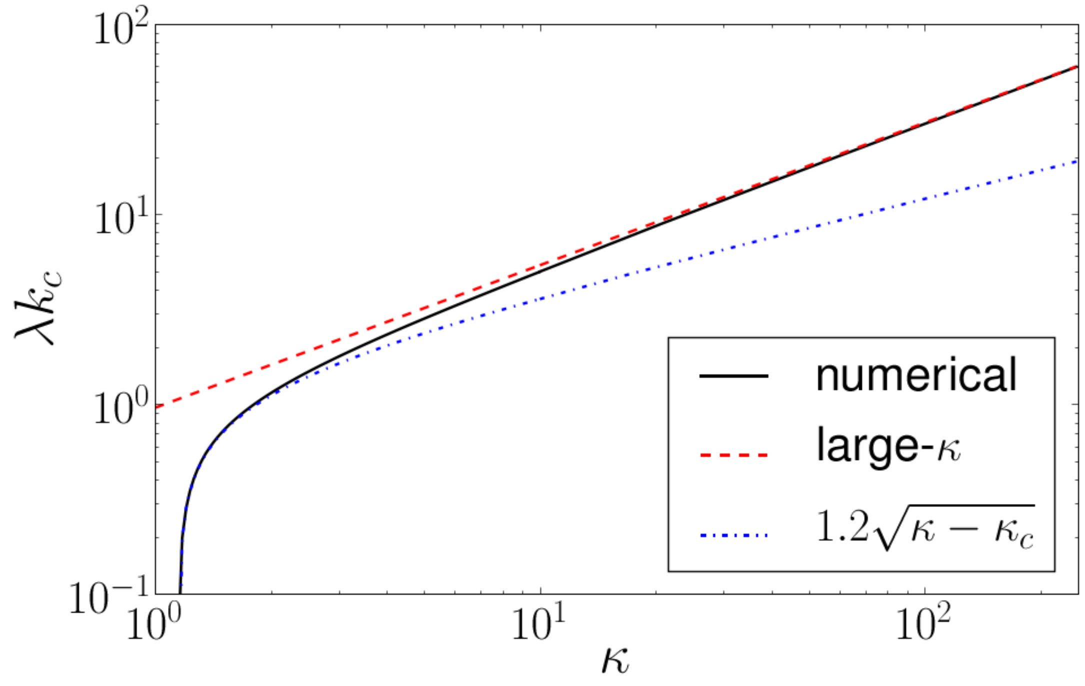


Figure 4.3: Critical momentum vs. κ
(Color online) Comparison between the numerical and asymptotic critical momentum k_c, Eq. (4.13). The approximate behavior of k_c near κ_c (dot-dashed) is given in Eq. (4.12).

eter, when used in their respective validity regions. We note that the numerical estimate of H_{sh} in Ref. [37] differs from our own by a few percent, again presumably due to the limitations of their method. We believe the numerical results presented here to be reliable by comparing with asymptotic solutions of the order parameter, vector potential, and critical fluctuations for very large κ.

In Fig. 4.3 we show the numerical result for the critical momentum k_c versus κ. We see that $k_c \to 0$ as $\kappa \to \kappa_c$ from above. Near κ_c, the behavior of k_c is reminiscent of that of an order parameter near a second-order phase transition:

$$k_c \simeq 1.2\sqrt{\kappa - \kappa_c}, \tag{4.12}$$

where the prefactor has been estimated by fitting the numerics. The dashed line

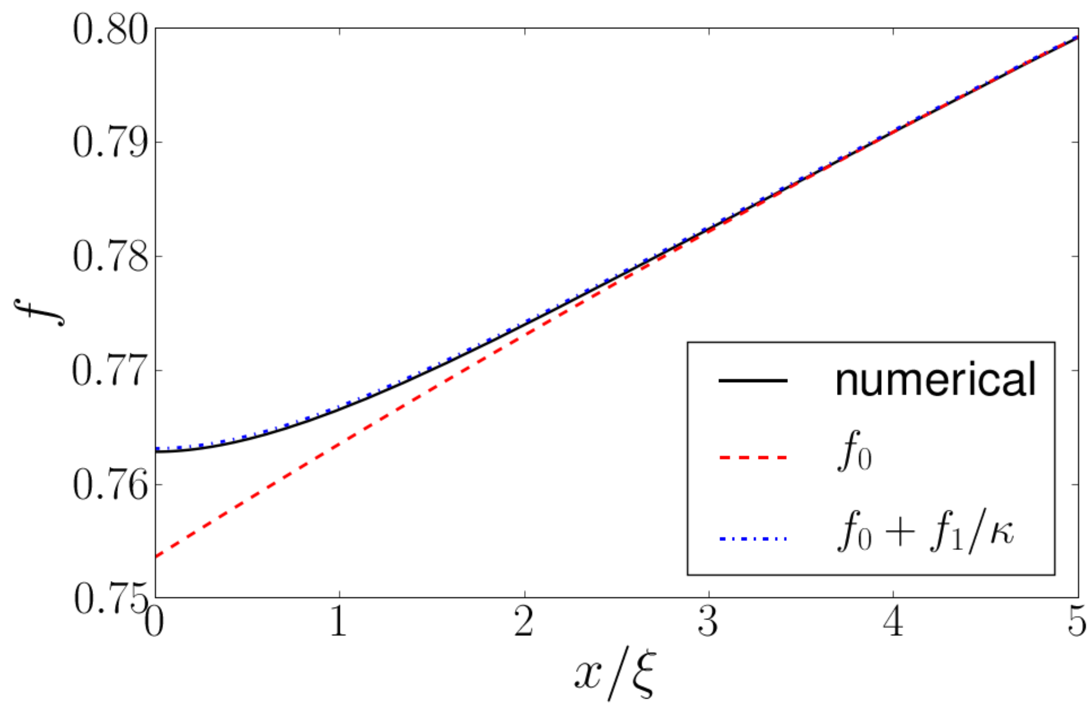


Figure 4.4: Profile of the order parameter at H_{sh}
(Color Online) Profile of the order parameter at H_{sh} for $\kappa = 50$ together with analytic approximations given in Eqs. (C.2) and (C.15).

is the asymptotic formula [26] (see also Appendix C.2)

$$\lambda k_c \approx 0.9558\kappa^{3/4} \tag{4.13}$$

which captures correctly the large-κ behavior.

In Fig. 4.4 we present a typical solution for the order parameter near the surface at $H = H_{\text{sh}}$ for a large value of κ, along with the analytic approximations presented in Appendix C.1. The zeroth order approximation f_0, Eq. (C.2), fails near the surface, as it does not satisfy the boundary condition $f'(0) = 0$. On the other hand, including the first order correction in $1/\kappa$, Eq. (C.15), leads to excellent agreement with the numerics. Finally, in Fig. 4.5 we show a typical example of the depth dependence of the perturbation $\delta\tilde{q}_y$ at the critical point where the solution first becomes unstable. We find again good agreement between numerical and

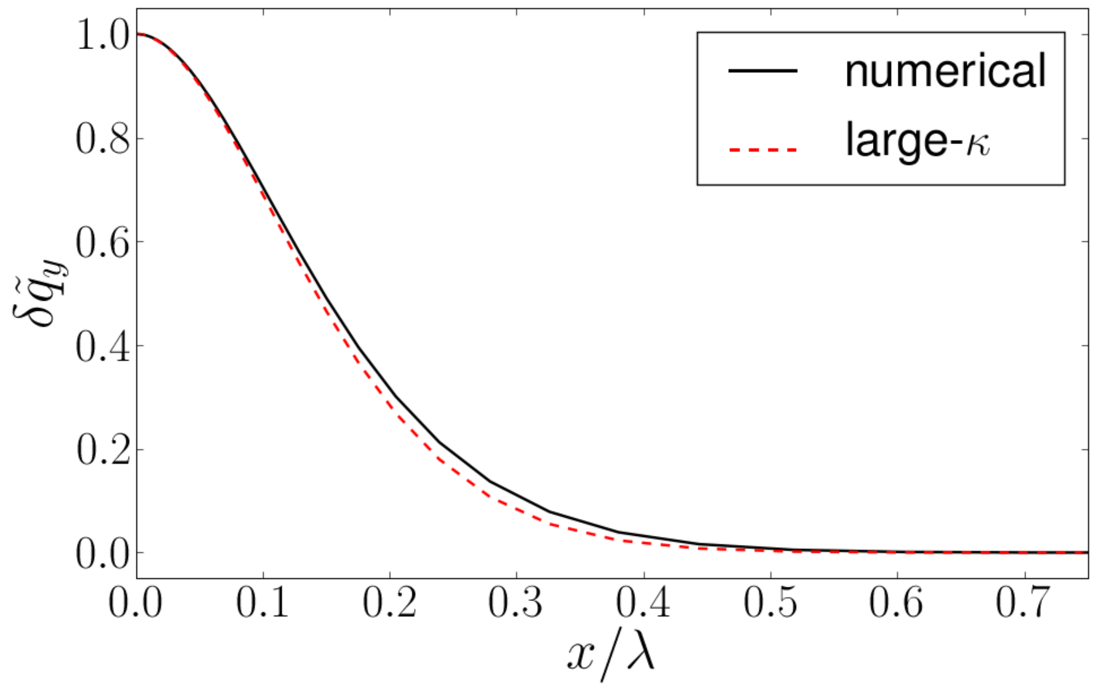


Figure 4.5: Profile of the critical perturbation
(Color online) Numerical profile (solid line) of the critical perturbation $\delta\tilde{q}_y$ determining H_{sh} for $\kappa = 50$ compared to the large-κ perturbative result (dashed), Eq. (C.37).

perturbative calculations.

It is interesting to compare the wavelength of the critical perturbation, $2\pi/k_c$, with the Abrikosov spacing a for the arrangement of vortices at the superheating field: the naive expectation is that the initial flux penetrations represent nuclei for the final vortices. Kramer argues that this picture is incorrect since the initial flux penetrations do not have supercurrent singularities and do not carry a fluxoid quantum [60].

We find the numerical discrepancies between the two lengths further support Kramer's argument. In the weakly type-II regime ($\kappa \sim 1$), the initial flux penetration is from infinitely long wavelengths ($k_c = 0$); in contrast, the final vortex state

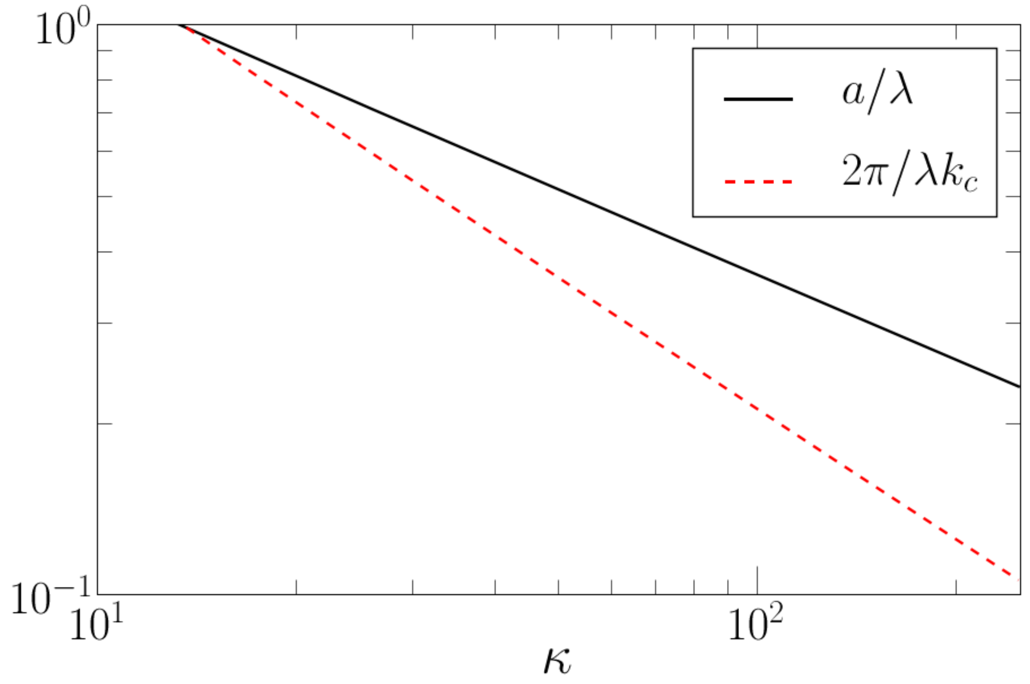


Figure 4.6: Critical momentum and Abrikosov vortex spacing
(Color online) The wavelength of the critical perturbation ($2\pi/k_c$) and the Abrikosov vortex spacing (a) calculated at the superheating field both vanish at large κ, although the former diminishes much more quickly.

has a very high density, since $H_{\text{sh}} \sim H_{c2}$ [2]. In the strongly type-II limit ($\kappa \to \infty$) both the inverse momentum and the Abrikosov spacing (evaluated at the superheating field) vanish, but at different rates, with $1/k_c \sim \kappa^{-3/4}$ while the Abrikosov spacing $a \sim \kappa^{-1/2}$ at H_{sh} [3], see Fig. 4.6. These results suggest that there is no immediate connection between the initial penetration and the final vortex array. A dynamical simulation could explore the transition between the initial penetration and the final vortex state, similar to that done by Frahm *et. al* for the transition from the normal state to the vortex state [38].

[2]We remind that in our units $H_{c2} = \kappa$

[3]The vortex spacing a is related to the induction B as $a/\lambda \sim 1/(\kappa B)^{1/2}$, and at intermediate fields H such that $H_{c1} \ll H \ll H_{c2}$, $B \approx H$ – see the reference in the previous footnote. Since at large κ the superheating field tends to a constant, we find $a/\lambda \sim \kappa^{-1/2}$.

4.5 Summary and outlook

In this paper, we have numerically calculated within Ginzburg-Landau theory the superheating field $H_{\rm sh}$ of superconductors. We have considered values of the Ginzburg-Landau parameter κ spanning over three orders of magnitude; this has enabled us to show that the analytic approximations in Eqs. (4.10) and (4.11) are in good agreement with the numerical results, see Fig. 4.2. For κ larger than the critical value $\kappa_c \simeq 1.1495$, the critical perturbations have two-dimensional character; their critical momentum k_c is plotted in Fig. 4.3, where we also show agreement between numerics and the asymptotic formula for k_c at large κ, Eq. (4.13). The above results have been obtained by mapping the linear stability threshold onto an eigenfunction problem. The technique of mapping the linear stability problem onto a one-dimensional eigenfunction problem is potentially a useful technique, and we hope others find useful applications of the methods described here.

One of the primary motivations for this work is the application to RF cavities in particle accelerators, where the maximum accelerating field is limited by $H_{\rm sh}$. While the results presented here provide good estimates of $H_{\rm sh}$ for many materials of interest near the critical temperature T_c, we emphasize that the operating temperature of these cavities are typically well below T_c, where Ginzburg-Landau theory is not quantitatively accurate. The techniques presented here can be applied to Eilenberger theory to more accurately determine $H_{\rm sh}$ at low temperatures. The Eilenberger approach has already been used [23] to evaluate $H_{\rm sh}(T)$ at any temperature in the infinite κ limit for clean superconductors, and work is in progress to address low temperatures for finite κ [91].

Acknowledgements

The authors would like to thank Hasan Padamsee, Georg Hoffstaetter, and Matthias Liepe for helpful discussion. This work was supported by NSF grant number DMR-0705167 (MKT & JPS), by the Department of Energy under contract de-sc0002329 (MKT), and by Yale University (GC).

CHAPTER 5

SUPERHEATING FIELD OF SUPERCONDUCTORS WITHIN EILENBERGER THEORY

5.1 Abstract[1]

We study the superheating field of a bulk superconductor within Eilenberger theory, which is valid at all temperatures. We calculate as functions of the Ginzburg-Landau parameter κ and the reduced temperature $t = T/T_c$ the superheating field H_{sh} and the critical momentum k_c describing the wavelength of the unstable perturbations to flux penetration. We demonstrate agreement with known results near T_c calculated from Ginzburg-Landau theory, but find discrepancies with the temperature-dependent results calculated for Eilenberger theory at infinite-κ. We attribute this discrepancy to lack of convergence of the integrals over the Fermi-surface.

5.2 Introduction

A bulk superconductor, when exposed to a sufficiently large magnetic field, experiences a "quench", that is a sudden penetration of the bulk material by magnetic flux. The resulting state of the material then depends upon the Ginzburg-Landau parameter $\kappa = \lambda/\xi$, the ratio of the London penetration depth to the superconducting coherence length. Type-I superconductors ($\kappa < 1/\sqrt{2}$) transition to a normal metal state while type-II materials ($\kappa > 1/\sqrt{2}$) form a mixed state in which the

[1]The current chapter will be submited for publication once the convergence problems at low T and high κ are addressed.

superconductivity is preserved interspersed with an array of magnetic flux tubes. This transition, apart from being theoretically interesting, is immediately relevant to the design of particle accelerators, where superconductors are used in the construction of accelerating resonance cavities and in which the maximum accelerating field is limited by the quench of the superconductor.

The magnetic field at which the normal or mixed state becomes energetically favorable is the thermodynamic critical H_c field for type-I superconductors or the so-called lower critical field H_{c1} for type-II superconductors. The superconducting Meissner state may persist, however, as a metastable state up to a larger magnetic field known as the superheating field $H_{\mathrm{sh}} > H_c, H_{c1}$. In contemporary superconducting RF cavities superheating is observed with fields well above H_{c1}, approaching the theoretical limits calcualted from Ginzburg-Landau theory extrapolated to lower temperatures[98]. It is unknown, however, whether these results represent the theoretical upper limit of achievable gradients, primarily because operating conditions are far from the regime with known theoretical results. This paper seeks to help resolve this problem by providing a numerical calculation of the temperature dependence of H_{sh}.

The metastability of the Meissner state has been studied extensively within Ginzburg-Landau theory[31, 42, 60, 61, 37, 26, 24, 33]. The authors have recently presented a thorough numerical calculation of H_{sh} within Ginzburg-Landau theory for the complete range of κ together with an analytic, asymptotic analysis[92]. The predictions of Ginzburg-Landau theory are only quantitatively accurate near the critical temperature; superconducting Radio Frequency (RF) cavities are operated, however, far below T_c. The temperature dependence of H_{sh} for the infinite κ limit has also been addressed previously using Eilenberger theory[23]. In this paper we extend our previous work within Ginzburg-Landau theory and the large-κ limit of Eilenberger to a range of temperatures and κ values applicable to materials used or considered for use in RF cavities. Of particular interest is niobium (Nb) with $\kappa = 1$, the primary material used in RF resonance cavities. Niobium has a critical temperature of 9 K but is typically operated near 2 K in particle accelerators, giving $T/T_c = 0.22$. With other materials being considered as replacements for niobium, such as $\mathrm{Nb_3Sn}$ ($\kappa \approx 20$ and $T_c \approx 18$ K) and $\mathrm{MgB_2}$ (with $\kappa \approx 3.6$ and $T_c \approx 39$ K), an operating temperature of 2 K would yield $T/T_c = 0.11$ and 0.05 respectively, far from the range where Ginzburg-Landau theory can be trusted.

In this chapter we present numerical results for H_{sh} as a function of temeprature. In addition to the superheating field, we also calculate the wavenumber of the critical fluctuations k_c to which the superconducting state first becomes unstable. We also find the critical value of the Ginzburg-Landau parameter at which two-dimensional fluctuations first become important. For each of these results, we recover the results of Ginzburg-Landau theory for temperatures near T_c and discuss how the results depend on temperature. For lower temperatures, however, the results presented in this chapter have not converged and more numerical investigations are necessary to give accurate predictions.

In the subsequent sections we introduce the Eilenberger equations (section 5.3) and describe our numerical methods for their solution (section 5.4). In section 5.5 we summarize our results for weakly type-II materials (section 5.5.1) and strongly type-II materials (section 5.5.2) and discuss issues regarding convergence at low temperatures in section 5.5.3. In section 5.6 we compare our results to recent experiments measuring the temperature dependence of H_{sh} for superconducting niobium. We give concluding remarks in section 5.7.

5.3 Eilenberger equations

The Eilenberger theory of superconductivity is derived from the microscopic BCS theory under the assumption that the Fermi wavelength λ_F is the smallest length scale characterizing the system[35]. This approximation is usually valid for low-T_c, weakly-coupled superconductors for which the zero-temperature coherence length $\xi_0 \gg \lambda_F$. See, for example, the discussion in reference [23].

In this approximation, Eilenberger theory summarizes the BCS equations into normal and anomalous Greens funtions. The semi-classical equations for the anomalous Greens functions $f(\omega_n, \mathbf{n}, \mathbf{r})$ and $\bar{f}(\omega_n, \mathbf{n}, \mathbf{r})$ which depend on the Matsubara frequences $\omega_n = 2\pi T(n+1/2)$, the position $\mathbf{r}$, and the unit vector $\mathbf{n}$ on the Fermi surface are given by

$$\begin{aligned} \{\omega_n + \mathbf{n}\cdot[\nabla - i\mathbf{A}(\mathbf{r})]\}\, f(\omega_n, \mathbf{n}, \mathbf{r}) &= \Delta(\mathbf{r}) g(\omega_n, \mathbf{n}, \mathbf{r}) \\ \{\omega_n - \mathbf{n}\cdot[\nabla + i\mathbf{A}(\mathbf{r})]\}\, \bar{f}(\omega_n, \mathbf{n}, \mathbf{r}) &= \Delta^\dagger(\mathbf{r}) g(\omega_n, \mathbf{n}, \mathbf{r}). \end{aligned} \tag{5.1}$$

Here $\mathbf{A}$ is the vector potential and Δ is the superconducting order parameter. The normal Greens function g is given by the relation $g^2 + f\bar{f} = 1$. Differentiating this normalization condition gives an equation for $g(\omega_n, \mathbf{n}, \mathbf{r})$

$$2\mathbf{n}\cdot\nabla g(\omega_n, \mathbf{n}, \mathbf{r}) = \Delta^\dagger(\mathbf{r}) f(\omega_n, \mathbf{n}, \mathbf{r}) - \Delta(\mathbf{r})\bar{f}(\omega_n, \mathbf{n}, \mathbf{r}), \tag{5.2}$$

consistent with that originally derived by Eilenberger. The Greens functions equations are accompanied by self-consistent equations for Δ and $\mathbf{A}$, given by

$$\begin{aligned} &\Delta(\mathbf{r}) \log\frac{T}{T_c} + 2\pi T \sum_n \left[\frac{\Delta(\mathbf{r})}{\omega_n} - \int d\mathbf{n}\ \rho(\mathbf{n}) f(\omega_n, \mathbf{n}, \mathbf{r})\right] = 0 \\ &\nabla\times\mathbf{H} + \frac{i2\pi T}{\kappa_0^2}\sum_n \int d\mathbf{n}\ 3\mathbf{n}\ \rho(\mathbf{n}) g(\omega_n, \mathbf{n}, \mathbf{r}) = 0. \end{aligned} \tag{5.3}$$

In Eq. (5.3) $\rho(\mathbf{n})$ is the normal density of states for one spin at $\mathbf{n}$ normalized to $\int d\mathbf{n}\rho = 1$. In these equations we have used the same units as in reference [23].

One may derive a 'free energy' functional whose Euler-Lagrange equations are given by Eqs. (5.1, 5.3):

$$\begin{aligned}\Omega = \nu \int d^3r \Bigg\{ & [\mathbf{H}(\mathbf{r}) - \mathbf{H}_a]^2 + |\Delta(\mathbf{r})|^2 \log\left(\frac{T}{T_c}\right) \\ & \int (dn) \left[\frac{|\Delta(\mathbf{r})|^2}{\omega_n} - \Delta^\dagger(\mathbf{r}) f - \bar{f}\Delta(\mathbf{r}) - 2\omega_n(g-1) \right. \\ & \left. -g\mathbf{n}\cdot\left(\nabla \log\frac{f}{\bar{f}} - 2i\mathbf{A}(\mathbf{r})\right)\right]\Bigg\},\end{aligned} \tag{5.4}$$

where we have used the shorthand $\int (dn) = 2\pi T \sum_n \int d\mathbf{n}$. Note that the functional in Eq. (5.4) is not the thermodynamic potential, but for any given Δ and $\mathbf{A}$ gives the difference between the free energy of the superconducting and normal states. In evaluating Eq. (5.4), therefore, one should evaluate the Greens functions f, $\bar{f}$ and g using Eqs. (5.1)- (5.2) for a given Δ and $\mathbf{A}$.

The boundary conditions for Eqs. (5.1)-(5.3) are determined by the problem's geometry. We consider the same geometry as described in references [23, 95]. The bulk superconductor occupies an infinite halfspace $z > 0$ with a surface in the $x - y$ plane. An external magnetic field is applied parallel to the surface in the $\hat{\mathbf{x}}$ direction. We note that the order parameter can be assumed to be real, which essentially fixes the gauge choice. Under this assumption the vector potential has only components in the y-direction, $\mathbf{A} = A_y\hat{\mathbf{y}}$. The boundary condition on A_y at the surface is determined by the applied magnetic field, $\mathbf{H}_a = \nabla \times \mathbf{A}(z=0) = -A_y'(z=0)\hat{\mathbf{x}}$. We also require that infinitely far from the surface, the sample be superconducting, $A_y(z=\infty) = A_y'(z=\infty) = 0$. The boundary condition at $z = 0$ on the Greens function is specular reflection, $f(\omega, \mathbf{n}, 0) = f(\omega, \mathbf{m}, 0)$, where the reflected vector $\mathbf{m} = \mathbf{n} - 2(\mathbf{n}\cdot\hat{\mathbf{z}})\hat{\mathbf{z}}$. The same surface condition holds for $\bar{f}$ and g.

5.4 Numerical Methods

In order to calculate the superheating field using the Eilenberger theory, we proceed much in the same way as in reference [92] replacing the Ginzburg-Landau free energy with the Eilenberger potential in (5.4). In brief, for a given solution to Eqs. (5.1 - 5.3), one explores the local stability of the functional in (5.4) to two dimensional perturbations. By using a linear-stability technique, one can decompose the two dimensional fluctuations into Fourier modes perpendicular to the surface, removing the need to solve any partial differential equations. The increased complexity of the Eilenberger theory introduces many numerical challenges. In this section, we summarize our numerical methods for solving for H_{sh} using Eilenberger theory.

The nonlocal electrodynamics captured in the Eilenberger Greens functions lead us to use a different numerical approach than that used for the analogous calculation in Ginzburg-Landau. We employ a Galerkin method to solve Eqs. (5.1-5.3). In this approach we express the desired solutions as a linear combination of basis functions

$$\Delta(z) = \sum_i \hat{\Delta}_i \phi_i(z), \tag{5.5}$$

with similar expressions for each A_y, f, $\bar{f}$, and g, (where we have dropped the arguments of the functions for brevity). We choose basis function, $\phi_i(z)$, to be piecewise cubic hermite interpolating polynomials. This choice allows us to specify the solutions' value and first derivative at a series of nodal points, as illustrated in figure 5.1. Nodal points are chosen in a manner similar to that described in reference[92].

Eqs. (5.1) and (5.2) are solved for the values of $\hat{f}_i$, $\hat{\bar{f}}_i$ and $\hat{g}_i$ for a particular

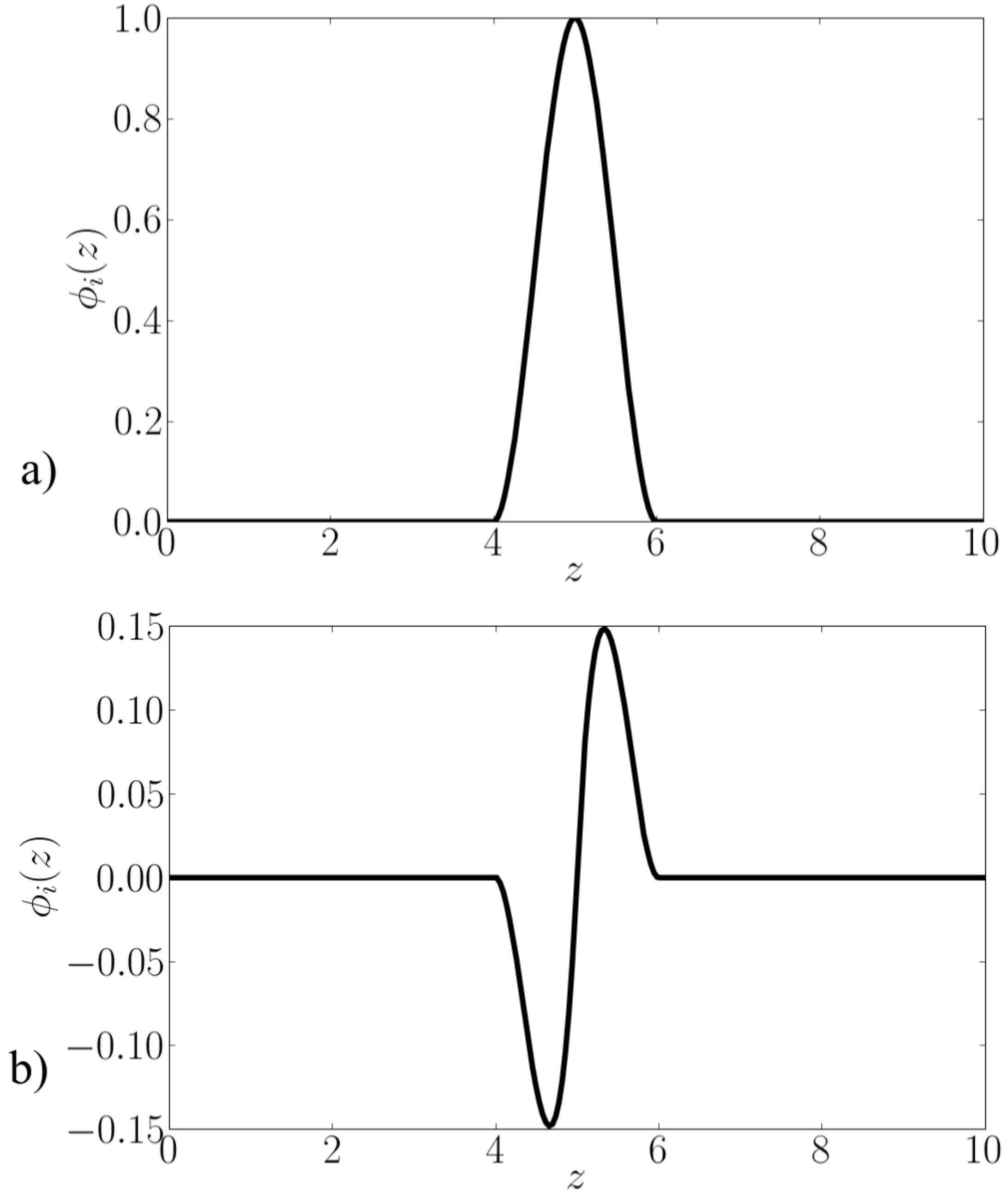


Figure 5.1: **Piecewise Cubic Hermite Interpolating Polynomials**. We choose basis functions that allow us to specify the function's value and first derivative at a series of nodal points. In this case, nodal points are chosen at integer values from zero to ten. The basis function in a) controls the function's value at the node $z = 5$ while the basis function in b) controls the function's derivative at this point.

configuration of the order parameter and vector potential $\hat{\Delta}_i$ and $\hat{A}_{yi}$, by multiplying the each equation by $\phi_j(\mathbf{r})$ and integrating over space. This results in a matrix equation

$$\begin{pmatrix} E_{ff} & E_{f\bar{f}} & E_{fg} \\ E_{\bar{f}f} & E_{\bar{f}\bar{f}} & E_{\bar{f}g} \\ E_{gf} & E_{g\bar{f}} & E_{gg} \end{pmatrix} \begin{pmatrix} \hat{f} \\ \hat{\bar{f}} \\ \hat{g} \end{pmatrix} = 0. \tag{5.6}$$

The nonzero matrix elements are given by

$$(E_{ff})_{ij} = \omega_n M_{ij} + \mathbf{n}\cdot\hat{\mathbf{z}} D_{ij} - i\mathbf{n}\cdot\hat{\mathbf{y}}\hat{A}_{yk} N_{ijk} \tag{5.7}$$

$$(E_{fg})_{ij} = -\hat{\Delta}_k N_{ijk} \tag{5.8}$$

$$\left(E_{\bar{f}\bar{f}}\right)_{ij} = \omega_n M_{ij}\mathbf{n}\cdot\hat{\mathbf{z}} D_{ij} - i\mathbf{n}\cdot\hat{\mathbf{y}}\hat{A}_{yk} N_{ijk} \tag{5.9}$$

$$\left(E_{\bar{f}g}\right)_{ij} = -\hat{\Delta}_k^\dagger N_{ijk} \tag{5.10}$$

$$(E_{gf})_{ij} = -\hat{\Delta}_k^\dagger N_{ijk} \tag{5.11}$$

$$\left(E_{g\bar{f}}\right)_{ij} = \hat{\Delta}_k N_{ijk} \tag{5.12}$$

$$(E_{gg})_{ij} = 2\mathbf{n}\cdot\hat{\mathbf{z}} D_{ij}, \tag{5.13}$$

$$\tag{5.14}$$

where we have introduced the matrices

$$M_{ij} = \int d\mathbf{r}\phi_i\phi_j \tag{5.15}$$

$$N_{ijk} = \int d\mathbf{r}\phi_i\phi_j\phi_k \tag{5.16}$$

$$D_{ij} = \int d\mathbf{r}\phi_i\phi_j' \tag{5.17}$$

$$\tag{5.18}$$

and used the convention that repeated indices should be summed. In practice it is necessary to considering only a linear combination of ϕ_i's that satisfy the boundary conditions.

The solution to Eqs. (5.1) and (5.2) corresponds to the nullspace of the matrix in Eq. (5.6) which can be found from a singular value decomposition. The scale is chosen to satisfy the normalization $g^2 + f\bar{f} = 1$. Notice that the matrices M_{ij}, N_{ijk}, and D_{ij} need only be calculated once for a set of ϕ's, greatly reducing the computational cost in solving Eqs. (5.1)- (5.3).

Having solutions to the Eilenberger equations, we solve the self consistent equations by a similar technique. Multiplying Eqs. (5.3) by ϕ_j and integrating over space give the nonlinear equations

$$\begin{aligned} \left[\hat{\Delta}_i \log\frac{T}{T_c} + 2\pi T \sum_n \left(\frac{\hat{\Delta}_i}{\omega_n} - \int d\mathbf{n}\rho(\mathbf{n})\hat{f}_i(\omega_n,\mathbf{n})\right)\right] M_{ij} &= 0 \qquad (5.19)\\ -\hat{A}_{yi} D^{(2)}_{ij} + \frac{i2\pi T}{\kappa_0^2}\sum_n \int d\mathbf{n}\; 3\mathbf{n}\; \rho(\mathbf{n})\hat{g}_i(\omega_n,\mathbf{n}) M_{ij} &= 0. \qquad (5.20)\end{aligned}$$

where we have introduced the matrix $D^{(2)}_{ij} = \int dz \phi_j \phi_i''$. These equations can be solved iteratively using a root-finding method (note that $\hat{f}$ and $\hat{g}$ are now nonlienar functions of $\hat{\Delta}$ and $\hat{\mathbf{A}}$).

For most values of the applied magnetic field convergence of the self-consistent equations occurs within a few iterations of Newton's method. However, for magnetic fields larger than H^{1D}_{sh}, the superheating field ignoring two-dimensional fluctuations, the self-consistent equations have no solution and the method fails altogether. For applied magnetic fields less than, but very near H^{1D}_{sh} the method converges slowly as the Jacobian matrix (Hessian matrix of the free energy) is very nearly singular. For smaller values of κ, $H_{\text{sh}} = H^{1D}_{\text{sh}}$, so this slowing down becomes a serious bottleneck. To remedy the problem, we use a technique due to Christopher Myers[76] in which we add a degree of freedom to the self-consistent equations associated with the applied magnetic field and an additional root associated with the smallest eigenvalue of the Hessian. In this way, the root-finding algorithm is able to quickly converge to the appropriate solution for H_{sh}^{1D}.

To evaluate the Fermi-surface integrals in Eqs. (5.19) and (5.20), we follow the method described in reference [4]. Essentially, we convert the integral into a weighted sum similar to Gaussian quadrature in one dimension. Specifically, if θ and ϕ are the spherical-polar angles associated with the unit vector $\mathbf{n}$, then $\cos\theta_i = \mathbf{n}\cdot\hat{\mathbf{z}}$ are chosen to be the Gauss-Legendre nodes on $[-1, 1]$. If there are m such points, we discretize ϕ with $2m$ points evenly spaced on $[-\pi, \pi]$. We therefore solve the Eilenberger equations for $2m^2$ values of $\mathbf{n}$. Since these solutions are found in pairs to satisfy boundary conditions we only require m^2 singular value decompositions. Furthermore, the symmetries of the Eilenberg equations and the geometry under considerations allows us to further reduce this number by a factor of four. We find that for a uniform density of states ρ, we achieve good convergence for $m = 2$ near T_c; however, at low temperatures many points are necessary as we discuss in section 5.5.3.

We do not solve the Eilenberger equations for each Matsubara frequency. Instead we find the Greens functions for a few, logarithmically spaced values of ω and interpolate to find intermediate values. At low temperatures, the spacing between Matsubaru frequencies, $\Delta\omega = 2\pi T$ becomes very small and so this approximation is very good. On the other hand, at high temperatures, the solution converges very quickly in Matsubaru frequencies, so that it is not necessary to solve for many frequencies. In practice, we never need to explicitly solve the Eilenberger equations at more than eight values of ω.

With self-consistent solutions for the vector potential and order parameter, we are prepared to explore its stability to fluctuations using the procedure described in reference[92]. The Eilenberger formalism presents one major complication to the

the problem absent from the Ginzburg-Landau case, however, from the nonlocal effects captured in the response of the Greens functions. A fluctuation in the order parameter or vector potential at one position will produce a response in the Greens functions at all other locations. In the case of the Ginzburg-Landau free energy, the stability analysis can be mapped onto an eigenvalue problem of a linear, differential operator. Because of the nonlocal effects in Eilenberger theory, the stability analysis is mapped onto an eigenvalue problem of a linear, nonlocal integral-differential operator. By employing a Galerkin method, we are able to approximate this eigenvalue problem as a matrix eigenvalue problem.

In order to explore the metastability, it is necessary to show that the second variation of the thermodynamic potential is positive definite for all infinitesimal fluctuations. In general this takes the form:

$$\delta^2\Omega = \int d\mathbf{r} \int d\mathbf{r}' \begin{pmatrix} \delta\Delta(\mathbf{r}) & \delta\mathbf{A}(\mathbf{r}) \end{pmatrix} \begin{pmatrix} \frac{\delta^2\Omega}{\delta\Delta(\mathbf{r})\delta\Delta(\mathbf{r}')} & \frac{\delta^2\Omega}{\delta\Delta(\mathbf{r})\delta\mathbf{A}(\mathbf{r}')} \\ \frac{\delta^2\Omega}{\delta\mathbf{A}(\mathbf{r})\delta\Delta(\mathbf{r}')} & \frac{\delta^2\Omega}{\delta\mathbf{A}(\mathbf{r})\delta\mathbf{A}(\mathbf{r}')} \end{pmatrix} \begin{pmatrix} \delta\Delta(\mathbf{r}') \\ \delta\mathbf{A}(\mathbf{r}') \end{pmatrix}. \tag{5.21}$$

It therefore necessary and sufficient to show that the operator

$$L(\mathbf{r},\mathbf{r}') = \begin{pmatrix} \frac{\delta^2\Omega}{\delta\Delta(\mathbf{r})\delta\Delta(\mathbf{r}')} & \frac{\delta^2\Omega}{\delta\Delta(\mathbf{r})\delta\mathbf{A}(\mathbf{r}')} \\ \frac{\delta^2\Omega}{\delta\mathbf{A}(\mathbf{r})\delta\Delta(\mathbf{r}')} & \frac{\delta^2\Omega}{\delta\mathbf{A}(\mathbf{r})\delta\mathbf{A}(\mathbf{r}')} \end{pmatrix}, \tag{5.22}$$

has all positive eigenvalues in order to guarantee that $\delta^2\Omega$ is positive definite.

To find the eigenvalues of $L(\mathbf{r},\mathbf{r}')$, we express $\delta\Delta$ and $\delta\mathbf{A}$ as a linear combination of basis functions.

$$\delta\Delta(\mathbf{r}) = \sum_i \delta\hat{\Delta}_i \psi_i(\mathbf{r}) \tag{5.23}$$

$$\delta\mathbf{A}(\mathbf{r}) = \sum_i \delta\hat{\mathbf{A}}_i \psi_i(\mathbf{r}). \tag{5.24}$$

In particular, if we denote the fluctuation parameters as $\theta = (\delta\hat{\Delta}, \delta\hat{\mathbf{A}})$, then to quadratic order, we have

$$\Omega = \Omega_0 + \sum_{ij} H_{ij}\theta_i\theta_j, \tag{5.25}$$

where we have introduced the Hessian matrix $H_{ij} = \frac{\partial^2\Omega}{\partial\theta_i\partial\theta_j}$. If H_{ij} has all positive eigenvalues then the state is stable to the space spanned by $\{\psi_i(\mathbf{r})\}$.

To understand the relationship between H_{ij} and $L(\mathbf{r}, \mathbf{r}')$, consider the eigenvalue equationn

$$\int d\mathbf{r}' L(\mathbf{r}, \mathbf{r}') \begin{pmatrix} \delta\Delta(\mathbf{r}') \\ \delta\mathbf{A}(\mathbf{r}') \end{pmatrix} = E \begin{pmatrix} \delta\Delta(\mathbf{r}) \\ \delta\mathbf{A}(\mathbf{r}) \end{pmatrix}, \tag{5.26}$$

with eigenvalue, E. Multiplying both sides by $\psi_i(\mathbf{r})$ and integrating over $\mathbf{r}$, we find

$$\sum_j H_{ij}v_j = E\sum_j g_{ij}v_j, \tag{5.27}$$

where

$$g_{ij} = \int d\mathbf{r} \frac{\partial\delta\Delta}{\partial\theta_i}\frac{\partial\delta\Delta}{\partial\theta_j} + \frac{\partial\delta\mathbf{A}}{\partial\theta_i}\cdot\frac{\partial\delta\mathbf{A}}{\partial\theta_j} \tag{5.28}$$

so that the vector $v^T = \left(\delta\hat{\Delta}, \delta\hat{\mathbf{A}}\right)$ is a generalized eigenvector of the Hessian and g matrices. In deriving Eq. (5.27) we have made use of the fact that

$$\frac{\partial^2\Omega}{\partial\delta\hat{\Delta}_i\partial\delta\hat{\Delta}_j} = \int d\mathbf{r}d\mathbf{r}'\psi_i(\mathbf{r})\frac{\delta^2\Omega}{\delta\Delta(\mathbf{r})\delta\Delta(\mathbf{r}')}\psi_j(\mathbf{r}'), \tag{5.29}$$

and the analgogous relations for $\partial^2\Omega/\partial\delta\hat{\Delta}_i\partial\delta\hat{\mathbf{A}}_j$ and $\partial^2\Omega/\partial\delta\hat{\mathbf{A}}_i\partial\delta\hat{\mathbf{A}}_j$.

For small values of the κ, the Meissner state first becomes unstable to perturbations that preserve the translational invariance parallel to the superconducting interface. In this regime we need only consider perturbations that depend on the depth from the surface:

$$\psi_i(\mathbf{r}) = \phi_i(z). \tag{5.30}$$

For type-II superconductors the superconducting Meissner state is unstable to two-dimensional perturbations, we choose as basis functions for the perturbations of the form

$$\psi_{2i}(\mathbf{r}) = \phi_i(z)\cos(ky) \tag{5.31}$$

and

$$\psi_{2i+1}(\mathbf{r}) = \phi_i(z)\sin(ky) \tag{5.32}$$

As in the Ginzburg-Landau case, the Fourier modes decouple, so one only need solve the family of generalized eigenvalue problems as the wave-number k is varied. Unlike the Ginzburg-Landau case where it was only necessary to consider perturbations of the form $\delta\Delta(\mathbf{r}) = \delta\tilde{\Delta}(z)\cos(ky)$ and $\delta\mathbf{A}(\mathbf{r}) = (0, \delta\tilde{A}_y(z)\cos(ky), \delta\tilde{A}_z(z)\sin(ky))$, here it is necessary to include the additional perturbations $\delta\Delta(\mathbf{r}) = \delta\tilde{\Delta}_c(z)\cos(ky) + \delta\tilde{\Delta}_s(z)\sin(ky)$ and similarly for $\delta\mathbf{A}$.

The Hessian matrix H_{ij} can be found by a straightforward, term-by-term differentiation of Eq. (5.4). It is necessary to calculate the sensitivities of the Green's functions, δf, $\delta\bar{f}$, and δg to fluctuations in the order parameter and vector potentials. These are found as solutions of the differential equations

$$\begin{gathered}
[\omega_n + \mathbf{n}\cdot(\nabla - i\mathbf{A})]\delta f - \Delta\delta g = if\mathbf{n}\cdot\delta\mathbf{A} + g\delta\Delta \\
[\omega_n - \mathbf{n}\cdot(\nabla + i\mathbf{A})]\delta\bar{f} - \Delta^\dagger\delta g = i\bar{f}\mathbf{n}\cdot\delta\mathbf{A} + g\delta\Delta^\dagger \\
2\mathbf{n}\cdot\nabla\delta g - \Delta^\dagger\delta f + \Delta\delta\bar{f} = f\delta\Delta^\dagger - \bar{f}\delta\Delta,
\end{gathered} \tag{5.33}$$

which can be solved in an manner analogous to the Eilenberger equations by expanding the perturbed Greens functions δf in terms of the perturbed basis $\psi(\mathbf{r})$. The resulting matrix equations are not given here, as they are rather long, but they can be derived in a straightforward way using a computer algebra system.

With the self-consistent solution for the order parameter and vector potential, together with the sensitivities of the Greens functions to perturbations of Δ and $\mathbf{A}$, we have all the elements necessary to calculate H_{ij} and solve the generalized eigenvalue problem. In order to find the super-heating field we proceed in the same was as in[92], in which we vary both the applied magnetic field and the Fourier mode until the smallest eigenvalue first becomes negative. The magnetic field for which this occurs is the superheating field H_{sh} and wavenumber of the Fourier mode is known as the critical momentum k_c.

5.5 Results

In this section we present our numerical estimates of H_{sh} and k_c as functions of κ_0 and temperature, T. As we will see, these results are not converged and require further work to give accurate results.

Within Ginzburg-Landau theory it is customary to present results for H_{sh} and k_c in units of the thermodynamic critical field and the penetration depth λ. This is potentially confusing since H_c and λ are themselves temperature dependent. We present our results in these units, in spite of the possible confusions in order to compare our results to the Ginzburg-Landau predictions. We first describe how H_c and λ depend upon T and κ_0 in order to alleviate this confusion. We will also present how the parameter κ_0 relates to the Ginzburg-Landau parmaeter κ_{GL}.

To make the connection with Ginzburg-Landau theory, we use a phenomenological definition of λ given by Tinkham[90] in terms of the linear response of the of the system to a small applied field:

$$\lambda = \frac{\int_{-\infty}^{0} dz H(z)}{H(0)} = -\frac{A(0)}{A_y'(0)}, \tag{5.34}$$

where $H(z) = -A_y'(z)$ is the magnitude of the magnetic field penetrating inside of the superconductor and the second equality is true for our choice of gauge. Near T_c, Ginzburg-Landau theory predicts that $\lambda \propto \frac{\kappa_0}{\sqrt{2(1-t)}}$. At low temperatures we

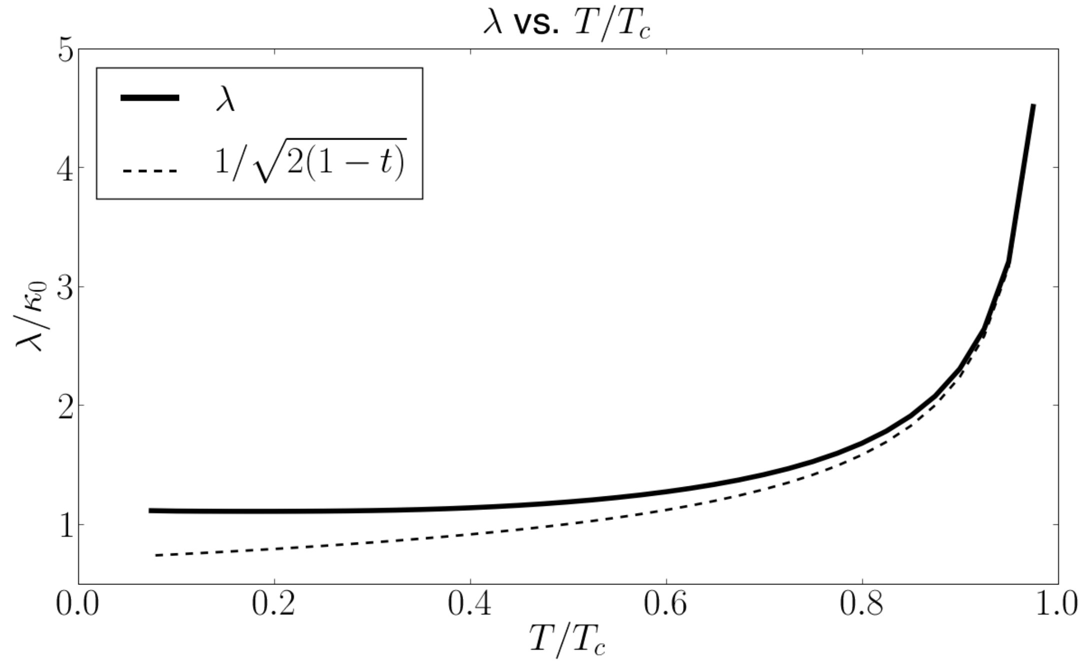


Figure 5.2: λ **vs.** T. The phenomenological penetration depth given by Eq. (5.34) diverges near T_c and approaches κ_0 at low temperature. In our units, λ scales approximately as κ_0.

expect $\lambda \approx \kappa_0$ since we have chosen the BCS coherence length as the unit of length. This behavior is summarized in the numerical calculation in Fig. 5.2. The curve in Fig. 5.2 is nearly independent of κ_0, so that $\lambda \propto \kappa_0$.

The thermodynamic critical field is defined by equating the energy density of the magnetic field with that of the difference between the superconducting and normal states at zero field[90]. Inspecting our expression for the thermodynamic potential, Eq. (5.4), we see that the H_c is given by:

$$H_c^2 \kappa_0^2 = 6\pi T \sum_n \left(\frac{\Delta_0^2 + 2\omega_n^2}{\sqrt{\omega_n^2 + \Delta_0^2}} - 2\omega_n \right), \tag{5.35}$$

where $\Delta_0(T)$ is the zero-field value of the order parameter described in Figure 5.6. This expression is usually approximated by the parabola $H_c(t) \approx H_c(0)(1 - t^2)$, where we find numerically that $H_c(0)\kappa_0 = 1.225$, as can be seen in Figure 5.3.

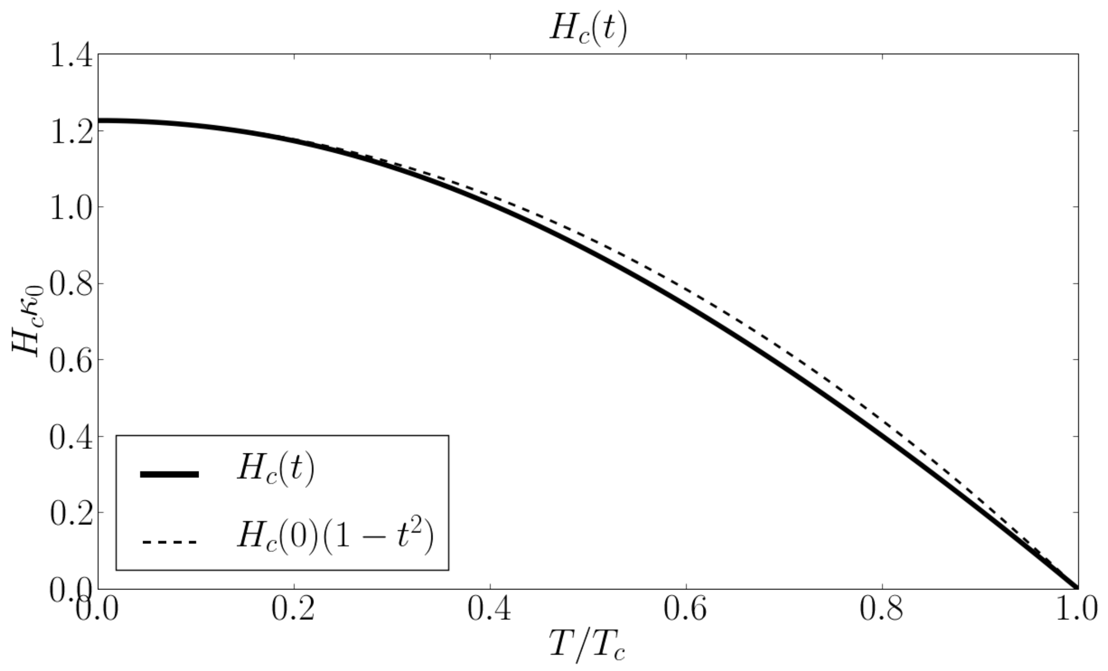


Figure 5.3: H_c **vs.** T. The thermodynamic critical field H_c as a function of $t = T/T_c$.

Note that H_c is proprtional to $1/\kappa_0$.

The other relevant length scale of the problem is the order parameter coherence length, ξ. Near T_c, Ginzburg-Landau theory relates ξ to λ and H_c by

$$\xi = \Phi_0/(2\sqrt{2}\pi\lambda H_c), \tag{5.36}$$

where $\Phi_0 = hc/2e$ is the fluxoid quantum. Away from T_c, there is no unambiguous definition of ξ. If we use Eq. (5.36) as a definition of ξ at lower temperatures as well, then the temperature dependence of ξ is summarized in Figure 5.4. Since $\lambda \propto \kappa_0$ and $H_c \propto 1/\kappa_0$, ξ has a very weak dependence on κ_0 and can be considered a function of temperature only. Notice that at low temperature ξ is not equivalent to the BCS coherence length ξ_0 ($\xi_0 = 1$ in our units).

Defining the phenomenological Ginzburg-Landau parameter $\kappa(T) = \lambda/\xi$, we

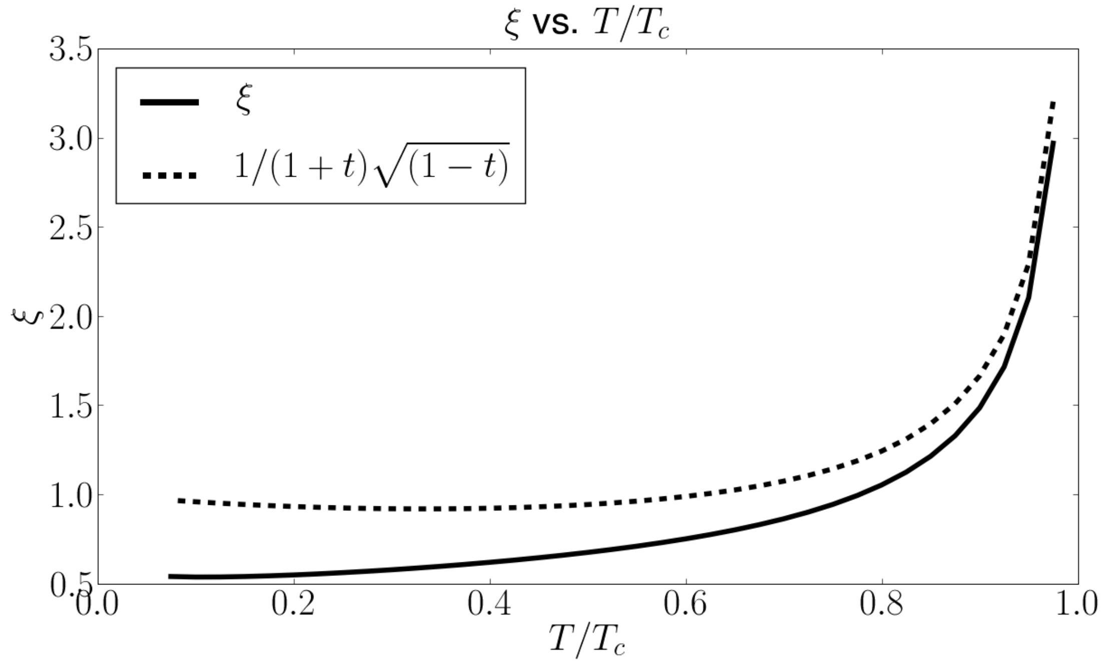


Figure 5.4: ξ **vs.** T/T_c. The phenomenological coherence length xi as a function of $t = T/T_c$. Notice that at $T = 0$, ξ is not equivalent to the BCS coherence length $\xi_0 = 1$.

find in Figure 5.5 that at T_c, $\kappa_{GL} \approx 1.5\kappa_0$ and increases significantly at lower temperatures. It is common to give the Ginzburg-Landau parameter as a material specific property measured at T_c. Using the relation $\kappa_{GL} = \kappa(T_c) \approx 1.5\kappa_0$, we are thus able to connect the predictions of the Eilenberger theory to specific materials. In particular, niobium, the most common metal used in constructing superconducting RF cavities has a Ginzburg-Landau parameter of about 1, corresponding to $\kappa_0 \approx 2/3$. In the subsequent sections, we express our results in terms of the more common κ_{GL} instead of κ_0.

Finally, for completeness we give the temperature dependence of the zero-field order parameter $\Delta_0(T)$ in Figure 5.6 which satisfies

$$\log\left(T/T_c\right) + 2\pi T \sum_n \left(\frac{1}{\omega_n} - \frac{}{\sqrt{\omega_n^2 + \Delta_0^2}} \right). \tag{5.37}$$

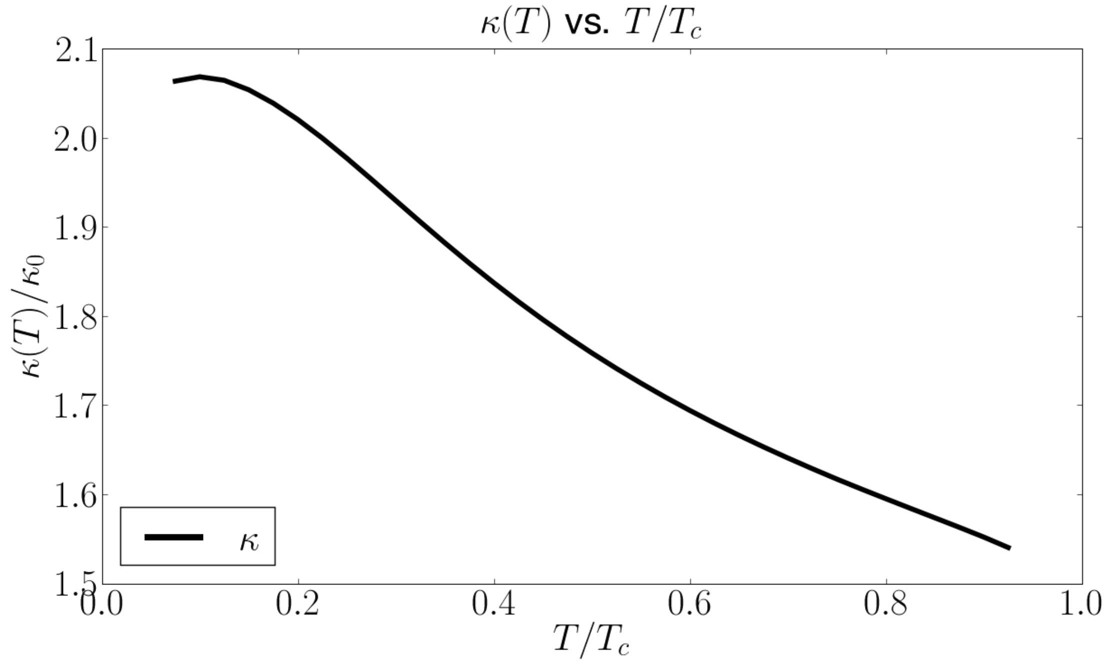


Figure 5.5: $\kappa(T)$ **vs.** T. The phenomenological Ginzburg-Landau parameter $\kappa(T) = \lambda/\xi$ as a function of $t = T/T_c$. At $T = T_c$ we have $\kappa_{GL} \approx 1.5\kappa_0$ and has a relatively weak temperature dependence, increasing slightly at lower tmperatures

We have chosen the energy unit to be the zero-field order parameter value at $T = 0$. Near T_c the order parameter falls quickly to zero.

5.5.1 Small κ

Numerical calculation of H_{sh} is perhaps most straightforward in the regime of $\kappa \sim 1$, where there is not a strong separation of length scales. This is also the most experimentally relevant regime to consider, as it contains niobium (Nb), $\kappa_{GL} = 1.0$, the most commonly used material in superconducting RF cavities.

Our results in this regime are summarized in figure 5.7 - 5.8. At temperatures near T_c we observe that our results appear to converge nicely to the Ginzburg-

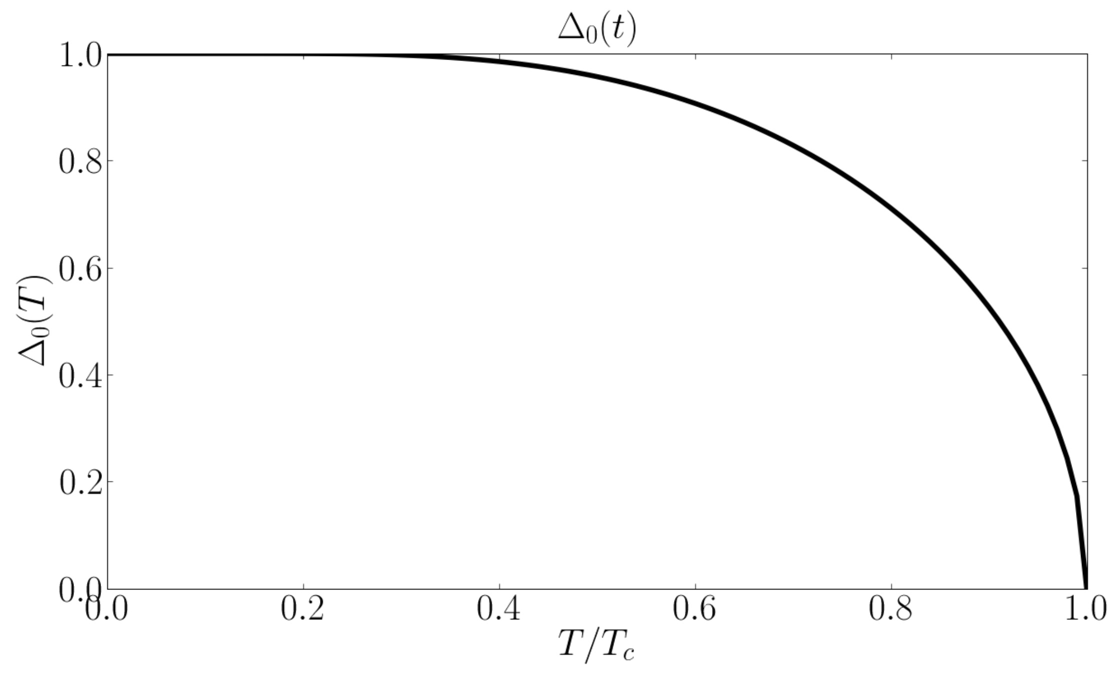


Figure 5.6: $\Delta_0(T)$ **vs.** T/T_c. The zero-field order parameter $\Delta_0(T)$ versus T. The order parameter vanishes at T_c, while rising quickly to 1.

Landau predictions. At lower temperatures, we observe a dramatic increase in the superheating field. In particular, consider the case of $\kappa_{GL} = 1.0$ (niobium) in Figure 5.9. Niobium has a $T_c \approx 9$ K, while RF cavities are operated at approximately 2 K. For this temperature, our solution of the Eilenberger equations predicts a nearly 80% increase in the superheating field compared to the Ginzburg-Landau prediction. If such an increase could realized in an actual RF cavity, the potential performance gains would be dramatic. In section 5.5.3 we discuss convergence issues at low temperature that suggest these results have not converged.

For weakly type-II materials, the metastability first becomes unstable to a uniform penetration of magnetic flux. Within Ginzburg-Landau theory, this is true up to a critical value of $\kappa_c \approx 1.15$. In Figure 5.10 we calculate the temperature dependence of this critical $\kappa_c(T)$. We find that it grows nearly linearly with temperature

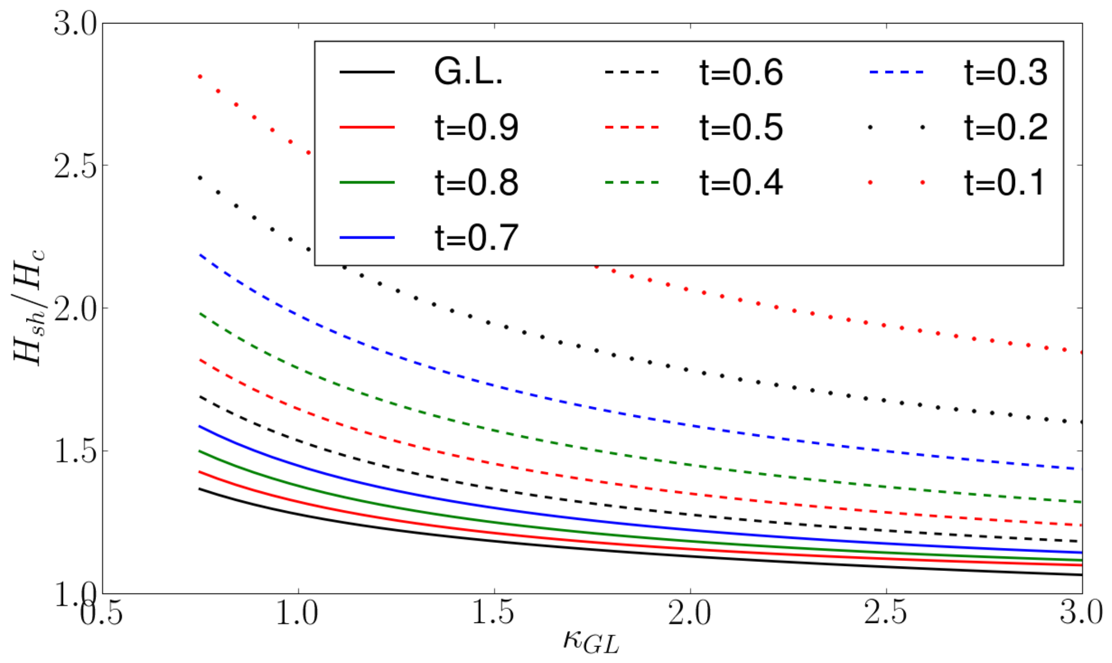


Figure 5.7: **Unconverged H_{sh}/H_c for moderate κ.** The superheating field at several temperatures versus κ_{GL} in units of the thermodynamic critical field. At low temperatures, we predict a dramatic increase in H_{sh} as compared to the corresponding results from Ginzburg-Landau theory. (near T_c).

up to approximately $\kappa_c(T=0) \approx 1.40$ at very low temperatures.

5.5.2 Large κ

In this section we present our results for the extreme type-II case (large κ). This regime includes materials that are being considered as possible replacements for Niobium in RF cavities, such as Niobium-Tin (Nb_3Sn) with $\kappa_{GL} \approx 20$.

Our results are summarized in figure 5.11. As in the weakly type-II case, we observe a dramatic increase in H_{sh} at low temperatures as compared to the Ginzburg-Landau predictions. In Figure 5.12 we plot the critical wavenumber to which the superconducting state first becomes unstable, which does not vary as

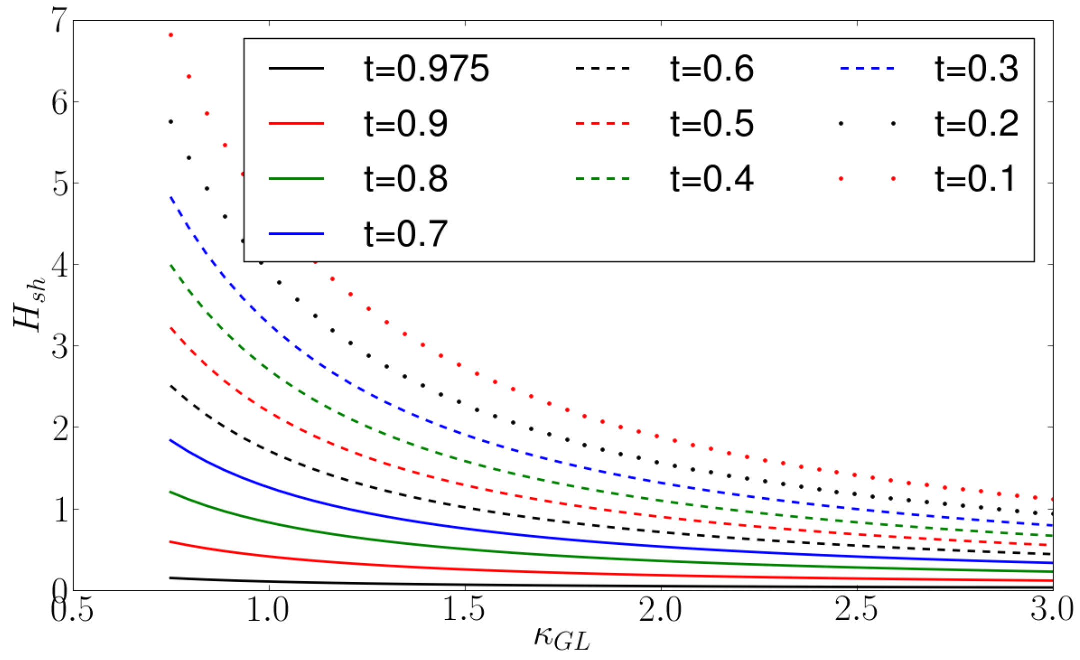


Figure 5.8: **Unconverged H_{sh} for moderate κ.** The superheating field at several temperatures versus κ_{GL}.

dramatically with temperature as H_{sh}. In each case, we note that our results appear to converge to the Ginzburg-Landau prediction near T_c.

5.5.3 Convergence

One of the challenges of numerically calculating H_{sh} within Eilenberger theory is the lack of known results with which to compare. The superheating field has been studied extensively in Ginzburg-Landau theory, corresponding to the $T \to T_c$ limit of Eilenberger theory. Indeed, we have seen in section 5.5.1 and 5.5.2 that our results appear to converge nicely to the Ginzburg-Landau results near T_c.

The other limit that has been studied recently is that of infinite-κ in reference [23], where it was found that H_{sh}/H_c experienced a 13% increase over the Ginzburg-

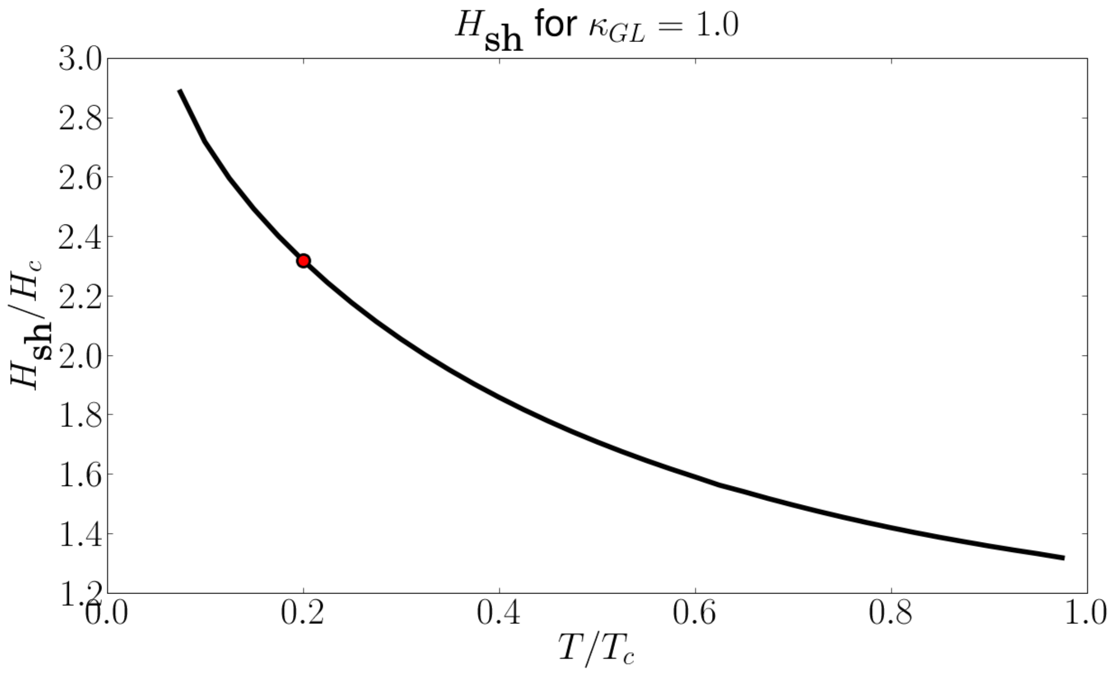


Figure 5.9: **Unconverged $H_{\rm sh}$ of niobium vs. T.** The temperature dependence of the superheating field of niobium, corresponding to $\kappa_{GL} = 1.0$, the most commonly used material in superconducting RF cavities. With a critical temperature of about 9 K, these cavities are operated at $T \approx 2$ K. At this temperature the numerical results from Eilenberger theory predict $H_{\rm sh}/H_c = 2.3$ (red dot), over an 80% increase over the Ginzburg-Landau prediction $H_{\rm sh}/H_c = 1.28$.

Landau prediction (from $H_{\rm sh}/H_c = 0.745$ at T_c up to $H_{\rm sh}/H_c = 0.845$ at $T/T_c = 0.06$). By extrapolating the results at large, but finite κ in Figure 5.11 to $\kappa = \infty$, our results clearly overestimate the results in reference[23]. While it is tempting to blame this discrepancy on a lack of convergence due to the two-well separated length scale λ and ξ, it is curious that our results appear to converge to the known Ginzburg-Landau results, even at large κ. This suggests that our convergence problem must be specific to low temperatures.

Although the most likely cause of poor convergence at low temperatures is the number of Matsubara frequencies included, we can check that including more frequencies in the summations does not significantly alter the results. Since we

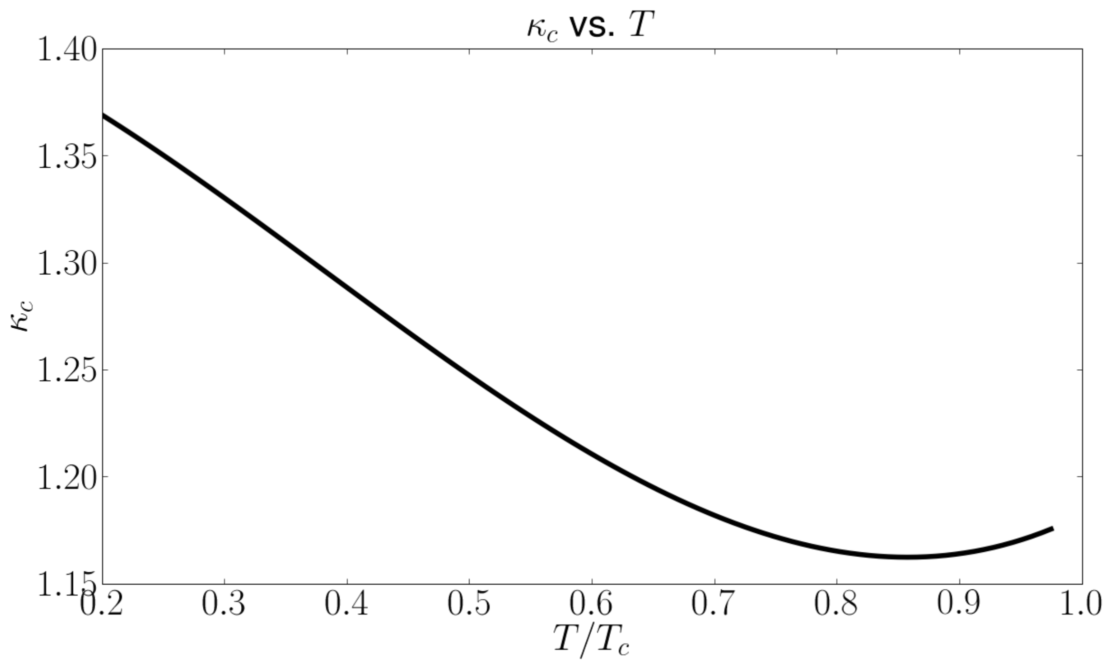


Figure 5.10: **Unconverged κ_c vs. T.** The critical κ_c at which two dimensional fluctuations first become relevant for the penetration of magnetic flux. The Ginzburg-Landau theory predicts $\kappa_c \approx 1.15$, and our numerics recover this result within a few percent. At lower temperatures κ_c grow linearly with temperature up to approximately 1.40.

only explicitely solve the Eilenberger equations for a few Matsubara frequencies and interpolate the remaining results, it is not difficult to effectively use thousands of Matsubara frequencies. Numerical exploration suggests that our results have converged in Matsubara frequencies.

For type-II superconductors at very low temperatures, it is known that the Eilenberger equations predict the characteristic length scale of a vortex core to be $\xi_1 \sim \xi_0 \; T/T_c$, where ξ_0 is the BCS coherence length[62]. It is possible, therefore, that our low temperature calculations are limited by our basis functions not properly probing this length scale. Indeed, our basis functions were originally chosen to probe length scales comparable to the coherence length given by Eq. (5.36), which

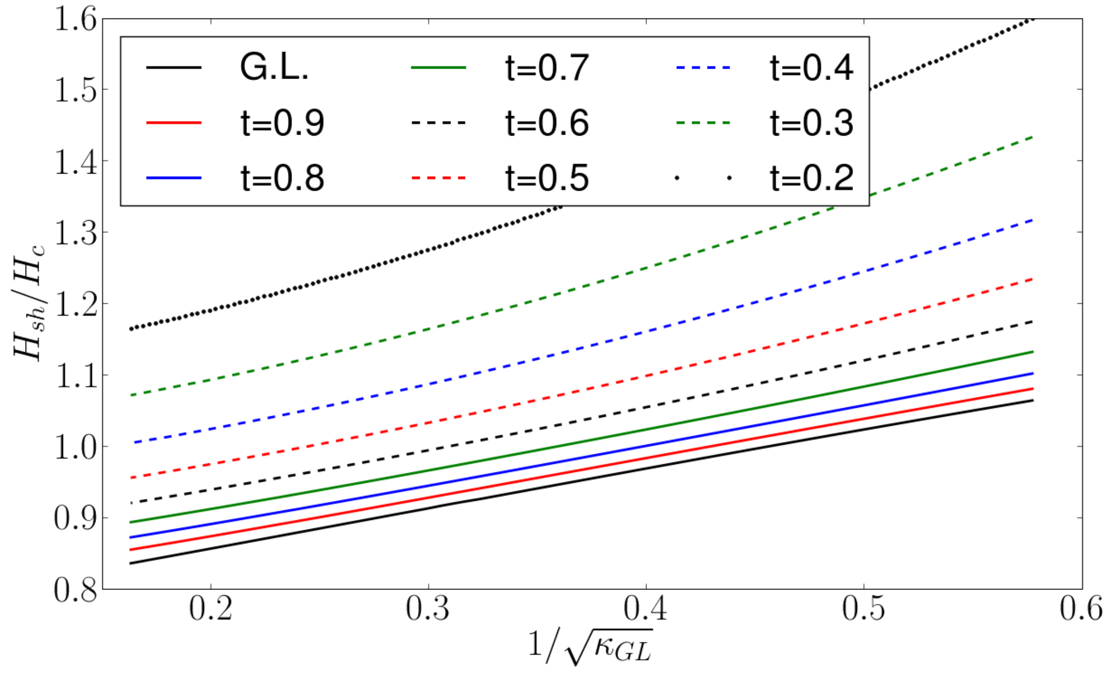


Figure 5.11: **Unconverged H_{sh}/H_c for large κ.** The superheating field at several temperatures versus κ_{GL} in units of the thermodynamic critical field. Near T_c our results appear to converge to the Ginzburg-Landau prediction. However, extrapolating our results to infinite κ clearly predicts a larger H_{sh} that that calculated in reference [23].

is comparable to the BCS coherence length at low T. However, by numerical experiments that vary both the number and spacing of basis functions, we find our results have only a weak dependence on these details.

Finally, we explore the convergence properties of our integrals over the Fermi surface. At temperatures near T_c, we find that the results have already converged with only very few points ($m = 2$ as described in section 5.4). However, at lower temperatures the results seem to depend very strongly on the number of Fermi surface points used. The results presented in sections 5.5.1 and 5.5.2 correspond to $m = 2$, where it was predicted that fields could be increased by up to 80% larger than previously thought for niobium at operating temperature. However,

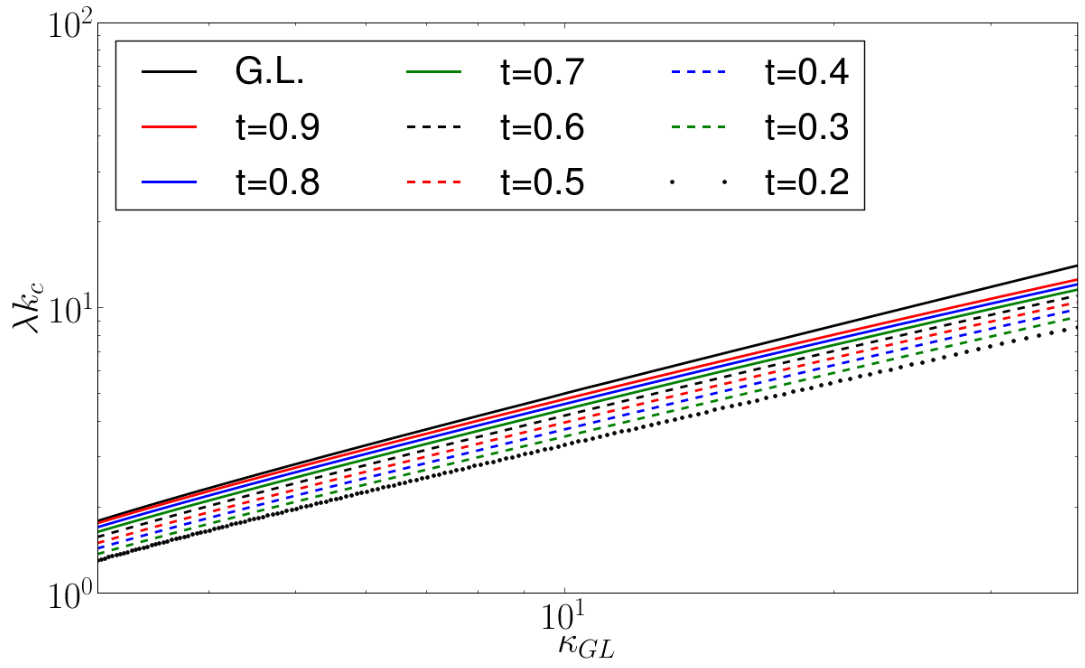


Figure 5.12: **Unconverged λk_c for large κ.** Unlike the superheating field, the critical momentum k_c at to which the Meissner state is first unstable does not vary much with temperature. We see here, that at all temperatures, k_c scales roughly the same as for the Ginzburg-Landau case, although it is smaller at low temperatures, consistent with the observation in Fig. 5.10 that two-dimensional fluctuations first become important for larger κ at low temperature.

increasing the number of Fermi surface points from $m = 2$ up to $m = 4$, 8, and 16 (corresponding to 8, 32, 128, and 512 points respectively), changes the prediction for H_{sh}/H_c from 2.45 to 1.10, 1.50, and 1.43 respectively. Although the final prediction is much more in line with the 13% increase observed at infinite κ, these results have clearly not yet converged. Furthermore, the non-monotonic behavior of H_{sh} also makes it difficult to extrapolate the results to larger m.

It may be surprising that the convergence properties of the Fermi-surface integrals depend so strongly on temperature. By inspecting the Eilenberger equations (Eqs. (5.1)) we can see that for larger temperatures the dependence of the Greens functions f and $\bar{f}$ will be washed out by the larger ω_n, even for $n = 0$. However, at low temperatures, as ω_0 becomes smaller, f and $\bar{f}$ will depend much more strongly on their orientation on the Fermi surface. Indeed, we observe that the values of f and $\bar{f}$ at the superconducting interface ($z = 0$) seem to develop a cusp at the equator $\mathbf{n} \cdot \hat{\mathbf{z}} = 0$. We speculate that by choosing an alternative spacing than that described in section 5.4 and reference [4] which places a higher density near the equator, we will be able to achieve better convergence at low temperatures without requiring so many evaluations of the Eilenberger equations. Indeed the method described in section 5.4 chooses points such that their density is lowest near the equator.

5.6 Comparison with experiments

A quantitatively accurate description of H_{sh} is of immediate practical concern in the construction of superconducting RF cavities used in particle accelerators, where the maximum accelerating field is limited by the penetration of flux into the bulk superconducting material. Experiments within the accelerator community have recently been conducted to measure the temperature dependence H_{sh} [98].

In order to make a connection between the theoretical predictions of this work and the materials used in cavity construction it is necessary to experimentally measure the sample's critical temperature, T_c, thermodynamic critical field H_c and Ginzburg-Landau parameter $\kappa_{GL} \approx 1.5\kappa_0$. The experimentally measured superheating field for a cavity constructed of niobium is presented in figure 5.13. This particular sample had a critical temperature of 8.83 K and a Ginzburg-Landau parameter of $\kappa_{GL} = 3.5$. (Note the large Ginzburg-Landau parameter is due to operating in the dirty limit.) We inferred the thermodynamic critical field by

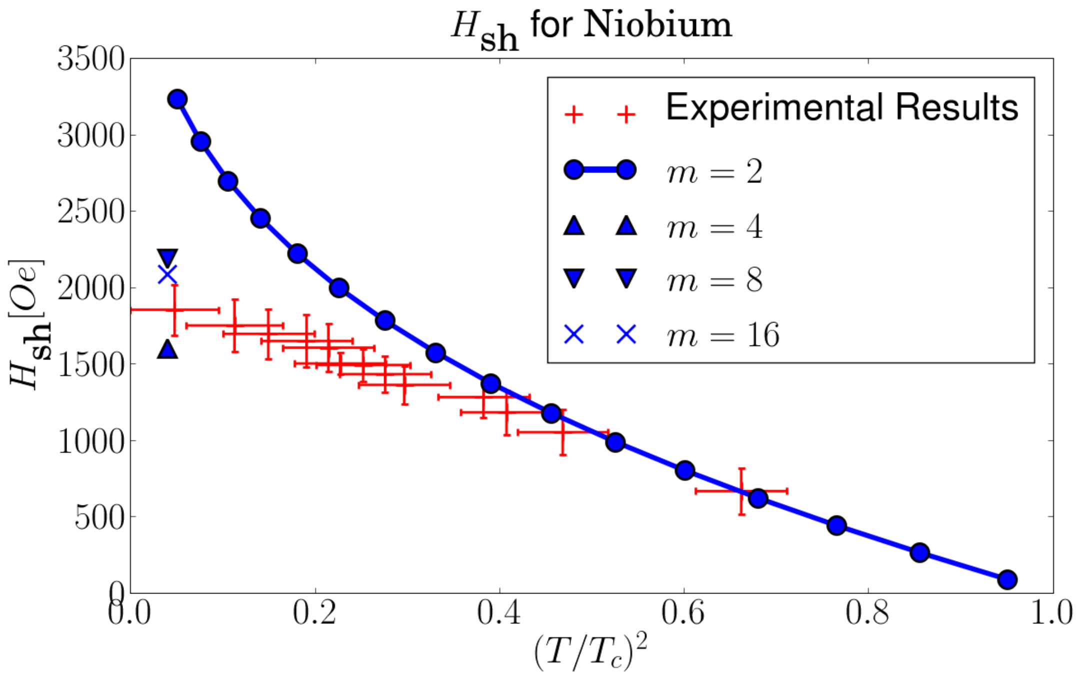


Figure 5.13: **Experimentally measured superheating field**. The experimentally measured superheating field for a niobium resonance cavity, together with the Eilenberger prediction at all temperatures evaluated for 8 Fermi surface points ($m = 2$). Since the theoretical prediction is in units of H_c, we infer H_c for this sample by fitting to the first data point. By including additional Fermi surface points ($m = 4$, $m = 8$, and $m = 16$) the predictions approach the bounds of the experimental error bars although the results have not yet converged. The prediction of Ginzburg-Landau theory would be very nearly a straight line, consistent with the experimental observations.

matching H_{sh} near T_c to the Ginzburg-Landau prediction. Observe that the preliminary results described in section 5.5 predict theoretical limit much larger than that experimentally observed. By increasing the the number of points on the Fermi-surface the predictions approach the bounds of the experimental error bars.

5.7 Conclusions

In this paper we have numerically explored the metastability of the superconducting Meissner state within Eilenberger theory. These results extend previous investigations of this phenomenon within Ginzburg-Landau theory[31, 42, 60, 61, 37, 26, 24, 33], which is quantitatively valid only near the critical temperature, and the infinite-κ London limit[23]. Although preliminary numerical results predicted that H_{sh}/H_c can be much larger at low temperatures than near T_c, it is clear that these low temperature results are not fully converged. Further investigation is necessary to resolve these convergence difficulties.

In addition to the calculation presented here, further refinements are possible within the Eilenberger theory. In particular, we assumed a uniform density of states on the Fermi surface; however, in principle more realistic information could be used for either niobium or another material of interest. Additionally, the methods used to produce of RF cavities introduce impurities which place the superconductor in the dirty limit. Eilenberger theory can also account for impurity scattering by including additional terms in Eqs. (5.1), although including this information directly would be difficult as it would require solving for all of the Greens functions simultaneously. In practice, the impurities have the effect of increasing the Ginzburg-Landau parameter and washing out the details of the Fermi-surface. It is hopeful, therefore, that by calculating H_{sh} for the larger, experimentally measured κ_{GL} while assuming a spherical Fermi-surface we can reasonably approximate the actual superheating field attainable in an RF cavitiy.

Finally, we remind the reader that the results presented in section 5.5 should not be taken too seriously due to the convergence problems discussed in section 5.5.3. Additional computer time is necessary resolve these difficulties.

Acknowledgements

The authors would like to thank Hasan Padamsee, Georg Hoffstaetter, Matthias Liepe, Nick Valles, and Vinay Ambegaokar for helpful discussion. We would also like to thank Matthias Liepe and Nick Valles for sharing their experimental data with us. This work was supported by NSF grant number DMR-1005479 (MKT & JPS), by the Department of Energy under contract de-sc0002329 (MKT), and by Yale University (GC).

APPENDIX A

APPENDICES TO CHAPTER 2

A.1 Information Geometry

The Fisher information matrix, or simply Fisher information, I, is a measure of the information contained in a probability distribution, p. Let ξ be the random variable whose distribution is described by p, and further assume that p depends on other parameters θ that are not random. This leads us to write

$$p = p(\xi; \theta),$$

with the log likelihood function denoted by l:

$$l = \log p.$$

The information matrix is defined to be the expectation value of the second derivatives of l,

$$I_{\mu\nu} = \langle -\frac{\partial^2 l}{\partial\theta^\mu \partial\theta^\nu}\rangle = -\int d\xi \; p(\xi, \theta)\frac{\partial^2 l}{\partial\theta^\mu \partial\theta^\nu}. \tag{A.1}$$

It can be shown that the Fisher information can be written entirely in terms of first derivatives:

$$I_{\mu\nu} = \langle \frac{\partial l}{\partial\theta^\mu}\frac{\partial l}{\partial\theta^\nu}\rangle = \int d\xi \; p(\xi, \theta)\frac{\partial l}{\partial\theta^\mu}\frac{\partial l}{\partial\theta^\nu}. \tag{A.2}$$

Eq. (A.2) makes it clear that the Fisher information is a symmetric, positive definite matrix which transforms like a covariant rank-2 tensor. This means that it has all the properties of a metric in differential geometry. Information geometry considers the manifolds whose metric is the Fisher information matrix corresponding to various probability distributions. Under such an interpretation, the Fisher information matrix is known as the Fisher information metric.

As we saw in Section 2.2, least squares problems arise by assuming a Gaussian distribution for the deviations from the model. Under this assumption, the cost function is the negative of the log likelihood (ignoring an irrelevant constant). Using these facts, it is straightforward to apply Eq. (A.1) or Eq. (A.2) to calculate the information metric for least squares problems. From Eq. (A.1), we get

$$g_{\mu\nu} = \langle \frac{\partial^2 C}{\partial \theta^\mu \partial \theta^\nu} \rangle = \sum_m \langle \partial_\mu r_m \partial_\nu r_m + r_m \partial_\mu \partial_\nu r_m \rangle, \tag{A.3}$$

where we have replaced I by g to indicate that we are now interpreting it as a metric.

Eq. (A.3), being an expectation value, is really an integral over the random variable (i.e. the residuals) weighted by the probability. However, since the integral is Gaussian, it can be evaluated easily using Wick's theorem (remembering that the residuals have unit variance). The only subtlety is how to handle the derivatives of the residuals. Inspecting Eq. (2.1), reveals that the derivatives of the residuals have no random element, and can therefore be treated as constant. The net result is

$$g_{\mu\nu} = \sum_m \partial_\mu r_m \partial_\nu r_m = (J^T J)_{\mu\nu}, \tag{A.4}$$

since $\langle r_m \rangle = 0$. Note that we have used the Jacobian matrix, $J_{m\mu} = \partial_\mu r_m$ in the final expression.

We arrive at the same result using Eq. (A.2) albeit using different properties of the distribution:

$$g_{\mu\nu} = \sum_{m,n} \langle r_m \partial_\mu r_m r_n \partial_\mu r_n \rangle.$$

Now we note that the residuals are independently distributed, $\langle r_m r_n \rangle = \delta_{mn}$, which immediately gives Eq. (A.4), the same metric found in Section 2.2.

There is a class of connections consistent with the Fisher metric, known as the

α-connections because they are parametrized by a real number, α [3]. They are given by the formula

$$\Gamma^{(\alpha)}_{\mu\nu,\epsilon} = \langle \partial_\epsilon l \partial_\mu \partial_\nu l + \left(\frac{1-\alpha}{2}\right) \partial_\epsilon l \partial_\mu l \partial_\nu l \rangle.$$

This expression is straightforward to evaluate. The result is independent of α,

$$\Gamma^{\epsilon}_{\mu\nu} = g^{\epsilon\kappa} \sum_m \partial_\kappa r_m \partial_\mu \partial_\nu r_m.$$

It has been shown elsewhere that the connection corresponding to $\alpha = 0$ is in fact the Riemann connection. It is interesting to note that all the α-connections, for the case of the nonlinear least squares problem, are the Riemann connection.

These results are of course valid only for a cost function that is a sum of squares. For example, one might wish to minimize

$$C = \sum_m |r_m|^p, \tag{A.5}$$

which is naturally interpreted as the p^{th} power of the L^p norm in data space. The case of $p = 1$ is used in "robust estimation", while "minimax" fits correspond to the case of $p = \infty$ [83]. Note that under a general L^p norm, data space does not have a metric tensor as it has no natural inner product consistent with the norm.

Consider a cost function that is a differentiable function of the residuals, but is otherwise arbitrary. In this case, the metric becomes

$$g_{\mu\nu} = \langle \partial_\mu C \partial_\nu C \rangle,$$

where

$$\partial_\mu C = J_{m\mu} \frac{\partial C}{\partial r_m}.$$

As we argue above, the Jacobian matrix has no stochastic element and may be factored from the expectation value, giving

$$g_{\mu\nu} = J_{m\mu} G_{mn} J_{n\nu},$$

where we have introduced

$$G_{mn} \propto \int d\vec{r}\, e^{-C} \frac{\partial C}{\partial r_m} \frac{\partial C}{\partial r_n}$$

as the metric of the space in which the model manifold is now embedded. The proportionality constant is determined by normalizing the distribution of the residuals. Although the metric of the embedding space is not necessarily the identity matrix, it is constant, which implies that the embedding space is generally flat. In a practical sense, the transition from least squares to an arbitrary cost functions merely requires replacing the metric $J^T J \to J^T G J$; however, the distinction that the embedding space does not have the same norm as data space is important.

For the case of the cost function in Eq. A.5, corresponding to the L^p norm, $G_{mn} \propto \delta_{mn}$, so the metric of model manifold is the same as for least squares, $g = J^T J$. However, unless $p = 2$, the distance between nearby points on the model manifold is proportional to the Euclidean distance not the L^p norm distance natural to data space. For the cases $p = 1$ and $p = \infty$ the cost contours in geodesic coordinates (circular for $p = 2$) become squares. A Newton-like method, such as Levenberg-Marquardt, would no longer take the most direct path to the best fit in geodesic coordinates and would additionally have no sense for how far away the best fit would lie. As a consequence, many of the results of this work are specific to quadratic costs and it is unclear how well the methods would generalize to more arbitrary functions.

The field of information geometry is summarized nicely in several books [3, 75].

1. Initialize values for the parameters, x, the Levenberg-Marquardt parameter λ, as well as λ_{up} and λ_{down} to be used to adjust the damping term. Evaluate the residuals r and the Jacobian J at the initial parameter guess.
2. Calculate the metric, $g = J^T J + \lambda I$ and the cost gradient $\nabla C = J^T r$, $C = \frac{1}{2}r^2$.
3. Evaluate the new residuals, r_{new} at the point given by $x_{new} = x - g^{-1}\nabla C$, and calculate the cost at the new point, $C_{new} = \frac{1}{2}r^2_{new}$.
4. If $C_{new} < C$, accept the step, $x = x_{new}$ and set $r = r_{new}$ and $\lambda = \lambda/\lambda_{down}$. Otherwise, reject the step, keep the old parameter guess x and the old residuals r, and adjust $\lambda = \lambda \times \lambda_{up}$.
5. Check for convergence. If the method has converged, return x as the best fit parameters. If the method has not yet converged but the step was accepted, evaluate the Jacobian J at the new parameter values. Go to step 2.
Traditional Levenberg-Marquardt as described in [64, 69, 83]

Algorithm 1: Traditional Levenberg-Marquardt algorithm

A.2 Algorithms

Since we are optimizing functions with the form of sums of squares, we are primarily interested in algorithms that specialize in this form, specifically variants of the Levenberg-Marquardt algorithm. The standard implementation of the Levenberg-Marquardt algorithm involves a trust region formulation. A FORTRAN implementation, which we use, is provided by MINPACK [73].

The traditional formulation of Levenberg-Marquardt, however, does not employ a trust region, but adjusts the Levenberg-Marquardt term based on whether the cost has increased or decreased after a given step. An implementation of this algorithm is described in Numerical Recipes [83] and summarized in Algorithm 1. Typical values of λ_{up} and λ_{down} are 10. We use this formulation as the basis for our modifications.

The delayed gratification version of Levenberg-Marquardt that we describe in section 2.9.3 modifies the traditional Levenberg-Marquardt algorithm to raise and lower the Levenberg-Marquardt term by differing amounts. The goal is to accept

```
1. Initialize values for the parameters, $x$, the Levenberg-Marquardt parameter $\lambda$, as well as $\lambda_{up}$ and $\lambda_{down}$ to be used to adjust the damping term, and $\alpha$ to control the acceleration/velocity ratio. Evaluate the residuals $r$ and the Jacobian $J$ at the initial parameter guess.
2. Calculate the metric, $g = J^T J + \lambda I$ and the Cost gradient $\nabla C = J^T r$, $C = \frac{1}{2} r^2$.
3. Calculate the velocity $v = -g^{-1} \nabla C$, the geodesic acceleration of the residuals in the direction of the velocity $a = -g^{-1} J^T \left( v^\mu v^\nu \partial_\mu \partial_\nu r \right)$
4. Evaluate the new residuals, $r_{new}$ at the point given by $x_{new} = x + v + \frac{1}{2} a$ , and calculate the cost at the new point, $C_{new} = \frac{1}{2} r_{new}^2$.
5. If $C_{new} < C$ and $|a|/|v| < \alpha$, accept the step, $x = x_{new}$ and set $r = r_{new}$ and $\lambda = \lambda / \lambda_{down}$. Otherwise, reject the step, keep the old parameter guess $x$ and the old residuals $r$, and adjust $\lambda = \lambda \times \lambda_{up}$.
6. Check for convergence. If the method has converged, return $x$ as the best fit parameters. If the method has not yet converged but the step was accepted evaluate the Jacobian $J$ at the new parameter values. Go to step 2.   Geodesic Acceleration in the traditional Levenberg-Marquardt algorithm
```

Algorithm 2: Geodesic acceleration Levenberg-Marquardt algorithm

a step with the smallest value of the damping term that will produce a downhill step. This can typically be accomplished by choosing $\lambda_{up} = 2$ and $\lambda_{down} = 10$.

The geodesic acceleration algorithm can be added to any variant of Levenberg-Marquardt. We explicitly add it to the traditional version and the delayed gratification version, as described in Algorithm 2. We do this by calculating the geodesic acceleration on the model graph at each iteration. If the step raises the cost or if the acceleration is larger than the velocity, then we reduce the Levenberg-Marquardt term and reject the step by default. If the step moves downhill and the velocity is larger than the acceleration, then we accept the step. For accepted steps we raise the Levenberg-Marquardt term; otherwise, we decrease the Levenberg-Marquardt term. In our experience the algorithm described in Algorithm 2 is robust enough for most applications; however, we do not consider it to be a polished algorithm. We will present elsewhere an algorithm utilizing geodesic acceleration that is further optimized and that we will make available as a FORTRAN routine [96].

In addition to the variations of the Levenberg-Marquardt algorithm, we also compare algorithms for minimization of arbitrary functions not necessarily of the least squares form. We take several such algorithms from the Scipy optimization package [57]. These fall into two categories, those that make use of gradient information and those that do not. Algorithms utilizing gradient information include a quasi-Newton of Broyden, Fletcher, Goldfarb, and Shannon (BFGS), described in [79]. We also employ a limited memory variation (L-BFGS-B) described in [21] and a conjugate gradient (CG) method of Polak and Ribiere, also described in [79]. We also explored the downhill simplex algorithm of Nelder and Mead and a modification of Powells' method [57], neither of which make use of gradient information directly, and were not competitive with other algorithms.

APPENDIX B

APPENDIX TO CHAPTER 3

B.1 Test Problems

In order to gauge the relative effectiveness of the improvements descrived in this paper, we use 17 test problems (denoted by the letters A-Q throughout this work) which we take from the Minpack-2 project [5] and the NIST Statistical Reference datasets [70] and some of our own research. We summarize these problems in this appendix.

Problem A: Isomerization of α-pinene (Direct formulation) taken from the Minpack-2 project, consisting of five parametes and 40 residuals. This model is evaluated as a linear ordinary differential equation with unknown coefficients.

Problem B: Isomerization of α-pinene (Collocation formulation) taken from the Minpack-2 project, consisting of 130 parameters and 165 residuals. This is an example of a constrained optimization problem in which the constraint is implemented as an l_2 penalty. In our impelementation, we have used the relatively weak penalty strength of $\sigma = 1000$ (as opposed to $\sigma = 10^6$ as suggested in [5]). As the strength of σ is increased, the algorithm must more closely maintain the constraints at each iteration, making the algorithm become much slower. Anecdotally, we observe that geodesic acceleration and bold acceptance can be very helpful in these cases, although the relatively large computational cost of this problem makes exploring this for all possible algorithms prohibitive.

Problem C: Coating thickness standardization from taken from the Minpack-2 project, consisting of 134 parameters and 252 residuals. This problem is a multiple-response data-fitting problem. Because of its larger size it is one of the more computationally intensive problems in the set.

Problem D: Exponential data fitting taken from the Minpack-2 Project, consisting of 5 parameters and 33 residuals. The functional form of this problem is

$$y(t, \theta) = \theta_1 + \theta_2 e^{-t\theta_4} + \theta_3 + e^{-t\theta_5}. \tag{B.1}$$

This problem similar to those used in references [94, 95] for which geodesic acceleration was shown to be very effective.

Problem E: Gaussian data fitting taken from the Minpack-2 Project, consisting of 11 parameters and 65 residuals. The functional form is

$$y(t, \theta) = \theta_1 e^{-\theta_5} + \theta_2 e^{-(t-\theta_9)^2\theta_6} + \theta_3 e^{-(t-\theta_{10})^2\theta_7} + \theta_4 e^{-(t-\theta_{11})^2\theta_8}. \tag{B.2}$$

This problem is difficult for starting points far from the minimum.

Problem F: Analysis of thermistor resistance taken from the Minpack-2 Project, also known as the MGH10 problem from the NIST dataset. This problem consists of 3 parameters and 16 data points. The functional form of this problem is

$$y(t) = \theta_1 e^{\frac{\theta_2}{t-\theta_3}}. \tag{B.3}$$

For starting points with large values of θ_3, this problem becomes very difficult as the t dependence is lost. Including a small value of α in the geodesic acceleration acceptance criterion is very helpful to force the algorithm move towards smaller θ_3 in this case.

Problem G: Analysis of enzyme reaction taken from the Minpack-2 Project, also known as the MGH09 problem from the NIST dataset. This problem consists of 4 parameters and 11 data points. This problem takes the form

$$y(t,\theta) = \frac{\theta_1\left(t^2 + t\theta_2\right)}{t^2 + t\theta_3 + \theta_4}. \tag{B.4}$$

Many algorithms have a low success rate because of a local minimizer at infinity. As discussed in [95], this scenario is likely to be a generic feature of large data, ill-conditioned data fitting problems.

Problem H: Chebyshev quadrature taken from the Minpack-2 Project, consisting of 8 parameters and 11 residuals. This problem exhibits a disparity between the success rate and the convergence rate due to algorithms converging to local minima.

Problem I: Thurber problem from the NIST dataset, consisting of 7 parameters and 37 residuals. This problem is a rational function of the form

$$y(t,\theta) = \frac{\theta_1 + \theta_2 t + \theta_3 t^2 + \theta_4 t^3}{1 + \theta_5 t + \theta_6 t^2 + \theta_7 t^3} \tag{B.5}$$

Problem J: BoxBOD problem from the NIST dataset, consisting of 2 parameters and 6 residuals. The functional form of the problem is

$$y(t,\theta) = \theta_1\left(1 - e^{-\theta_2 t}\right) \tag{B.6}$$

Problem K: Rat42 problem from the NIST dataset, consisting of 3 parameters and 9 residuals. The functional form of this problem is

$$y(t,\theta) = \frac{\theta_1}{1 + e^{\theta_2 - \theta_3 t}} \tag{B.7}$$

Problem L: Eckerle4 problem from the NIST dataset, consisting of 3 parameters 35 residuals. The functional form of this problem is

$$y(t,\theta) = \frac{\theta_1}{\theta_2} e^{\frac{-(t-\theta_3)^2}{2\theta_2^2 p}} \tag{B.8}$$

Problem M: Rat43 problem from the NIST dataset, consisting of 4 parameters and 15 residuals. The functional form of this problem is

$$y(t, \theta) = \frac{\theta_1}{\left(1 + e^{\theta_2 - \theta_3 t}\right)^{1/\theta_4}} \tag{B.9}$$

Problem N: Bennett5 problem from the NIST dataset, consisting of 3 parameters and 154 residuals. The functional form of this problem is

$$y(t, \theta) = \theta_1 \left(\theta_2 + t\right)^{-1/\theta_3} \tag{B.10}$$

Problem O: A problem from systems biology described in [18]. This model consists of a differential equation model of 48 parameters, mostly reaction rates and Michaelis-Menten constants fit to 68 data points. In order to help keep the parameters bounded, we have also introduced weak priors as described in [95].

Problem P: A problem for fitting a scaling function describing the distribution of avalanche sizes [25]. This model has 32 parameters fit to 398 data points.

Problem Q: A training problem for a feed forward artificial neural network. The network is trained to data describing the compressive strength of concrete, as described in[102] and available here [39]. In our formulation, there are 81 parameters, consisting of the connection weights of the neural networks, and 1030 data points. We also include a weak quadratic prior on the parameters centered at 0 in order to help avoid parameter evaporation. These priors serve the same function as those in Problem O, described in [95].

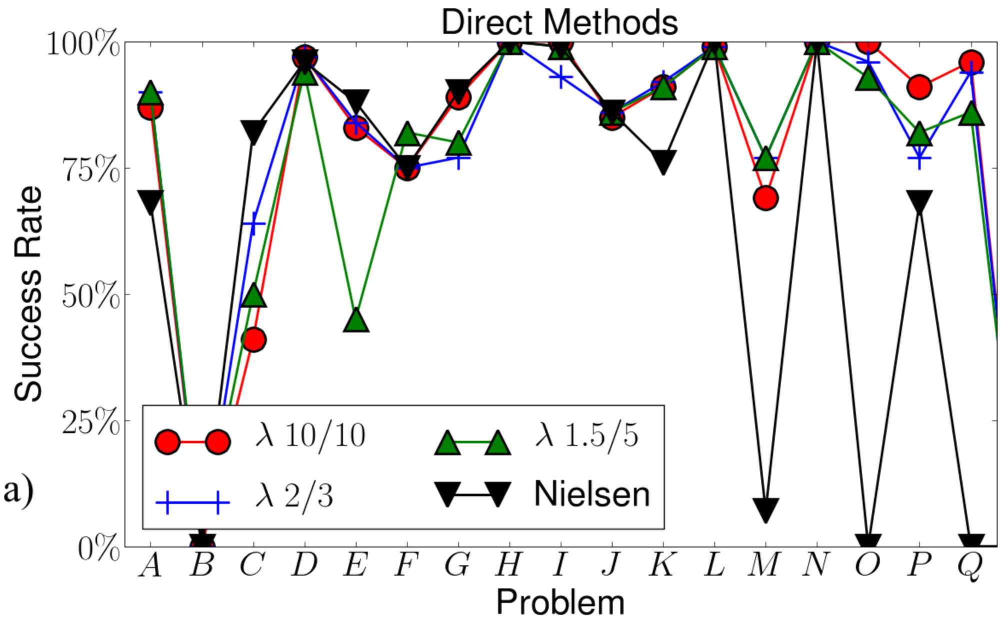
Although there are other methods available for training artificial neural networks, they provide a good test problem for the general least squares. In particular, neural networks have many of the same properties as other fitting problems and provide an easy framework for varying the amount data, the number of inputs and outputs of the function as well as the number of parameters. They can also be evaluated relatively quickly and easily.

For each of these problem, we choose starting points from a Gaussian distribution centered at one of the suggested starting points for each problem. The width of the Gaussian is manually adjusted until one of the standard algorithms begins to show a noticeable variation in performance among the points. This method of choosing the starting points makes many of the test problems much harder than they otherwise would have been. It is fortunate that by choosing starting points in this way the easy problems can be made of comparable difficulty to the more realistic problems O, P, and Q. In particular, the ease and quick evaluation of the smaller problems make them ideal test cases provided they can be made sufficiently difficult to imitate more realistic problems. In addition to making the problems more difficult, by considering the performance from several starting points, we can avoid the complication that an algorithm may perform well by accident.

We give three measures of an algorithm's performance on a given problem. First, we consider the fraction of the attempts for which the algorithm claimed to have found a minimum; we refer to this as the success rate. Since all the algorithms that we compare use the same convergence criterion described in section 3.4.3, this measure indicates to what extent the algorithm is able to avoid becoming lost in parameter space. An algorithm with a high success rate was usually able to find a minimum within the alloted number of iterations. In figure B.1 we plot the average success rate for several standard algorithms.

Although an algorithm may claim success, it may have converged by quickly evaporating parameters and failed to have actually found a good fit. To measure

a)



b)

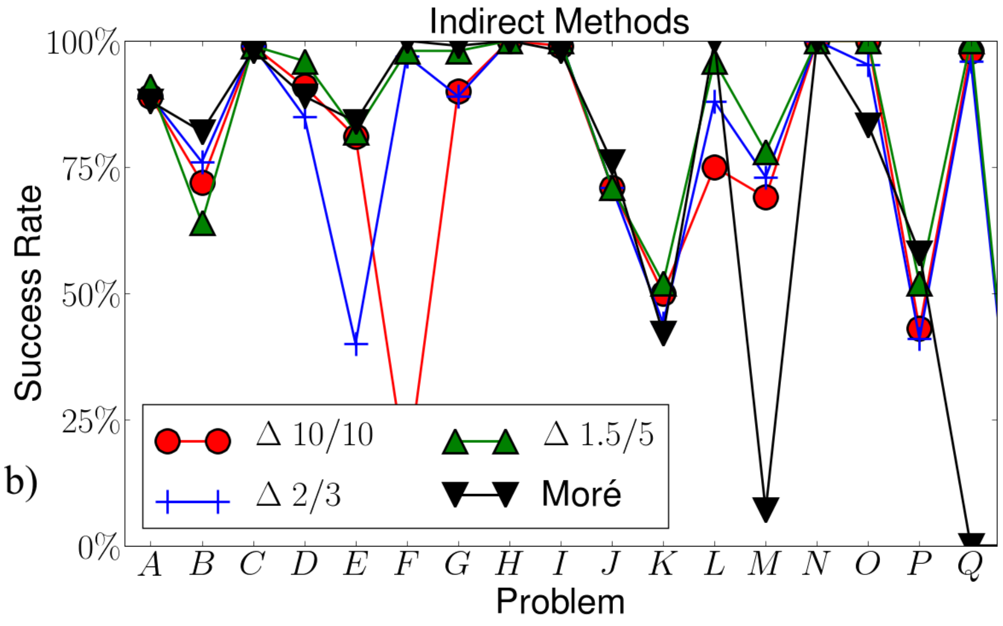


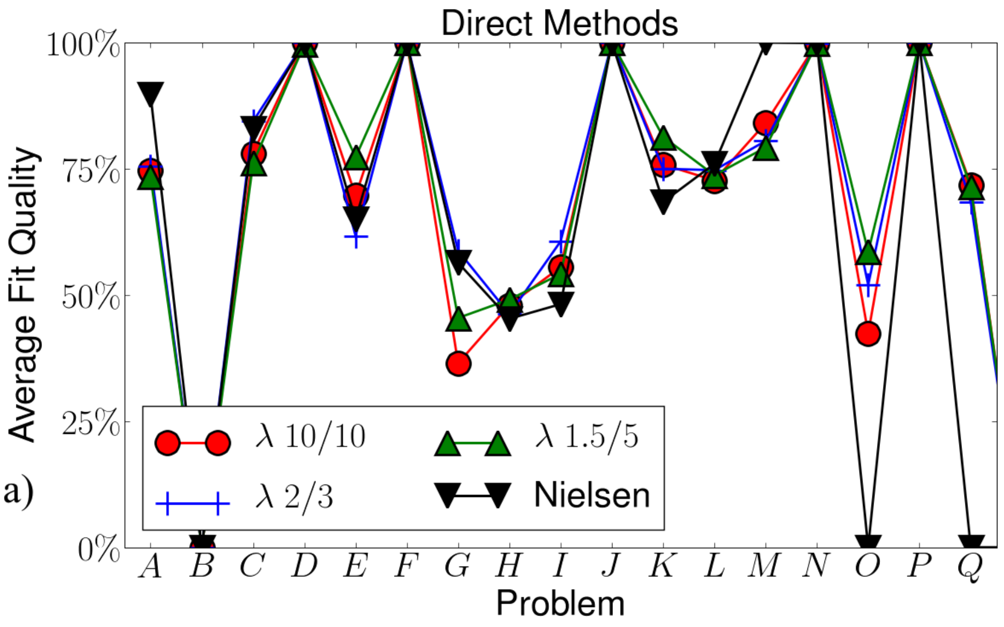
Figure B.1: The average success rates for several direct (a) and indirect(b) methods on each of the 17 test problems.

the relative quality of the fits, we define the factor

$$Q = \exp\left(1 - C_{\text{final}}/C_{\text{best}}\right), \tag{B.11}$$

where C_{final} is the final cost found by the algorithm and C_{best} is the best known cost. (Although not applicable to any of these problem, a analogous formula for a problem whose solution has zero cost is $Q = \exp -C_{\text{final}}/T$, where T is some tolerance.) This term will be very near one if the algorithm has found the best fit, and exponentially suppressed otherwise. For many of the problems from the Minpack-2 and NIST collections, the problems have either one minimum or a few minima with one much less than the others. In these cases, Eq. (B.11) will evaluate to either 0 or 1 depending on whether the best minima was found. On the other hand, for many problems, particularly problems O and Q, algorithms will converge to a variety of local minima with a wide range of final costs. In these cases, the quality factor, Q, will give partial weight to algorithms who find reasonable but not optimal fits. Figure B.2 displays the average value of this quality factor for each problem and several variations of the Levenberg-Marquardt algorithm. Note that in calculating the average quality factor, we only include results for which an algorithm claimed success.

Finally, in order to gauge the efficiency with which an algorithm converges to the best fit, we choose as a measure the number of Jacobian evaluations. The advantage of this measure is that it is easy to extrapolate results for these simple test problems to larger, more computationally intensive problems where most of the computer time is spent calculating the Jacobian matrix. Often, an algorithm will converge quickly to a poor fit. In order to not bias results in favor of algorithms which find poor fits quickly, we calculate a weighted average of the number of Jacobian evaluations, weighted by the quality factor Q in Eq. (B.11). As a convention, we plot the inverse of the average number of Jacobian evaluations

a) 

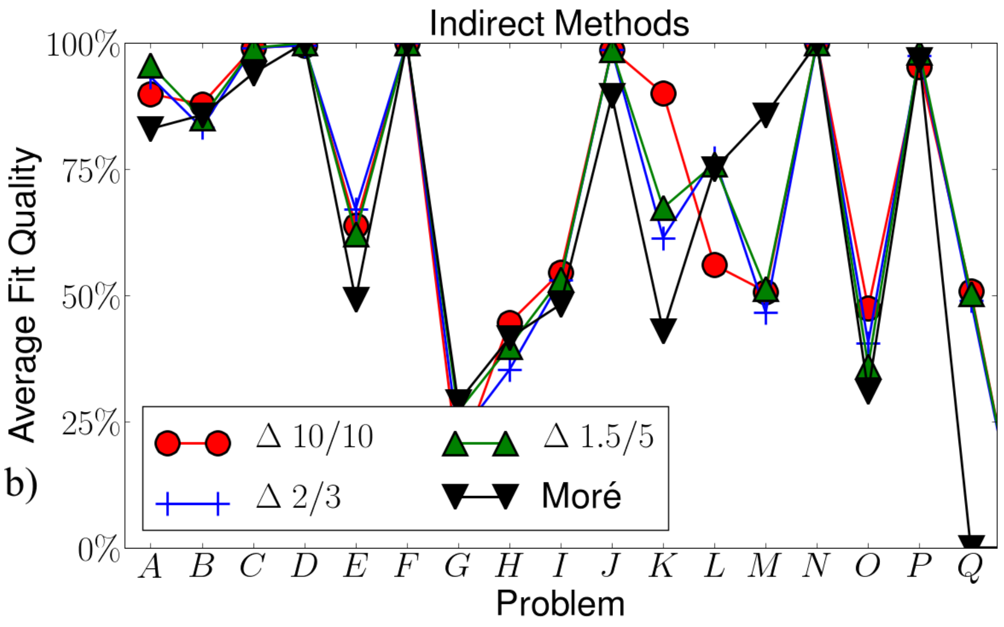
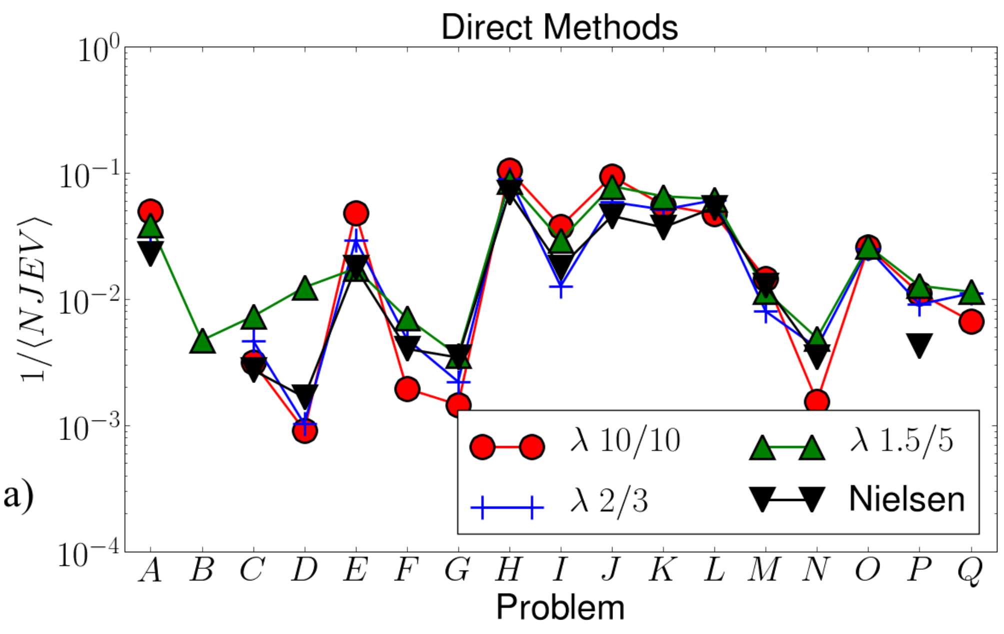
b) 

Figure B.2: The average quality factor, defined in Eq. (B.11) for several direct (a) and indirect (b) methods on each of the seventeen test problems.

so that larger numbers are preferable. The inverse average Number of Jacobian Evaluations (NJEV) each algorithm required for each problem is shown in figure B.3.

a)



b)

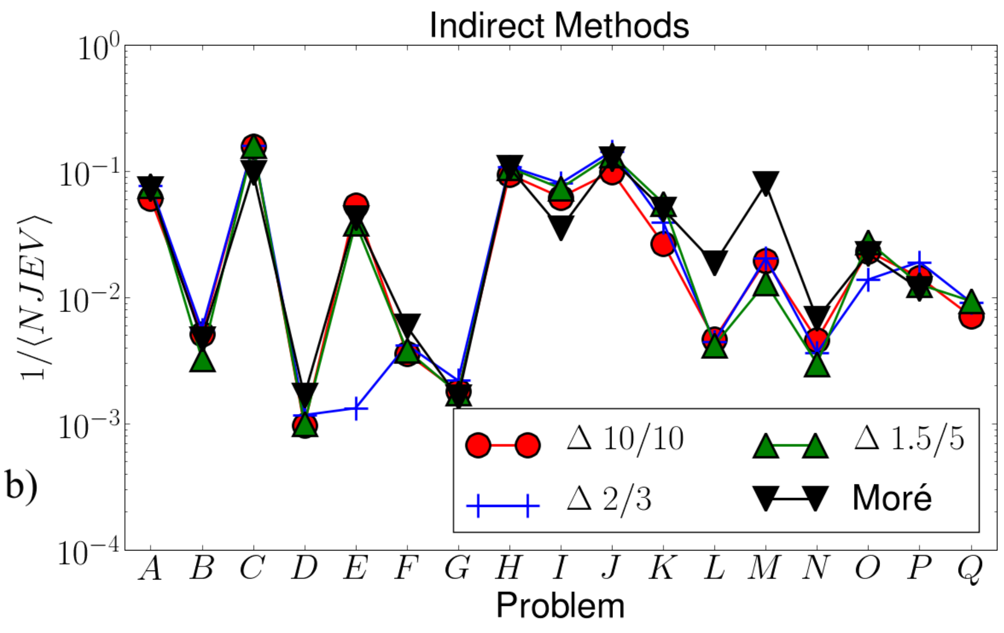


Figure B.3: The inverse average Number of Jacobian Evaluations (NJEV) necessary for convergence for several direct (a) and indirect (b) methods on each of the seventeen test problems.

APPENDIX C

APPENDICES TO CHAPTER 4

C.1 Order parameter and vector potential in the large κ limit

In this Appendix, we derive solutions to the Ginzburg-Landau equations Eq. (4.2) valid in the large-κ limit. For convenience, we work in units $\lambda = 1$, $\xi = 1/\kappa$. As a first step, we consider the limit $\kappa \to \infty$. Then Eqs. 4.2 reduce to

$$\begin{aligned} q_0'' &= f_0^2 q_0, \\ 0 &= f_0\left(f_0^2 - 1 + q_0^2\right), \end{aligned} \tag{C.1}$$

with solution [31]

$$\begin{aligned} q_0(x) &= -\frac{\sqrt{2}}{\cosh(x+\ell)}, \\ f_0(x) &= \sqrt{1-q_0^2(x)}, \end{aligned} \tag{C.2}$$

where the parameter ℓ is determined by the field at the surface via

$$H_a = q_0'(0) = \frac{\sqrt{2}\sinh\ell}{\cosh^2\ell}. \tag{C.3}$$

The above solution satisfies the boundary conditions at infinity, but it cannot satisfy the boundary condition for f at the surface. An approximate solution, valid at finite but large κ, which satisfies all boundary conditions can be obtained by boundary layer theory. We follow the approach of Ref. [24], so we only sketch the steps of the calculation. Note that away from the thermodynamic critical field, the scaling is different than that used in Ref. [24]: there the expansion is in powers of

$\kappa^{-\alpha}$ and the inner variable is $X = \kappa^\alpha x$ with $\alpha = 2/3$, here we use $\alpha = 1$:

$$\begin{aligned} q &= q_0 + \frac{1}{\kappa} q_1 + \dots \\ f &= f_0 + \frac{1}{\kappa} f_1 + \dots \end{aligned} \tag{C.4}$$

Substituting into Eqs. (4.2), we find the following "outer layer" equations for q_1 and f_1:

$$\begin{aligned} q_1'' &= 2 f_0 f_1 q_0 + f_0^2 q_1, \\ 0 &= f_1 (3 f_0^2 - 1 + q_0^2) + 2 f_0 q_0 q_1 \end{aligned} \tag{C.5}$$

which have the simple solutions $f_1 = q_1 = 0$. For the inner layer, we introduce the variable $X = \kappa x$ and find the equations

$$\begin{aligned} \tilde{f}_0'' &= \tilde{f}_0 \left(\tilde{f}_0^2 - 1 + \tilde{q}_0^2 \right), \\ \tilde{q}_0'' &= 0, \end{aligned} \tag{C.6}$$

and

$$\begin{aligned} \tilde{f}_1'' &= \tilde{f}_1 \left(3 \tilde{f}_0^2 - 1 + \tilde{q}_0^2 \right) + 2 \tilde{f}_0 \tilde{q}_0 \tilde{q}_1, \\ \tilde{q}_1'' &= 0, \end{aligned} \tag{C.7}$$

where we use tildes to denote functions of the inner variable X. Equations (C.6) have constant solutions

$$\tilde{q}_0 = -b\,, \quad \tilde{f}_0 = \sqrt{1 - b^2}, \tag{C.8}$$

while from the second of Eqs. (C.7) and the boundary conditions we get

$$\tilde{q}_1 = H_a X\,. \tag{C.9}$$

Then the first of Eqs. (C.7) becomes

$$\tilde{f}_1'' = 2(1 - b^2) \tilde{f}_1 - 2b\sqrt{1 - b^2} H_a X, \tag{C.10}$$

with solution

$$\tilde{f}_1 = \frac{b H_a}{\sqrt{1 - b^2}} X + A e^{-\sqrt{2}\sqrt{1-b^2} X} + B e^{\sqrt{2}\sqrt{1-b^2} X} \tag{C.11}$$

with A, B integration constants. Since f tends to a constant far from the surface, we set $B = 0$. Vanishing of the derivative at the surface then fixes

$$A = \frac{bH_a}{\sqrt{2}(1-b^2)} \tag{C.12}$$

Next, we match the inner and outer solutions. Comparing Eqs. (C.2) and (C.8) we get

$$b = \frac{\sqrt{2}}{\cosh \ell}. \tag{C.13}$$

We can express b in terms of the applied field using Eq. (C.3) to find

$$b = \sqrt{1 - \sqrt{1 - 2H_a^2}}. \tag{C.14}$$

Then, since $f_1 = q_1 = 0$, we need to compare the linear order expansion of Eqs. (C.2) at small x with $\tilde{q}_1/\kappa$ and $\tilde{f}_1/\kappa$ at large $X = \kappa x$. Using Eqs. (C.3) and (C.13), we find that the inner and outer solutions match. Finally, the uniform approximate solution is

$$\begin{aligned} q(x) &= q_0(x), \\ f(x) &= \sqrt{1 - q_0^2(x)} + \frac{1}{\kappa}\frac{bH_a}{\sqrt{2}(1-b^2)} e^{-\sqrt{2}\sqrt{1-b^2}\kappa x} \end{aligned} \tag{C.15}$$

with corrections of order $1/\kappa^2$. In Fig. 4.4 we compare the second of Eq. (C.15) to numerics.

C.2 Superheating field in the large-κ limit

The calculation of the superheating field H_{sh} as a function of κ for stability with respect to one-dimensional perturbations (i.e., $k = 0$) can be found in Ref. [33] for $\kappa \to 0$ and Ref. [24] for $\kappa \to \infty$. The latter calculation, however, is of little physical relevance, as the actual instability at sufficiently large κ is due to two-dimensional perturbations. Here we present for completeness (albeit in a different form) Christiansen's perturbative calculation [26] of the true superheating field $H_{\mathrm{sh}}(\kappa)$ for $\kappa \gg 1$.

Our starting point is the following expression for the "critical" second variation of the thermodynamic potential as functional of perturbations $\delta\tilde{f}$, $\delta\tilde{q}_y$, and momentum k [see also Eq. (10) in Ref. [60]]:

$$\begin{aligned}\delta^2\mathcal{F} = \int_0^\infty dx \Big\{ &\left[3f^2 + q^2 - 1 + (k/\kappa)^2\right]\delta\tilde{f}^2 + \kappa^{-2}\delta\tilde{f}'^2 \\ &+ 4fq\delta\tilde{f}\delta\tilde{q}_y + f^2\delta\tilde{q}_y^2 + (f^2+k^2)^{-1}f^2\delta\tilde{q}'^2_y\Big\}\end{aligned} \tag{C.16}$$

It is straightforward to check that variation of this functional with respect to f and q_y leads to Eqs. (4.6)-(4.7) with $E = 0$ and rescaled units $\lambda = 1$. Kramer estimated that the critical momentum $k \propto \sqrt{\kappa}$. While we will show that this is not the correct scaling, this form suggests to rescale lengths by $1/\sqrt{\kappa}$ by defining $x = w/\sqrt{\kappa}$:

$$\begin{aligned}\delta^2\mathcal{F} = \int_0^\infty \frac{dw}{\sqrt{\kappa}} \Big\{ &\left[3f^2 + q^2 - 1 + (k/\kappa)^2\right]\delta\tilde{f}^2 + \kappa^{-1}\delta\tilde{f}'^2 \\ &+ 4fq\delta\tilde{f}\delta\tilde{q}_y + f^2\delta\tilde{q}_y^2 + (f^2+k^2)^{-1}f^2\kappa\delta\tilde{q}'^2_y\Big\}\end{aligned} \tag{C.17}$$

where now prime is derivative with respect to w. (Note that although k has units of inverse length, it is momentum parallel to the surface, and therefore does not scale with x.)

Minimization with respect to k leads to the equation

$$k\int dw \left[\frac{\delta\tilde{f}^2}{\kappa^2} - \frac{\kappa f^2}{(f^2+k^2)^2}\delta\tilde{q}'^2_y\right] = 0 \tag{C.18}$$

Assuming $k \gg 1$, we can neglect $f^2 \leq 1$ in the denominator and find

$$k^4\int dw\, \delta\tilde{f}^2 = \kappa^3\int dw\, f^2\delta\tilde{q}'^2_y \tag{C.19}$$

which shows that (if our length rescaling is correct) the proper scaling for the critical momentum is $k \propto \kappa^{3/4}$. If this is true, then $(k/\kappa)^2 \propto 1/\sqrt{\kappa}$ and $\kappa/k^2 \propto 1/\sqrt{\kappa}$, which shows that the next to leading order terms in curly brackets in Eq. (C.17) are proportional to $1/\sqrt{\kappa}$. Therefore, terms of order $1/\kappa$ can be neglected and, in particular, we can neglect $\kappa^{-1}\delta\tilde{f}'^2$ and use everywhere the lowest order solution for f and q, Eq. (C.2). Hence the approximate functional in the large-κ limit is

$$\begin{aligned} \delta^2\mathcal{F} \simeq & \frac{1}{\sqrt{\kappa}}\int_0^\infty dw \Big\{ \left[2f_0^2 + (k/\kappa)^2\right]\delta\tilde{f}^2 \\ & + 4f_0q_0\delta\tilde{f}\delta\tilde{q}_y + f_0^2\delta\tilde{q}_y^2 + k^{-2}f_0^2\kappa\delta\tilde{q}_y'^2 \Big\}. \end{aligned} \tag{C.20}$$

By minimizing Eq. (C.20) with respect to $\delta\tilde{f}$, we find

$$\left[2f_0^2 + (k/\kappa)^2\right]\delta\tilde{f} = -2f_0q_0\delta\tilde{q}_y, \tag{C.21}$$

and solving for $\delta\tilde{f}$

$$\delta\tilde{f} = -\frac{2f_0q_0\delta\tilde{q}_y}{2f_0^2 + (k/\kappa)^2} \simeq -\frac{q_0\delta\tilde{q}_y}{f_0} + \left(\frac{k}{\kappa}\right)^2 \frac{q_0\delta\tilde{q}_y}{2f_0^3} \tag{C.22}$$

where in the last step we kept only the leading and the next to leading order terms. Substituting back into Eq. (C.20) gives

$$\begin{aligned} \delta^2\mathcal{F} = \int_0^\infty \frac{dw}{\sqrt{\kappa}} \Big[& \left(1 - 3q_0^2\right)\delta\tilde{q}_y^2 \\ & + \left(\frac{k}{\kappa}\right)^2 \frac{q_0^2}{f_0^2}\delta\tilde{q}_y^2 + \frac{\kappa}{k^2}f_0^2\delta\tilde{q}_y'^2 \Big] \end{aligned} \tag{C.23}$$

The first term in square brackets is the leading term. Neglecting the other terms, since q_0^2 is monotonically decreasing function of w the variation q_y that minimizes the functional is a δ-function at the surface. Then the condition for the metastability is

$$1 - 3q_0^2(0) = 0. \tag{C.24}$$

Using Eq. (C.2) we obtain

$$\cosh\ell = \sqrt{6}\,, \quad \sinh\ell = \sqrt{5} \tag{C.25}$$

and substituting into Eq. (C.3)

$$H_{\rm sh}^{\infty} = \frac{\sqrt{10}}{6} \tag{C.26}$$

To calculate the large-κ correction, we expand the function $q_0(w)$ in the first term in square brackets in Eq. (C.23) to linear order, while q_0 and f_0 in the sub-leading terms can be simply evaluated at the surface. Setting

$$\ell \simeq \mathrm{arccosh}\sqrt{6} - \frac{c}{\sqrt{\kappa}}, \tag{C.27}$$

$$k = \left(\frac{5}{6}\right)^{1/4} \tilde{k}\kappa^{3/4}, \tag{C.28}$$

and using Eq. (C.3) we find

$$\delta^2\mathcal{F} = 2\sqrt{\frac{5}{6}}\int_0^{\infty}\frac{dw}{\kappa}\left[\left(-c + w + \frac{1}{4}\tilde{k}^2\right)\delta\tilde{q}_y^2 + \frac{2}{5\tilde{k}^2}\delta\tilde{q}_y'^2\right] \tag{C.29}$$

The variational equation for $\delta\tilde{q}_y$ derived from this functional has as solution the Airy function

$$\delta\tilde{q}_y(w) = \mathrm{Ai}\left[\left(\frac{5\tilde{k}^2}{2}\right)^{1/3}\left(w - c + \frac{1}{4}\tilde{k}^2\right)\right] \tag{C.30}$$

Imposing the boundary condition $\delta\tilde{q}_y'(0) = 0$, we find that for a given $\tilde{k}$ the lowest possible c is

$$c = z_0\left(\frac{5\tilde{k}^2}{2}\right)^{-1/3} + \frac{1}{4}\tilde{k}^2, \tag{C.31}$$

where

$$z_0 \approx 1.018793 \tag{C.32}$$

is the smallest number satisfying $\mathrm{Ai}'(-z_0) = 0$. Finally minimizing c with respect to $\tilde{k}$ we find

$$\tilde{k} = \left(\frac{4}{3}z_0\right)^{3/8}\left(\frac{2}{5}\right)^{1/8} \tag{C.33}$$

and

$$c = \left(\frac{2}{5}\right)^{1/4}\left(\frac{4}{3}z_0\right)^{3/4}. \tag{C.34}$$

Substituting Eq. (C.27) into Eq. (C.3) we obtain

$$H_{\mathrm{sh}} = \frac{\sqrt{10}}{6} + \frac{2c}{3\sqrt{3\kappa}} \approx \frac{\sqrt{10}}{6} + \frac{0.3852}{\sqrt{\kappa}} \tag{C.35}$$

and from Eqs. (C.28), (C.32), and (C.33)

$$\begin{aligned} k &= \left(\frac{160}{243}\right)^{1/8} z_0^{3/8}\kappa^{3/4} \\ &\approx 0.9558\kappa^{3/4}. \end{aligned} \tag{C.36}$$

These results agree with those of Ref. [26]. We compare these two formulas with numerics in Figs. 4.2 and 4.3, respectively.

Finally, fixing the arbitrary normalization of the perturbation by requiring $\delta\tilde{q}_y(0) = 1$, using Eqs. (C.30)-(C.34), and restoring dimensions we find

$$\delta\tilde{q}_y(x) = \mathrm{Ai}\left[\left(\frac{10}{3}z_0\right)^{1/4}\frac{\sqrt{\kappa}x}{\lambda} - z_0\right] \Big/ \mathrm{Ai}[-z_0], \tag{C.37}$$

which shows that the "penetration depth" of the perturbation is of the order of the geometric average of coherence length and magnetic field penetration depth. This functional form is plotted in Fig. 4.5 for $\kappa = 50$ along with the numerically calculated $\delta\tilde{q}_y$.

BIBLIOGRAPHY

[1] P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[2] S. Amari, H. Park, and T. Ozeki. Singularities affect dynamics of learning in neuromanifolds. *Neural Computation*, 18(5):1007–1065, 2006.

[3] S.I. Amari and H. Nagaoka. *Methods of Information Geometry*. Amer Mathematical Society, 2007.

[4] C. Atkinson and A.F.S. Mitchell. Rao's distance measure. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 345–365, 1981.

[5] B.M. Averick, R.G. Carter, J.J. More, and G.L. Xue. The minpack-2 test problem collection. *Preprint MCS-P153-0694, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois*, 1992.

[6] AB Bakushinskii. The problem of the convergence of the iteratively regularized gauss-newton method. *Computational Mathematics and Mathematical Physics*, 32(9):1353–1359, 1992.

[7] OE Barndorff-Nielsen, DR Cox, and N. Reid. The role of differential geometry in statistical theory. *International statistical review*, 54(1):83–96, 1986.

[8] D.M. Bates, D.C. Hamilton, and D.G. Watts. Calculation of intrinsic and parameter-effects curvatures for nonlinear regression models. *Communications in Statistics-Simulation and Computation*, 12(4):469–477, 1983.

[9] D.M. Bates and D.G. Watts. Relative curvature measures of nonlinearity. *J. Roy. Stat. Soc*, 42:1–25, 1980.

[10] D.M. Bates and D.G. Watts. Parameter transformations for improved approximate confidence regions in nonlinear least squares. *Ann. Statist*, 9(6):1152–1167, 1981.

[11] D.M. Bates and D.G. Watts. A relative offset orthogonality convergence criterion for nonlinear least squares. *Technometrics*, 23(2):179–183, 1981.

[12] D.M. Bates and D.G. Watts. *Nonlinear Regression Analysis and Its Applications*. John Wiley, 1988.

[13] EML Beale. Confidence regions in non-linear estimation. *Journal of the Royal Statistical Society*, 22(1):41–88, 1960.

[14] E. Ben-Jacob, N. Goldenfeld, JS Langer, and G. Schön. Dynamics of interfacial pattern formation. *Physical Review Letters*, 51(21):1930–1932, 1983.

[15] E. Bodenschatz, W. Pesch, and G. Ahlers. Recent developments in rayleigh-bénard convection. *Annual review of fluid mechanics*, 32(1):709–778, 2000.

[16] R.C. Brower, D.A. Kessler, J. Koplik, and H. Levine. Geometrical approach to moving-interface dynamics. *Physical review letters*, 51(13):1111–1114, 1983.

[17] K. S. Brown. *Signal Transduction, Sloppy Models, and Statistical Mechanics*. PhD thesis, Cornell University, 2003.

[18] K.S. Brown, C.C. Hill, G.A. Calero, C.R. Myers, K.H. Lee, J.P. Sethna, and R.A. Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical biology*, 1(3):184–195, 2004.

[19] K.S. Brown and J.P. Sethna. Statistical mechanical approaches to models with many poorly known parameters. *Physical Review E*, 68(2):21904, 2003.

[20] C.G. Broyden et al. A class of methods for solving nonlinear simultaneous equations. *Math. Comp*, 19(92):577–593, 1965.

[21] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[22] FP Casey, D. Baird, Q. Feng, RN Gutenkunst, JJ Waterfall, CR Myers, KS Brown, RA Cerione, and JP Sethna. Optimal experimental design in an epidermal growth factor receptor signalling and down-regulation model. *Systems Biology, IET*, 1(3):190–202, 2007.

[23] G. Catelani and J.P. Sethna. Temperature dependence of the superheating field for superconductors in the high-κ london limit. *Physical Review B*, 78(22):224509, 2008.

[24] S.J. Chapman. Superheating field of type II superconductors. *SIAM Journal on Applied Mathematics*, pages 1233–1258, 1995.

[25] Yan-Jiun Chen, Gianfranco Durin, and James P Sethna. Sloppyscaling: Python software for automatic fits to multivariable scaling functions at critical points. http://www.lassp.cornell.edu/sethna/Sloppy/SloppyScaling/SloppyScaling.htm.

[26] P. Voetmann Christiansen. Magnetic superheating of high-[kappa] superconductors. *Solid State Communications*, 7(10):727 – 729, 1969.

[27] GPY Clarke. Marginal curvatures and their usefulness in the analysis of nonlinear regression models. *Journal of the American Statistical Association*, pages 844–850, 1987.

[28] R.D. Cook and M.L. Goldberg. Curvatures for parameter subsets in nonlinear regression. *The Annals of Statistics*, pages 1399–1418, 1986.

[29] R.D. Cook and J.A. Witmer. A note on parameter-effects curvature. *American Statistical Association*, 80(392):872–878, Dec 1985.

[30] B.C. Daniels, Y.J. Chen, J.P. Sethna, R.N. Gutenkunst, and C.R. Myers. Sloppiness, robustness, and evolvability in systems biology. *Current Opinion in Biotechnology*, 19(4):389–395, 2008.

[31] P. G. de Gennes. Vortex nucleation in type II superconductors. *Solid State Communications*, 3(6):127 – 130, 1965.

[32] E. Demidenko. Criteria for global minimum of sum of squares in nonlinear regression. *Computational Statistics and Data Analysis*, 51(3):1739–1753, 2006.

[33] A.J. Dolgert, S.J. Di Bartolo, and A.T. Dorsey. Superheating fields of superconductors: Asymptotic analysis and numerical results. *Physical Review B*, 53(9):5650–5660, 1996.

[34] J.R. Donaldson and R.B. Schnabel. Computational experience with confidence regions and confidence intervals for nonlinear least squares. *Technometrics*, 29(1):67–82, Feb 1987.

[35] G. Eilenberger. Transformation of gorkov's equation for type II superconductors into transport-like equations. *Zeitschrift für Physik A Hadrons and Nuclei*, 214(2):195–213, 1968.

[36] L.P. Eisenhart. *Riemannian geometry*. Princeton Univ Pr, 1997.

[37] HJ Fink and AG Presson. Stability limit of the superheated meissner state due to three-dimensional fluctuations of the order parameter and vector potential. *Physical Review*, 182(2):498, 1969.

[38] H. Frahm, S. Ullah, and A.T. Dorsey. Flux dynamics and the growth of the superconducting phase. *Physical review letters*, 66(23):3067–3070, 1991.

[39] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[40] S.L. Frederiksen, K.W. Jacobsen, K.S. Brown, and J.P. Sethna. Bayesian ensemble approach to error estimation of interatomic potentials. *Physical Review Letters*, 93(16):165501, 2004.

[41] D. Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

[42] VP Galaiko. Stability limits of the superconducting state in a magnetic field for superconductors of the second kind. *Soviet Journal of Experimental and Theoretical Physics*, 23:475, 1966.

[43] P.E. Gill and W. Murray. Algorithms for the solution of the nonlinear least-squares problem. *SIAM Journal on Numerical Analysis*, pages 977–992, 1978.

[44] G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:R1, 2003.

[45] G.H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[46] R.N. Gutenkunst. *Sloppiness, modeling, and evolution in biochemical networks*. PhD thesis, Cornell University, 2008.

[47] R.N. Gutenkunst, F.P. Casey, J.J. Waterfall, C.R. Myers, and J.P. Sethna. Extracting falsifiable predictions from sloppy models. *Annals of the New York Academy of Sciences*, 1115(1 Reverse Engineering Biological Networks: Opportunities and Challenges in Computational Methods for Pathway Inference):203–211, 2007.

[48] R.N. Gutenkunst, J.J. Waterfall, F.P. Casey, K.S. Brown, C.R. Myers, and J.P. Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Comput Biol*, 3(10):e189, 2007.

[49] LM Haines, TE O Brien, and GPY Clarke. Kurtosis and curvature measures for nonlinear regression models. *Statistica Sinica*, 14(2):547–570, 2004.

[50] D.C. Hamilton, D.G. Watts, and D.M. Bates. Accounting for intrinsic nonlinearity in nonlinear regression parameter inference regions. *Ann. Statist*, 10(38):393, 1982.

[51] HO Hartley. The modified gauss-newton method for the fitting of non-linear regression functions by least squares. *Technometrics*, pages 269–280, 1961.

[52] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation.* Westview Press, 1991.

[53] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* American Mathematical Society, 1999.

[54] C. Igel, M. Toussaint, and W. Weishui. Rprop using the natural gradient. *Trends and Applications in Constructive Approximation, International Series of Numerical Mathematics*, 151, 2005.

[55] T.T. Ivancevic. *Applied differential geometry: a modern introduction.* World Scientific Pub Co Inc, 2007.

[56] H. Jeffreys. *Theory of probability.* Oxford University Press, USA, 1998.

[57] E. Jones, T. Oliphant, P. Peterson, et al. Scipy: Open source scientific tools for python. *URL http://www. scipy. org*, 2001.

[58] R.E. Kass. Canonical parameterizations and zero parameter-effects curvature. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 86–92, 1984.

[59] L. Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT Numerical Mathematics*, 15(1):49–57, 1975.

[60] L. Kramer. Stability limits of the meissner state and the mechanism of spontaneous vortex nucleation in superconductors. *Phys. Rev.*, 170(2):475–480, Jun 1968.

[61] L. Kramer. Breakdown of the superheated meissner state and spontaneous vortex nucleation in type ii superconductors. *Zeitschrift für Physik A Hadrons and Nuclei*, 259(4):333–346, 1973.

[62] L. Kramer and W. Pesch. Core structure and low-energy spectrum of isolated vortex lines in clean superconductors att ≪ t c. *Zeitschrift für Physik A Hadrons and Nuclei*, 269(1):59–64, 1974.

[63] M. Lampton. Lamping-undamping strategies for the levenberg-marquardt nonlinear least-squares method. *Computers in Physics*, 11(1):110–115, 1997.

[64] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164–168, 1944.

[65] D.G. Luenberger. The gradient projection method along geodesics. *Management Science*, pages 620–631, 1972.

[66] R. Mahony. *Optimization algorithms on homogeneous spaces.* PhD thesis, Australian National University, 1994.

[67] R. Mahony and J.H. Manton. The geometry of the newton method on non-compact lie groups. *Journal of Global Optimization*, 23(3):309–327, 2002.

[68] J.H. Manton. On the various generalisations of optimisation algorithms to manifolds. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems, Leuven, Belgium*, 2004.

[69] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[70] B.D. McCullough. Assessing the reliability of statistical software: Part I. *The American Statistician*, 52(4):358–366, 1998.

[71] C.W. Misner, K.S. Thorne, and J.A. Wheeler. *Gravitation.* WH Freeman and Company, 1973.

[72] J.J. Moré. The levenberg-marquardt algorithm: implementation and theory. *Lecture notes in mathematics*, 630:105–116, 1977.

[73] J.J. Moré, B.S. Garbow, and K.E. Hillstrom. *User guide for MINPACK-1*, 1980.

[74] JJ Mortensen, K. Kaasbjerg, SL Frederiksen, JK Nørskov, JP Sethna, and KW Jacobsen. Bayesian error estimation in density-functional theory. *Physical Review Letters*, 95(21):216401, 2005.

[75] M.K. Murray and J.W. Rice. *Differential geometry and statistics*. Chapman & Hall New York, 1993.

[76] Christopher R. Myers. *Sliding Charge Density Waves: Dynamics and Criticality in Many Degrees-of-Freedom*. PhD thesis, Cornell University, 1991.

[77] H.B. Nielsen. Damping parameter in marquardt's method. *Department of mathematical modelling, Technical University of Denmark, Tech. Rep. IMM-REP-1999-05*, 1999.

[78] Y. Nishimori and S. Akaho. Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[79] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 1999.

[80] H. Padamsee, KW Shepard, and R. Sundelin. Physics and accelerator applications of rf superconductivity. *Annual Review of Nuclear and Particle Science*, 43(1):635–686, 1993.

[81] A. Pázman. Results on nonlinear least squares estimators under nonlinear equality constraints. *Journal of Statistical Planning and Inference*, 103(1-2):401–420, 2002.

[82] RLM Peeters. On a riemannian version of the levenberg-marquardt algorithm. Serie Research Memoranda 0011, VU University Amsterdam, Faculty of Economics, Business Administration and Econometrics, 1993.

[83] W.H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes: the art of scientific computing,*. Cambridge University Press, 2007.

[84] C.R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Vull. Calcutta Math. Soc.*, 37:81–91, 1945.

[85] CR Rao. On the distance between two populations. *Sankhya*, 9:246–248, 1949.

[86] S.T. Smith. *Geometric optimization methods for adaptive filtering*. PhD thesis, Harvard University, 1993.

[87] S.T. Smith. Optimization techniques on riemannian manifolds. *Hamiltonian and gradient flows, algorithms and control*, 3:113–136, 1994.

[88] M. Spivak. *A comprehensive introduction to differential geometry.* Publish or Perish, 1979.

[89] J. Stoer, R. Bulirsch, W. Gautschi, and C. Witzgall. *Introduction to numerical analysis.* Springer Verlag, 2002.

[90] M. Tinkham. *Introduction to superconductivity.* Dover Pubns, 2004.

[91] Mark K. Transtrum, Gianluigi Catelani, and James P. Sethna. Superconducting superheating field in Eilenberger theory. in preparation.

[92] Mark K. Transtrum, Gianluigi Catelani, and James P. Sethna. Superheating field of superconductors within Ginzburg-Landau theory. *Physical Review B*, 83(9):094505, 2011.

[93] Mark K. Transtrum, Benjamin B. Machta, and James P. Sethna. See supplemental material at [http://link.aps.org/supplemental/ 10.1103/PhysRevE.83.036701] for an animation of this figure.

[94] Mark K Transtrum, Benjamin B Machta, and James P Sethna. Why are nonlinear fits to data so challenging? *Physical Review Letters*, 104(060210):1060201, 2010.

[95] Mark K. Transtrum, Benjamin B. Machta, and James P. Sethna. Geometry of nonlinear least squares with applications to sloppy models and optimization. *Physical Review E*, 83(3):036701, 2011.

[96] Mark K. Transtrum, Benjamin B. Machta, Cyrus Umrigar, Peter Nightingale, and James P. Sethna. Development and comparison of algorithms for nonlinear least squares fitting. In preparation.

[97] C. Udriste. *Convex functions and optimization methods on Riemannian manifolds.* Kluwer Academic Pub, 1994.

[98] NRA Valles and MU Liepe. Temperature dependence of the superheating field in niobium. *Arxiv preprint arXiv:1002.3182*, 2010.

[99] J.J. Waterfall, F.P. Casey, R.N. Gutenkunst, K.S. Brown, C.R. Myers, P.W. Brouwer, V. Elser, and J.P. Sethna. Sloppy-model universality class and the vandermonde matrix. *Physical Review Letters*, 97(15):150601, 2006.

[100] B.C. Wei. On confidence regions of embedded models in regular parametric families (a geometric approach). *Australian & New Zealand Journal of Statistics*, 36(3):327–338, 1994.

[101] Y. Yang. Globally convergent optimization algorithms on riemannian manifolds: Uniform framework for unconstrained and constrained optimization. *Journal of Optimization Theory and Applications*, 132(2):245–265, 2007.

[102] I.C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.