Geometry of nonlinear least squares with applications to sloppy models and optimization

Mark K. Transtrum, Benjamin B. Machta, and James P. Sethna
Laboratory of Atomic and Solid State Physics, Cornell University, Ithaca, New York 14853, USA

(Received 8 October 2010; published 3 March 2011)

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-effect 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 parametrization as an extremely skewed coordinate basis and show that it induces a large parameter-effect curvature on 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-effect curvature, improving both efficiency and success rates at finding good fits.

DOI: 10.1103/PhysRevE.83.036701 PACS number(s): 02.60.Ed, 02.40.Ky, 02.60.Pn, 05.10.−a

I. INTRODUCTION

A 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

$$r_m(\theta) = \frac{y_m - f(t_m,\theta)}{\sigma_m} \tag{1}$$

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

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

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) is a likelihood function, with the most likely, or best-fit, parameters being those that minimize the cost function, C, defined by

$$C(\theta) = \frac{1}{2}\sum_m r_m(\theta)^2, \tag{3}$$

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 Sec. III, 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 by which 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 [1–7], models from many diverse fields have been shown to lie within the sloppy model universality class [8].

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 [9], and later continued by Rao [10,11] and many others [12,13]. An equivalent, alternative formulation began with Beale in 1960 [14], and continued with the work of Bates and Watts [15–18] and others [19–21]. The authors have used this geometric approach previously to explain the extreme difficulty of the data-fitting process [22], of which this work is a continuation.

In Sec. II, we present a review of the phenomenon of sloppiness and describe the *model manifold*, that is, 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 Sec. III. 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 Sec. IV 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 Sec. V. 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 Sec. VI, 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 VII discusses several measures of curvature and explores curvature of sloppy models. We show that the *parameter-effect* 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 Sec. VIII, 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.

II. 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 1(a) 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 contours of constant cost in parameters space, as is done for our toy model in Fig. 1(b). 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 Sec. IV. 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 ratio much more extreme than in Fig. 1(b), typically 1000 : 1 or more for models with 10 or more parameters [6].

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. 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

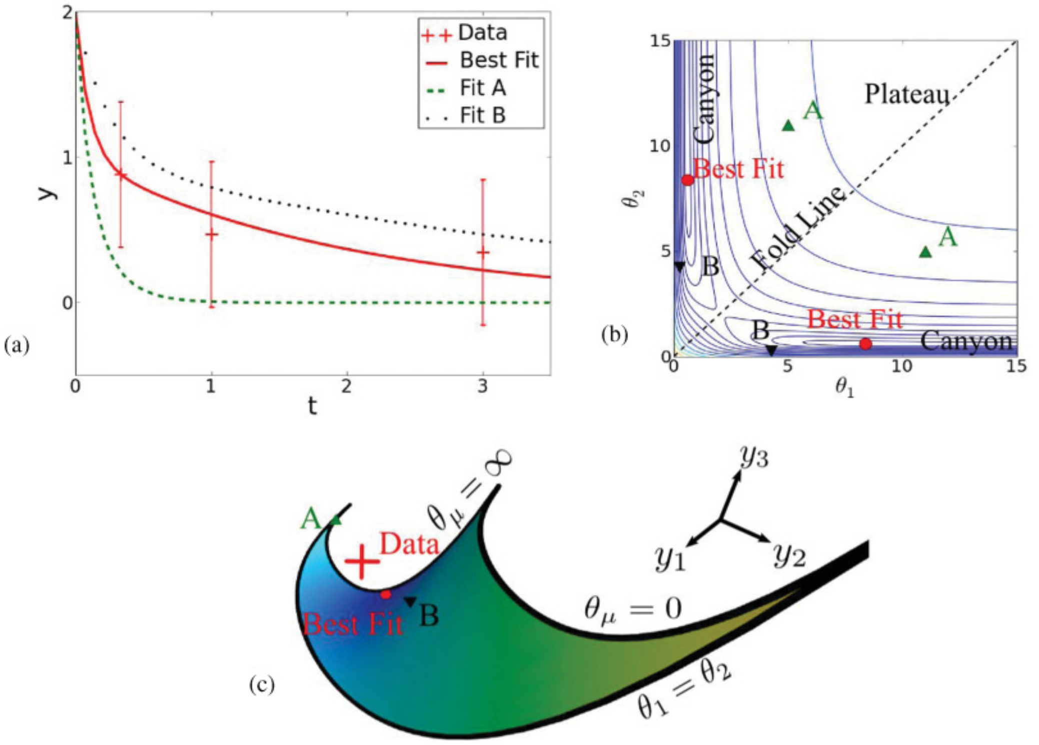

FIG. 1. (Color online) (a) Fitting a nonlinear function to data, in this case the sum of two exponentials to three data points. Fit A has rate constants that decay too quickly, resulting in a poor fit; B is an improvement over fit A, although the rates are too slow; the best fit minimizes the cost (the 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 are represented by a single point. The set of all possible fitting parameters induces 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 [23].

given by

$$\begin{aligned} H_{\mu\nu} &= \partial_\mu \partial_\nu C \\ &= \sum_m \partial_\mu r_m \partial_\nu r_m + \sum_m r_m \partial_\mu \partial_\nu r_m && (4) \\ &\approx \sum_m \partial_\mu r_m \partial_\nu r_m && (5) \\ &= (J^T J)_{\mu\nu}. && (6) \end{aligned}$$

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 correspond to the residuals being exactly zero, the Hessian is usually dominated by the term in Eq. (5) when evaluated at the best fit. Furthermore, the dominant term, $J^T J$, is an important quantity geometrically that 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. 1(b) 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 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, that is,

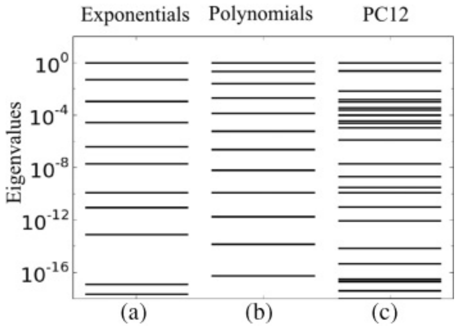

FIG. 2. Hessian eigenvalues for three sloppy models. Note the extraordinarily large range of eigenvalues (15–17 orders of magnitude, corresponding to valley aspect ratios of 10^7–10^9) in Fig. 1(b). Notice also the roughly equal fractional spacing between eigenvalues—there is no clean separation between 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 Sec. VII and as a test problem to compare algorithms in Sec. VIII E. (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 [2], which also has a sloppy eigenvalue spectrum. Many more examples can be found in [6,8].

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 a 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 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 that we refer to as *data space*. By considering Eq. (1), we see that the residual vector is the difference between a vector representing the data and a 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 [18,24]. The model manifold of our toy model is shown in Fig. 1(c). 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 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 [25–33]. 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

$$dr^2 = (J^T J)_{\mu\nu} d\theta^\mu d\theta^\nu. \tag{7}$$

Equation (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. (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 [34]. 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 as the *eigenpredictions*. The product of singular values describes the mapping of local volume elements of parameter space to data space. A unit hypercube 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 Sec. VI. 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

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

or in terms of the residuals

$$\Gamma^{\alpha}_{\mu\nu} = g^{\alpha\beta} \sum_m \partial_\beta r_m \partial_\mu \partial_\nu r_m, \tag{8}$$

where $g^{\mu\nu} = (g^{-1})^{\mu\nu}$. One could now also calculate the Riemann curvature by application of the standard formulas; however, we postpone a discussion of curvature until Sec. VII. For a more thorough discussion of concepts from differential geometry, we refer the reader to any text on the subject [35–38].

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 that induces a metric on the model manifold. Our approach is in the spirit of Bates and Watts' treatment of the subject [15–18]. 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 [9–13]. 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

$$P^T = \delta - P^N = J(g^{-1})J^T, \tag{9}$$

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 zigzag along the bottom of the canyon and only slowly approach the best fit [34].

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

$$\begin{aligned}\vec{v} &= -P^T \vec{r} \\ &= -J(g^{-1})J^T \vec{r} \\ &= -J(g^{-1})\nabla C \\ &= -\vec{J}_\mu g^{\mu\nu} \nabla_\nu C,\end{aligned} \tag{10}$$

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

$$\vec{v} = \vec{J}_\mu v^\mu; \tag{11}$$

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

$$v^\mu = -g^{\mu\nu} \nabla_\nu C. \tag{12}$$

The term "Gauss-Newton" direction comes from the fact that it is the direction given by the Gauss-Newton algorithm described in Sec. VIII A. 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 is a consequence of bad parametrization on the manifold, which this parameter-independent approach can help to remedy. Most sophisticated algorithms, such as conjugate gradient and Levenberg-Marquardt algorithms, attempt to follow the Gauss-Newton direction as much as possible so as not to 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. (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 Fig. 3. By reparametrizing, one can remove the skewedness and construct a more natural coordinate mesh. We will revisit this idea in Sec. VI. We will argue in this paper that, on the contrary, there is a geometrical component to sloppy nonlinear models that is independent of parametrization and in most cases that the human-picked "bare" parameters naturally illuminate the sloppy intrinsic structure of the model manifold.

In the original parametrization, 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

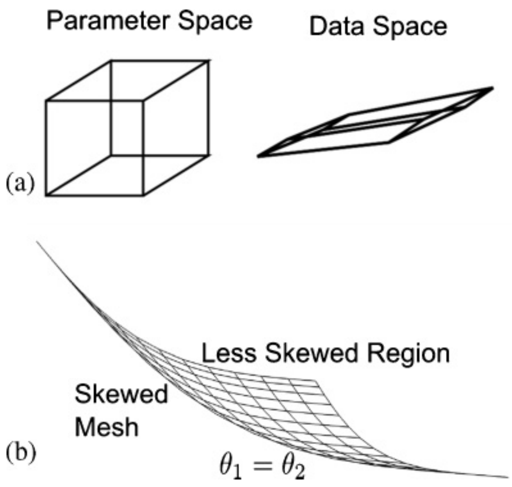

FIG. 3. Skewed coordinates. A sloppy model is characterized by a skewed coordinate mesh on the manifold. The volume of the parallelpiped is given by the determinant of the metric, which is equal to the product of the eigenvalues. Because sloppy models have many tiny eigenvalues, these volumes can be 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.

is covered by a very small region of parameter space that 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 corresponds 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. 1(c). The surface drawn in the figure corresponds to the patch of parameters within $0 \leqslant \theta_1,\theta_2 \leqslant \infty$. 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 discussed further in Sec. IV. 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 Sec. IV 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 Sec. V, we show that introducing other priors can be even more helpful in keeping algorithms away from the boundaries.

III. 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. 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. 4 are fairly generic; they come from general theorems governing the interpolation of functions. 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

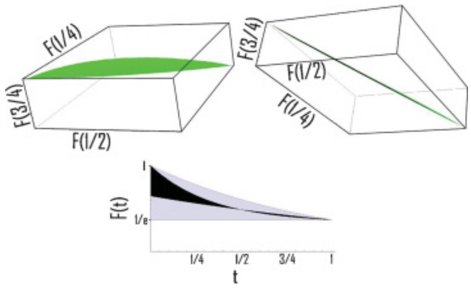

FIG. 4. (Color online) Fixing a few data points greatly restricts the possible range of the model behavior between those data points (lower). This is a consequence of interpolation of analytic functions. In this case, $f(t)$ is a sum of three exponentials with six parameters (amplitudes and rates). Shown 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 [23].) 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 Sec. V.

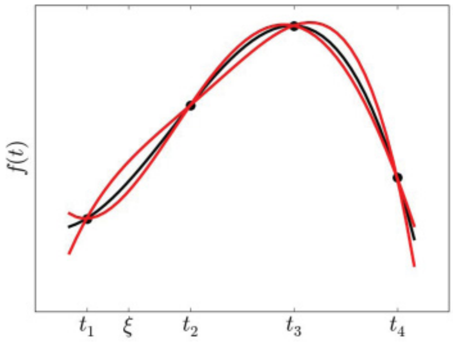

FIG. 5. (Color online) The possible values of a model at intermediate time points are restricted by interpolating theorems. Taking cross sections of the model manifold corresponds to fixing the model values at a few time points, restricting the possible values at the remaining times. Therefore, the model manifold will have a hierarchy of progressively thinner widths, much like a hyper-ribbon.

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 $(t_1,t_2,\ldots,t_n)$, 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

$$f(t) - P_{n-1}(t) = \frac{\omega(t) f^{(n)}(\xi)}{n!}, \tag{13}$$

where $f^{(n)}(t)$ is the nth derivative of the function and ξ lies somewhere in the range $t_1 < \xi < t_n$ [39]. The polynomial $\omega(t)$ has roots at each of the interpolating points

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

By inspecting Eq. (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. 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

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

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

$$f(t,\theta) = \sum_m \theta_m t^m. \tag{14}$$

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 $\sum_m \theta_m^2 \leqslant 1$. This constraint enforces the condition that higher derivatives of the function become small (roughly that the radius of convergence is 1) and correspond to the unit hyper-sphere in parameter space. If this function is sampled at time points $(t_1,t_2,\ldots,t_n)$, then the model vector in data space can be written as

$$\vec{f} = \begin{pmatrix} 1 & t_1 & t_1^2 & \cdots \\ 1 & t_2 & t_2^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_n & t_n^2 & \cdots \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \end{pmatrix}. \tag{15}$$

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 [8] 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. (14), the corresponding model manifold is limited to a hyperellipse in data space. The principal axes of this hyperellipse are the eigenprediction directions we discussed in Sec. II. 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 several widths of the corresponding model manifold in Fig. 6(a), where we see that they are accurately predicted by the singular values of the Jacobian. The widths in Fig. 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 an eigenprediction of the Jacobian that lies within the tangent plane. The $(M-N+1)$-dimensional hyperplane spanned by these unit vectors intersects the model manifold along a one-dimensional curve. The width can be

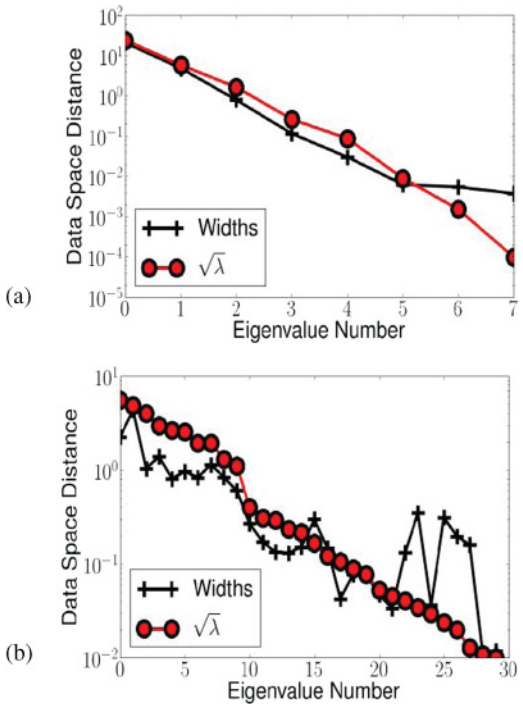

FIG. 6. (Color online) (a) Geodesic cross-sectional widths of an eight-dimensional model manifold along the eigendirections of the metric from some central point, together with the square root of the eigenvalues (singular values of the Jacobian) [22]. Notice the hierarchy of these data-space distances—the 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.

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,

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

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 have been normalized according to $y(t=0) \leqslant 1$, then it is natural to consider the hypertetrahedron of parameter space given by $A_n > 0$ and $\sum A_n \leqslant 1$. 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 [40]. 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. Figure 6(b) shows indeed that the singular values of the Jacobian agree with geodesic widths again for this model.

The result in Fig. 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 the 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 1. 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. 6 are particularly easy to fit to data. The fact that from a centrally located point geodesics can explore 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 [2], we find that 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 hypercorner 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 Sec. VI that algorithms are sluggish as a result of this poor parametrization. Figure 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 Sec. VII D. We will see that in addition to a hierarchy of boundaries, the manifold typically has a hierarchy of extrinsic and parameter-effect curvatures whose scales are set by the smallest and widest widths, respectively.

We argue elsewhere [22] that the ubiquity of sloppy models, appearing everywhere from models in systems biology [6], insect flight [8], variational quantum wave functions, interatomic potentials [41], and a model of the next-generation international linear collider [7], 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 multidimensional interpolation. Only a few stiff parameter combinations need to be tuned 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 [1–3,5].

For our fitting exponential example, part of the model boundary was the "fold lines" where pairs of the exponents are equal (see Fig. 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 exists a set of nonzero α^μ's for which

$$\alpha^\mu \vec{e}_\mu = 0. \tag{16}$$

To satisfy Eq. (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 nonextreme values of the parameters.

For models with $M > 2N$, we do not expect Eq. (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. (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 [42]. More generally, the process of fitting a sloppy model to data involves the frustrating experience of applying a black-box algorithm to the problem that 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.[1] 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 Sec. II 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 overestimated or underestimated the parameters, the Gauss-Newton direction can point toward the boundary of the manifold, as does fit A in Fig. 7. If one considers the parameter space picture, the Gauss-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. 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 Sec. VII G. Therefore, we 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

[1] The term parameter evaporation was originally used to describe the drift of parameters to infinite values in the process of Monte Carlo sampling [66]. 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.

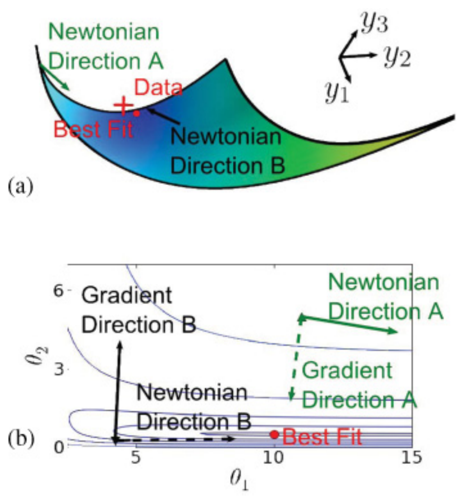

FIG. 7. (Color online) (a) Falling off the edge of the model manifold. The manifold in data space defines a "natural" direction, known as the Gauss-Newton direction, in which an algorithm will try to follow to the best fit. Often this direction will push parameters toward the edge of the manifold. (b) 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.

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 it is the subject of the next section. This natural interpolation is exploited by the Levenberg-Marquardt algorithm, which we discuss in Sec. VIII.

IV. THE MODEL GRAPH

We saw in Sec. III 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 Sec. II, the model manifold picture is a view that is complementary to the parameter space picture, as illustrated in Fig. 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 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, that is, 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

$$g_{\mu\nu} = g^0_{\mu\nu} + \lambda D_{\mu\nu}, \tag{17}$$

where $g^0_{\mu\nu} = (J^T T)_{\mu\nu}$ is the metric of the model manifold and $D_{\mu\nu}$ is the metric of parameter space. We discuss common parameter space metrics below. We have introduced the free parameter λ in Eq. (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. (17) the Levenberg metric because of its role in the Levenberg algorithm [43]. 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 [44] and has the advantage that the resulting method is invariant to rescaling the parameters, that is, 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 [45]. This method is scale invariant but guarantees that D is always positive definite. We discuss these algorithms further in Sec. VIII.

It is our experience that the Marquardt metric is much less useful than the Levenberg metric for preventing parameter evaporation. While it may seem counterintuitive 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 natural units will be important in the discussion of priors in Sec. V. 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. (8). The connection with a raised

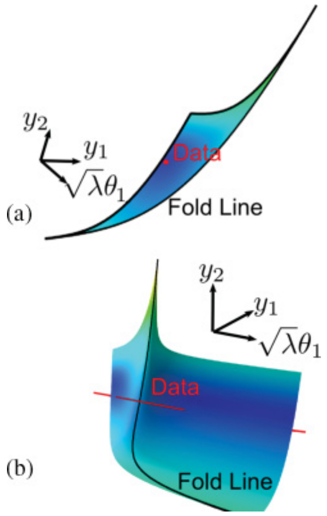

FIG. 8. (Color online) The effect of the damping parameter is to produce a new metric for the surface induced by the graph of the model vs the input parameters. (a) Model graph, $\lambda = 0$. If the parameter is zero, then the resulting graph is simply the original model manifold, with no extent in 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 [23].

index will include a 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. (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 three-dimensional projections of the embedding space. In Fig. 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. (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 sloppy model span many orders of magnitude, this means that all the eigenvalues that were originally less than λ are cut off 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 Sec. VIII. We demonstrate how λ can interpolate between the two natural directions for our example model in Fig. 9.

V. 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 preexisting measurements of the parameters $\theta_m = \theta_m^0 \pm \sigma_m$ with normally distributed uncertainties, then the probability density would be $\prod_m 1/\sqrt{2\pi\sigma_m^2} \exp[-(\theta_m - \theta_m^0)^2/(2\sigma_m^2)]$ 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,"

$$r_m = \left(\theta_m - \theta_m^0\right)/\sigma_m \tag{18}$$

(interpreting the preexisting 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.

That is, we want to use priors to prevent parameter combinations that 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 Sec. VII G, 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.

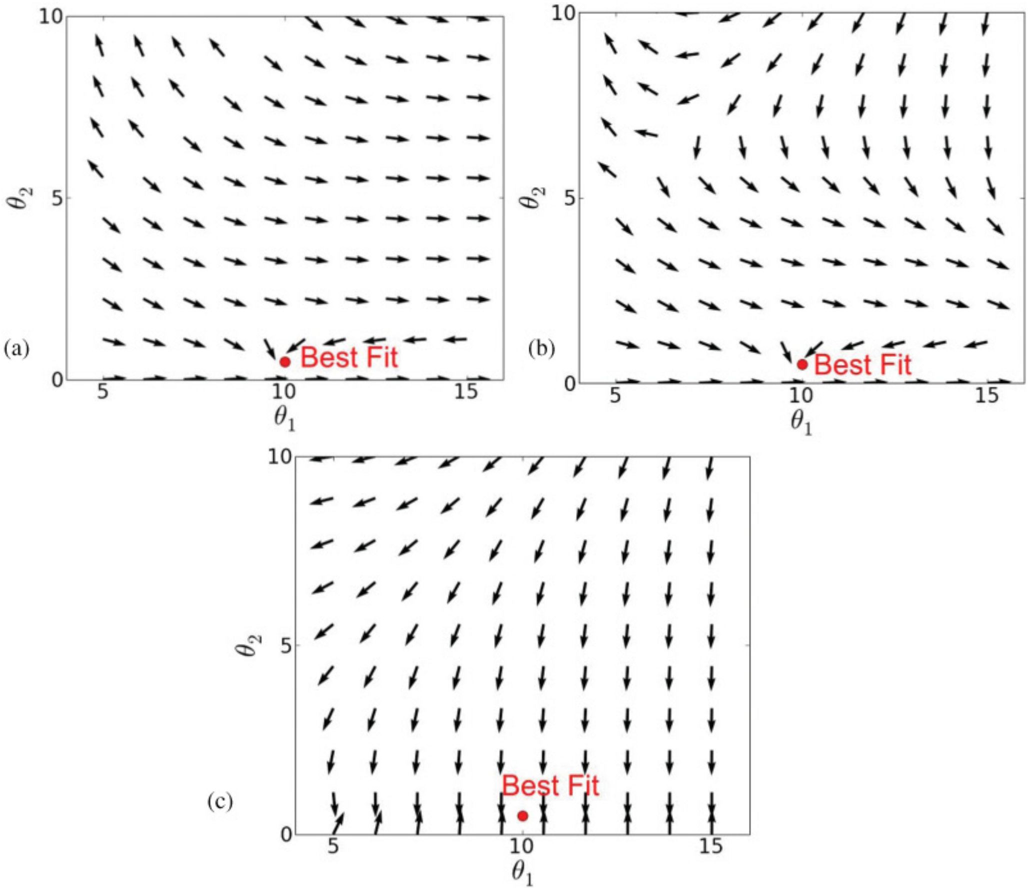

FIG. 9. (Color online) (a) Gauss-Newton directions. The Gauss-Newton direction is prone to pointing parameters toward infinity, especially in regions where the metric has very small eigenvalues. (b) Rotated Gauss-Newton directions. By adding a small damping parameter to the metric, the Gauss-Newton direction 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.

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. 7(a) 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, Ref. [2]):

$$\frac{d[x^*]}{dt} = \frac{k_x[y^*][x]}{1 + km_x[x]}. \tag{19}$$

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 1, then we could have modeled the system with one less parameter $k = k_x/km_x$; if it were much less than 1, the second term in the denominator could have been dropped entirely. Furthermore, if the data are best fit by one of these boundary cases, say $km_x x_{\max} \to \infty$, they will be fit quite well by taking $km_x x_{\max} \gg 1$, 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 that prevents $k_x y_{\max} x_{\max}/\tau$ from being far from 1, where τ is the typical time scale 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 1.

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 bookkeeping.) In these natural units, we expect all parameters to be of 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 that punishes high values and one that punishes small values:

$$\mathrm{Pr}(\theta) = \begin{pmatrix} \sqrt{w_h \theta} \\ \sqrt{w_l/\theta} \end{pmatrix}. \tag{20}$$

This prior has a 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

$$g_{\mu\nu} = \partial_\mu r^{0i} \partial_\nu r^{0i} + \partial_\mu \mathrm{Pr}(\theta) \partial_\nu \mathrm{Pr}(\theta) \tag{21}$$

$$= g^0_{\mu\nu} + \delta_{\mu\nu}\left(\frac{w_l}{\theta^\mu} + w_h \theta^\mu\right), \tag{22}$$

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 0.001 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,

$$C = C_0 + \sum_i w_h \theta_i/2 + w_l/(2\theta_i). \tag{23}$$

The second derivative of the extra cost contribution with respect to the log of the parameters is given by $\frac{\partial^2}{\partial \log(\theta)^2}(\frac{Pr(\theta)^2}{2}) = \frac{w_h \theta}{2} + \frac{w_l}{2\theta}$. 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 [2]. 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 0.1 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 reminimize. The results are plotted in Fig. 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 Sec. IV, 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. (17), and the metric for our particular choice of prior, Eq. (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. (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

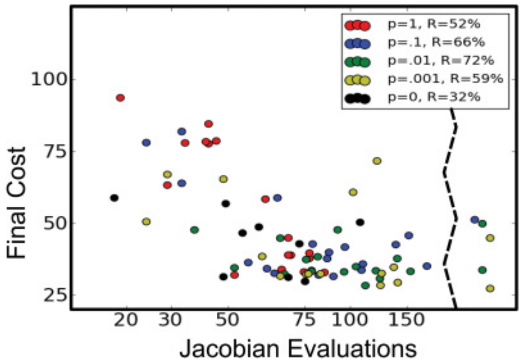

FIG. 10. (Color online) The final cost is plotted against the number of Jacobian evaluations for five strengths of priors. After minimizing with priors, the priors are removed and a maximum of 20 further Jacobian evaluations are performed. The prior strength is measured by p, with $p = 0$ meaning no prior. The success rate is R. The strongest priors converge the fastest, with medium strength priors showing the highest success rate.

boundary, but not biasing the algorithm permanently toward sensible values. Despite the use of a variable λ that can be used to tune the algorithm toward sensible behavior (Fig. 9), we shall see in Sec. VIII 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.

VI. 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

$$\ddot{x}^\mu + \Gamma^\mu_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta = 0, \tag{24}$$

where $\Gamma^\mu_{\alpha\beta}$ are the connection coefficients given by Eq. (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 polarlike 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 the 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 Sec. II). 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 1, 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 appear 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

$$\gamma^\nu \approx \theta^\nu_{\rm bf} + v^\nu_\mu \delta\theta^\mu + \tfrac{1}{2}\Gamma^\nu_{\alpha\beta}\delta\theta^\alpha\delta\theta^\beta. \tag{25}$$

The coordinates given in Eq. (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 [35].

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. 11.

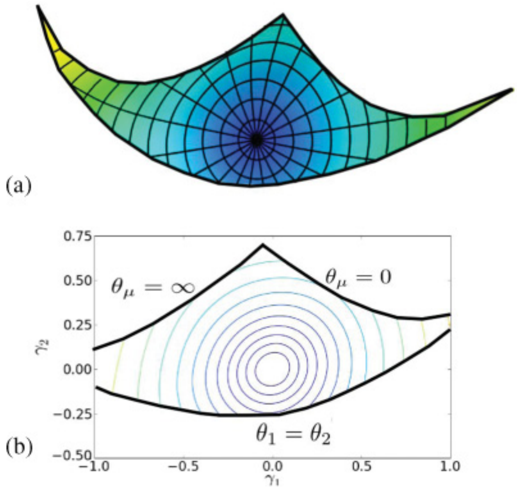

FIG. 11. (Color online) (a) Extended geodesic coordinates. The parameters of a model are not usually well suited to describing the behavior of a model. By considering the manifold induced in data space, one can construct more natural coordinates based on geodesic motion that are more well-suited to describing the behavior of a model (black grid). These coordinates remove all parameter-effect 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 nonlinearity does not translate into large extrinsic curvature. This type of nonlinearity is known as parameter-effect 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.

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 anharmonic "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. 11(b) 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 us 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

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

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

$$\frac{d^2C}{d\tau^2} = v^\mu v^\nu g_{\mu\nu} + r_m P^N_{mn}\partial_\mu\partial_\nu r_n \frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}. \qquad (26)$$

The term $v^\mu v^\nu g_{\mu\nu}$ in Eq. (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. (26) is the expression

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

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

Recognizing P^N in Eq. (26), we see that any deviation of the quadratic behavior of the cost will be when the nonlinearity 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,

$$P^N = \delta - J(J^TJ)^{-1}J^T = \delta - JJ^{-1}(J^T)^{-1}J^T = 0.$$

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 Sec. III 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 Sec. VII that most of the nonlinearities of a sloppy model do not produce extrinsic curvature, meaning the manifold is typically much more flat than one would have guessed.

Nonlinearities that do not produce extrinsic curvature are described as parameter-effect curvature [15]. As the name suggests, these are "curvatures" that can be removed through a different choice of parameter. By using geodesics, we have found a coordinate system on the manifold that removes all parameter-effect curvature at a point. It has been noted previously that geodesics are linked to zero parameter-effect curvature [46].

We believe it to be generally true for sloppy models that nonlinearities are manifested primarily as parameter-effect curvature, as we argue in [22] and in Sec. VII. 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. 11(b), 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 Sec. IV 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. 11(b) and the long canyon that is characteristic of parameter space. This can be seen clearly in Fig. 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. 11 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.

VII. 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 nonlinearity of a model [15,16,18]. This will be critical when we discuss the implications of geometry on numerical algorithms, since it is the nonlinearity that makes these problems difficult.

Curvature has also been used to study confidence regions [16,20,47–49], kurtosis (deviations from normality) in parameter estimation [50], and criteria for determining if a minimum is the global minimizer [51]. 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 can accurately step (Sec. VII F) and to estimate how many parameters a best fit will typically evaporate (Sec. VII G).

In our discussion of geodesic coordinates in Sec. VI, we saw how some of the nonlinearity of a model could be removed by a clever choice of coordinates. We also argued that the nonlinearity that could not be removed by a coordinate

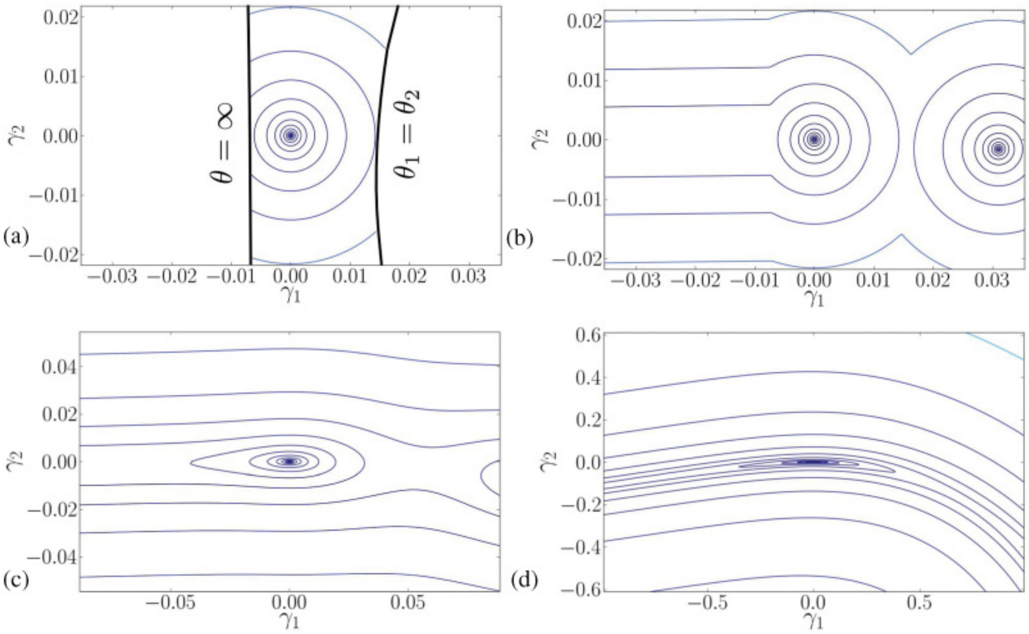

FIG. 12. (Color online) By changing the value of the Levenberg-Marquardt parameter, the course of the geodesics on the corresponding model graph are deformed, in turn distorting the shape of the cost contours in the geodesic coordinates. (a) $\lambda = 0$ is equivalent to the model manifold. The cost 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. 1(b), 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 [23].

change would be expressed as an extrinsic curvature on the expectation surface. Nonlinearity that does not produce an extrinsic curvature is not irrelevant; it can still have a strong influence on the model and can still limit the effectiveness of optimization algorithms. Specifically, this type of nonlinearity 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 nonlinearity as parameter-effect curvature [15,18]. 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 are 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 Sec. VI. A functional definition of parameter-effect curvature would be the nonlinearities that are annihilated by operating with P^N. Alternatively, one can think of the parameter-effect curvature as the curvatures of the coordinate mesh. We discuss parameter-effect curvature in Sec. VII C.

Bates and Watts refer to all nonlinearity that cannot be removed by changes of coordinates as intrinsic curvature [18]. We will not follow this convention; instead, we follow the differential geometry community and further distinguish between intrinsic or Riemann curvature (Sec. VII A) and extrinsic or embedding curvature [36] (Sec. VII B). 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 up 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.

A. 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 is the Gauss-Codazzi equation extended to the case of more than one independent normal direction [37].

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. 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-effect curvature, in contrast, cannot be simple tensors but will depend on a chosen direction. These curvatures are the key to understanding nonlinear least-squares fitting.

B. 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 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 rolling it into a cylinder. Rolling the paper does not affect distances on the surface, preserving its intrinsic properties, but it does change 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 [52]. While an extrinsic curvature does not always imply the existence of an intrinsic curvature, an intrinsic curvature requires that there also be extrinsic curvature. Our toy model, therefore, also exhibits extrinsic curvature as in Fig. 13. One model whose manifold is a ruled surface is given by a two-parameter model that varies an exponential rate and an amplitude:

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

The manifold for this model with three data points is drawn in Fig. 14.[2]

There are two measures of extrinsic curvature that we discuss. The first is known as geodesic curvature because 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-effect curvature that will allow us to compare the relative importance of the two types of curvature.

Measures of extrinsic and parameter-effect curvature to quantify nonlinearities have been proposed previously by Bates and Watts [15,17,18]. Although the measure they use is equivalent to the presentation of the next few sections, their

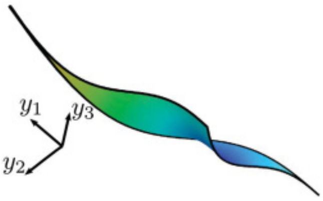

FIG. 13. (Color online) Intrinsic and extrinsic curvature. Intrinsic curvature is inherent to the manifold and cannot be removed by an alternative embedding. A model that is the sum of two exponential terms has all types of curvature. This is the same model manifold as in Fig. 1(c), viewed from an alternative 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 [23].

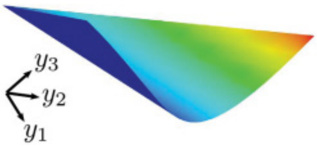

FIG. 14. (Color online) A ruled surface has no intrinsic curvature; however, it may have extrinsic curvature. The model manifold formed from a single exponential rate and amplitude is an example of a ruled surface. This model could be isometrically embedded in another space to remove the curvature. An animation of this surface rotating in three-dimensional space is available in the online supplemental material [23].

[2] 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 [67–69]. 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 submanifold that now depends on 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.

approach is different. The goal of this section is to express curvature measures of nonlinearity in a more standard way using the language of differential geometry. In so doing, we hope to make the results accessible to a larger audience.

1. *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 find a circle that locally 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, that is, an extrinsic curvature. The curvature associated with a geodesic path is illustrated in Fig. 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

$$K(v) = \frac{|v^\mu v^\nu P^N \partial_\mu \partial_\nu \vec{r}|}{v^\alpha v_\alpha}. \tag{27}$$

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

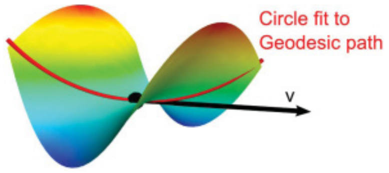

FIG. 15. (Color online) Geodesic curvature. A direction on a curved surface defines a geodesic. The deviation of the geodesic from a straight line in the embedding space is measured by the geodesic curvature. It is the inverse radius of the circle fit to the geodesic path at the point. A three-dimensional animation of this surface is available in the online supplemental material [23].

2. *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

$$S_{\mu\nu} = \hat{n} \cdot (\partial_\mu \partial_\nu \vec{r}). \tag{28}$$

$S_{\mu\nu}$ 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 in which there is only one direction normal to the surface, 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. 16.

In general, there will not be a 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 [36]. 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

$$S_{\mu\nu} = -\frac{\vec{r} P^N \partial_\mu \partial_\nu \vec{r}}{|P^N \vec{r}|}, \tag{29}$$

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

$$S(\vec{V})_{\mu\nu} = \frac{\vec{V} P^N \partial_\mu \partial_\nu \vec{r}}{|P^N \vec{V}|}. \tag{30}$$

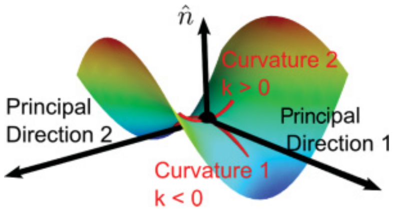

FIG. 16. (Color online) Shape operator. Specifying a direction normal to a curved surface, $\hat{n}$, defines a shape operator. The eigenvalues of the shape operator are the principal curvatures and the corresponding eigenvectors are the directions of principal curvature. A three-dimensional animation of this surface is available in the online supplemental material [23].

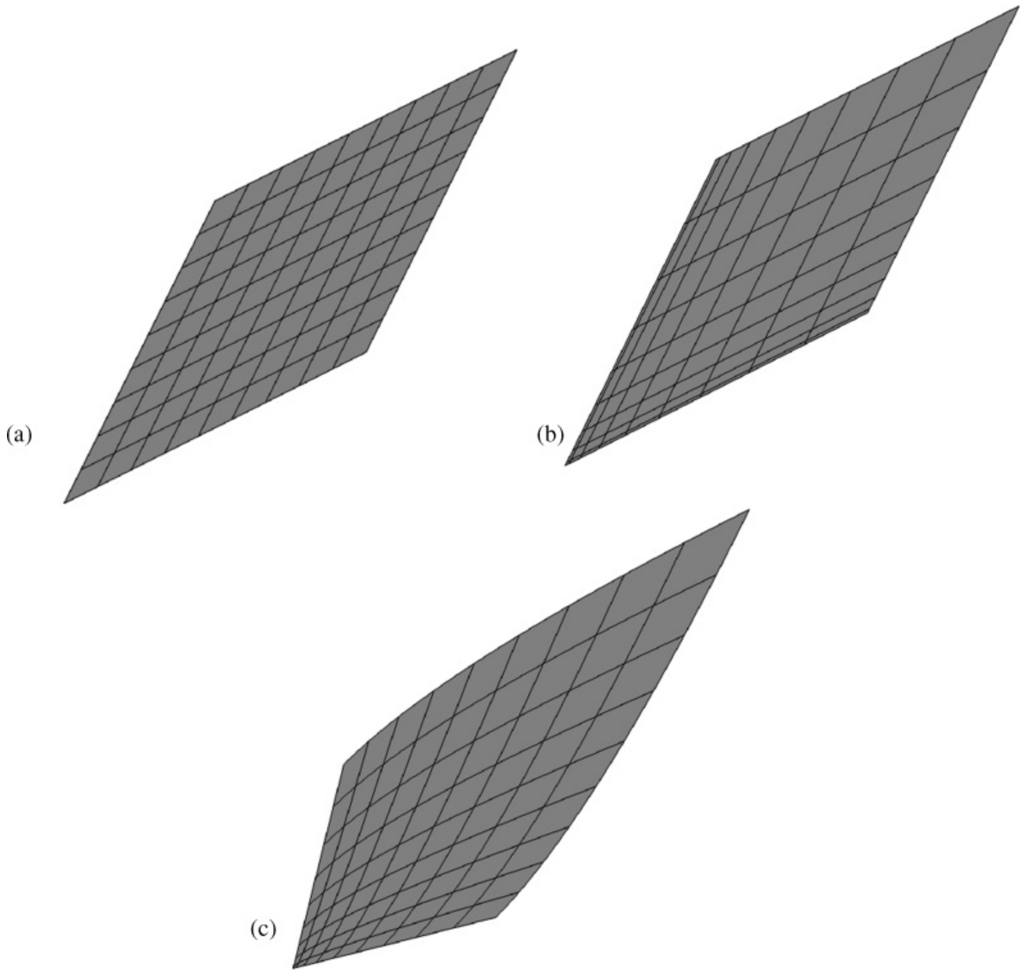

FIG. 17. (a) Linear grid. A sloppy linear model may have a skewed coordinate grid, but the shape of the grid is constant, having no parameter-effect curvature. (b) Compressed grid. By reparametrizing the model, the grid may become stretched or compressed in regions of the manifold. (c) Rotating, compressed grid. Another parametrization may not only stretch the grid, but also cause the coordinates to rotate. The parameter-effect 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-effect curvature (twisting).

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.

C. Parameter-effect curvature

We are now prepared to discuss parameter-effect curvature. We repeat that parameter-effect 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-effect 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 Sec. VI, we were able to remove the parameter-effect curvature from the model (at a point). In this section, we will discuss how to measure the parameter-effect curvature and compare it to the other curvatures that we discussed above.

To understand the meaning of parameter-effect 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

$$x = \epsilon\theta_1 + \theta_2, \quad y = \theta_1 + \epsilon\theta_2.$$

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. 17(a) for $\epsilon = 1/2$. By reparametrizing the linear model, we can introduce parameter-effect 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),

$$x = \epsilon\theta_1^2 + \theta_2^2, \quad y = \theta_1^2 + \epsilon\theta_2^2,$$

then the corresponding coordinate mesh will become compressed and stretched, as seen in Fig. 17(b). Alternatively, if we reparametrize the model as

$$x = (\epsilon\theta_1 + \theta_2)^2, \quad y = \left(\theta_1^2 + \epsilon\theta_2^2\right)^2,$$

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. 17(c). With more than two parameters, there is additionally a torsion parameter-effect curvature in which the lines twist around one another. None of these reparametrizations 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-effect curvature.

We now consider how to quantify parameter-effect 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-effect curvature can be found by replacing P^N with P^T in each expression. Thus, in analogy with Eq. (27), we can define the parameter-effect geodesic curvature by

$$K^P(v) = \frac{|v^\mu v^\nu P^T \partial_\mu \partial_\nu \vec{r}|}{v^\alpha v_\alpha}. \tag{31}$$

Likewise, we can define a parameter-effect shape operator by comparison with Eq. (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 nonlinearity) is of dimension $M - N$. The parameter-effect 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-effect 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-effect curvature is correctly interpreted as the connection coefficients. With this understanding, it is clear that geodesic coordinates remove parameter-effect 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.

D. 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 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 Sec. VI, by changing coordinates to extended geodesic coordinates, we discovered that the manifold generated by our sloppy model was surprisingly flat, that is, it had low intrinsic curvature. We have seen that if the model happens to have an 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 nonlinearities. This explains perhaps why we will find that the dominant curvature of sloppy models is the parameter-effect one.

We can better understand the size of the various curvatures by considering the interpretation presented in Sec. III 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. (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

$$K = \epsilon / W_j^2, \tag{32}$$

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 Sec. III following Eq. (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-effect curvature by recalling that the parameter-effect 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-effect curvature of the full model manifold in the stiffest direction now corresponds to the extrinsic curvature of this one-dimensional manifold,[3] and as such is set by the smallest width (which in this case is the only width), that is, the longest width of the full model manifold. The similar structure of parameter-effect curvature and extrinsic curvature, Eqs. (27) and (31), suggests that the parameter-effect curvature is also proportional to the inverse eigenvalues (squares of the widths) along several cross sections. Combining these results, we see that in general the ratio of extrinsic to parameter-effect curvature is given by the ratio of the widest to the most narrow width,

$$\frac{K}{K^P} \approx \frac{W_N}{W_1} \approx \sqrt{\frac{\lambda_N}{\lambda_1}}. \tag{33}$$

In our experience, the ratio of extrinsic to parameter-effect curvature in Eq. (33) is always very small. When Bates and Watts introduced parameter-effect curvature, they considered its magnitude on 24 models and found it to be universally larger than the extrinsic curvature, often much larger [15]. We have offered an explanation of this effect here based on the assumption that the deviation from flatness is given by Eq. (32).

We explicitly check the assumption of Eq. (32) by calculating cross sections for a model of several exponentials and for an artificial neural network. We have already seen in Sec. III in Fig. 6 that these widths span several orders of magnitude, as predicted by the singular values of the Jacobian. In Fig. 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 deviation from flatness is similar for all the cross sections. In Fig. 19, we see that the extrinsic curvature is comparable to the narrowest cross section and the parameter-effect 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. Figure 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-effect 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-effect 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. (33). We will use this large discrepancy between extrinsic and parameter-effect curvature when we improve the standard algorithms in Sec. VIII.

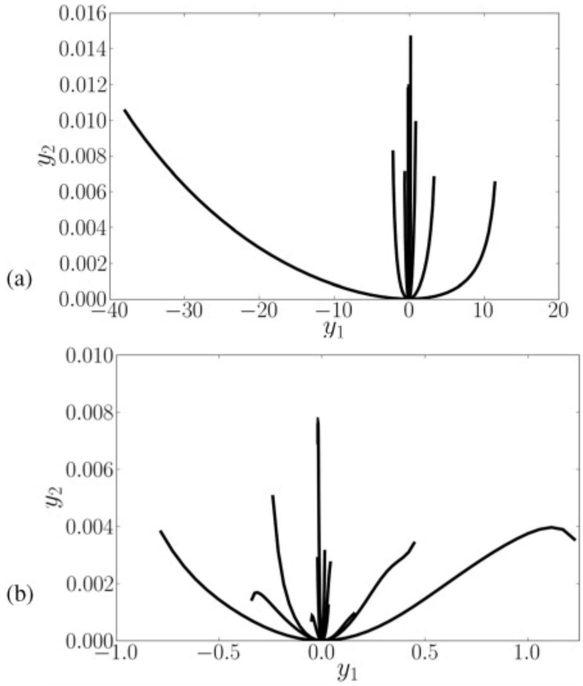

FIG. 18. (a) Cross sections of a summing exponential model projected into the plane spanned by the two principal components in data space. Notice the widths of successive cross sections are progressively more narrow, while the deviations from flatness are uniformly spread across the width. The magnitude 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 1.

We have seen that manifolds of sloppy models possess a number of universal characteristics. We saw in Sec. III 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-effect curvatures also possess a universal structure summarized in Figs. 18–21. A remarkable thing about the parameter-invariant, global structure of a sloppy model manifold is that it is typically well-described by the singular values of the parameter-dependent, local Jacobian matrix. We saw in Sec. III 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-effect and extrinsic curvature, respectively. This entire structure is a consequence of the observation that most models are a multidimensional interpolation scheme.

Let us summarize our conclusions about the geometry of sloppy models. We argued in Sec. III using interpolation theorems that multiparameter nonlinear least-squares models should have model manifolds with a hierarchy of widths, forming a hyper-ribbon with the nth 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. 6). This

[3] This is strictly only true if the parameter-effect curvature has no compression component. Bates and Watts observe that, typically, the compression is a large part of the parameter-effect curvature [15]. As long as the compression is not significantly larger than the rotation (i.e., is within an order of magnitude), the parameter-effect curvature will be the same order of magnitude as the extrinsic curvature of the one-dimensional model.

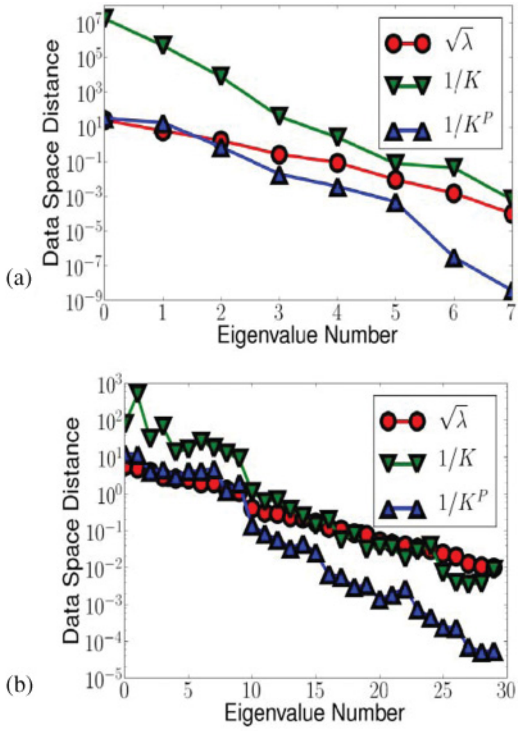

FIG. 19. (Color online) The extrinsic and parameter-effect curvature on the model manifold are strongly anisotropic, with the largest curvatures along the shortest widths (see Figs. 6 and 18). The slopes of the (inverse) curvature vs eigenvalue lines are roughly twice that of the singular values (which 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-effect 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 Ref. [22] and a feed forward artificial neural network (below).

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 17 systems biology models [6]. 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. 18(c)]. We find this hierarchy both measured along the eigenvectors of the (parameter-independent) shape operator (Fig. 20) or the geodesic curvatures measured along the (parameter-dependent) eigenpredictions at the best fit. Finally, we note that the parameter-effect curvature also scales as $1/\Delta^{2n}$ by inspecting the similarity in the two formulas, Eqs. (27) and (31). We argue that the parameter-effect 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-effect shape operator or along eigenpredictions. Thus the entire structure of the manifold can be summarized

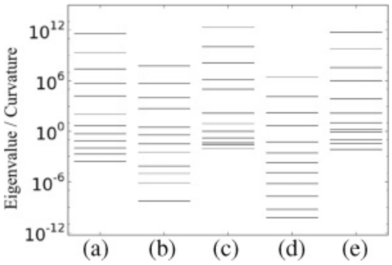

FIG. 20. Curvature anisotropy. (a) Inverse metric eigenvalues. The (inverse) metric has eigenvalues spread over several orders of magnitude, producing a strong anisotropy in the way distances are measured on the model manifold. (b) Geodesic curvature in eigendirections of the metric. The geodesic curvatures also cover many decades. The shortened distance measurements from the metric eigenvalues magnify the anharmonicities in the sloppy directions. (c) Parameter-effect geodesic curvature. The parameter-effect 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-effect shape operator eigenvalues. Two measures (geodesic and shape operator curvatures) span similar ranges, but in both cases the parameter-effect curvature is a factor of about 10^5 larger than the extrinsic curvature equivalent.

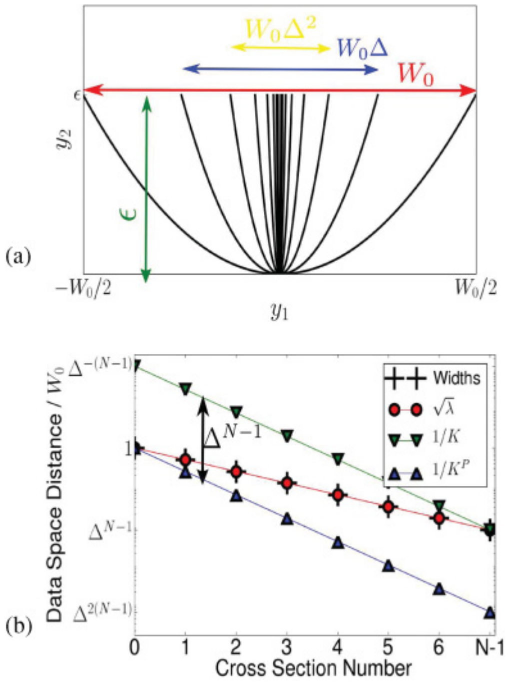

FIG. 21. (Color online) A caricature of the widths and curvatures of a typical sloppy model. (a) The manifold deviates by an amount Δ^N from a linear model for each width. As each width is smaller than the last by a factor of Δ, the curvature is largest along the narrow widths. This summary agrees well with the two real models in Fig. 18. (b) The scales of the extrinsic and parameter-effect curvature are set by the narrowest and widest widths, respectively. The parameter-effect curvature is, therefore, smaller than the extrinsic curvature by a factor of Δ^N. Both are strongly anisotropic. Compare this figure to the corresponding result for the two real models in Fig. 19.

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. 21.

E. Curvature on the model graph

Most of the nonlinearities of sloppy models appear as parameter-effect curvature on the model manifold. On the model graph, however, these nonlinearities become extrinsic curvature because the model graph emphasizes the parameter dependence. An extreme version of this effect can be seen explicitly in Fig. 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. 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. 22. As the Levenberg-Marquardt parameter is raised, the widely ranging values of the geodesic curvatures may either increase or decrease. The largest curvature directions (the sloppy directions) tend to flatten, but the directions with the lowest curvature (the stiff directions) become more curved. The main effect of 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-effect 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 effect becomes 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.

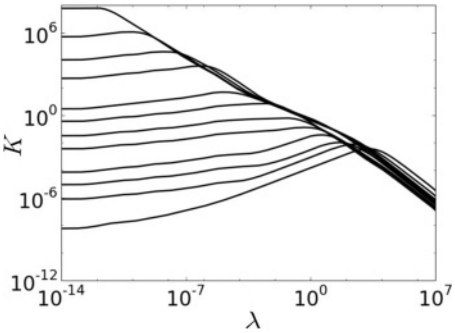

FIG. 22. Model graph curvature. As the Levenberg-Marquardt parameter λ is increased, directions with highest curvature become less curved. For stiff directions with less extrinsic curvature, the parameter-effect curvature may be transformed into extrinsic curvature. The damping term reduces the large anisotropy in the curvature. For sufficiently large values of the Levenberg-Marquardt parameters, all curvatures vanish.

The largest curvatures typically correspond to the sloppy directions. Most algorithms will try to step in sloppy directions 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.

F. Optimization curvature

The distinction between extrinsic and parameter-effect 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 nonlinearities, both extrinsic and parameter effects. We would like a measure of nonlinearity, 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 nonlinearity 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 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 does not 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,

$$\kappa(v) = -\frac{v^\mu v^\nu \partial_\mu \partial_\nu \vec{r}}{v^\alpha v_\alpha} \cdot \frac{\vec{r}}{|\vec{r}|}, \qquad (34)$$

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-effect curvature into a single measure of nonlinearity, although in practice it is dominated by the parameter-effect curvature. We now consider how κ is related to the allowed step size of an iterative algorithm.

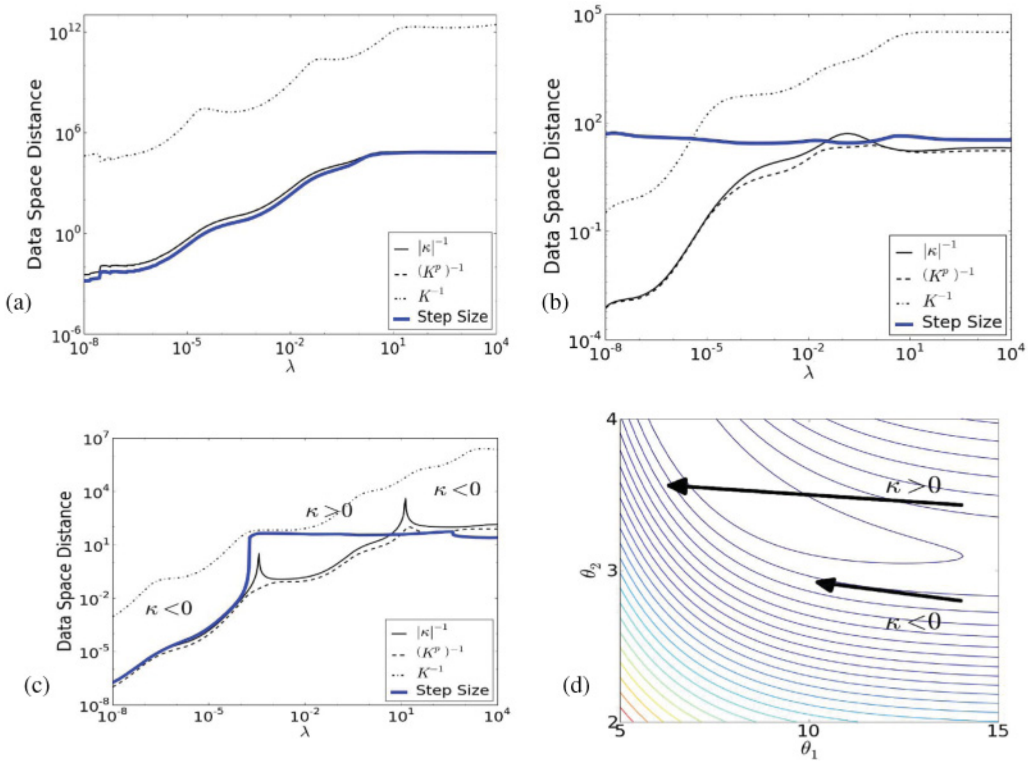

FIG. 23. (Color online) (a) Curvature and step size for $\kappa < 0$. If $\kappa < 0$, then the nonlinearities in the proposed direction are diverting the algorithm away from the desired path. Distances are limited by the size of the curvature. (b) Curvature and step size for $\kappa > 0$. The nonlinearities 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 nonlinearities 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.

Consider the scaled Levenberg step given by

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

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 Sec. VIII C; however, this simple criterion is useful for illustrating the limitations of 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), the parameter-effect curvature (K^P), and an optimization curvature (κ), each of which is measured in units of inverse distance in data space. We compare the step size with the inverse curvature in each direction in Fig. 23.

One might assume that the size of the nonlinearities 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. 23(a), where $\kappa < 0$; the step size closely follows the largest of the curvatures, the parameter-effect curvature $K^P \approx |\kappa|$.

However, the nonlinearities on occasion may inadvertently be helpful to an algorithm, as in Fig. 23(b) 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. 23(d). 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 the step runs down the canyon and eventually up the opposite wall, resulting in a much larger step size.

G. 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. To estimate the expected number of evaporated parameters, however, it is necessary to account for the extrinsic curvature of a model.

In Fig. 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

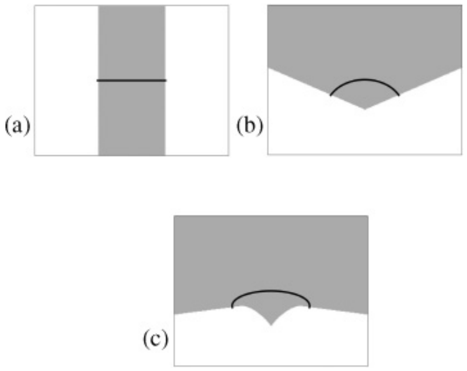

FIG. 24. The curvature along the width of a manifold effects if the best fit lies on the boundary or on the interior. For a cross-sectional width (thick black line), consider three possibilities: (a) extrinsically flat, (b) constant curvature along width, and (c) curvature proportional to distance from the boundary. Gray 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.

a cross-sectional width with no extrinsic curvature, as in Fig. 24(a). 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. Figures 24(b) and 24(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 nonevaporated parameters and the relative probabilities of the possible data [Eq. (2)], we can estimate the expected number of evaporated parameters for a given 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 of zero eigenvalues of the metric, corresponding to the number of nonevaporated parameters at the fit.

We can derive analytic estimates for the number of evaporated parameters using the approximation that the cross section is either flat or has constant curvature, as in Figs. 24(a) and 24(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

$$P_n^{\text{flat}} = 2\,\text{erf}\left(\frac{W_n}{2\sigma}\right), \tag{35}$$

where W_n is the nth 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. 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 1. 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 $W_n K_n$. Recall that we characterize a sloppy model manifold by three numbers, W_0, Δ, and N, which are 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

$$P_n^{\text{curved}} = \begin{cases} 1 & \text{if } \sigma < W_n, \\ \frac{W_0\Delta^n}{\sigma} & \text{if } W_n < \sigma < 1/K_n, \\ \Delta^{N-n} & \text{if } 1/K_n < \sigma. \end{cases} \tag{36}$$

From our caricature of a typical sloppy model summarized in Fig. 21, we estimate how many widths should belong in each category for a given σ. Summing the probabilities for the several widths in Eq. (36), we find the expected number of nonevaporated parameters to be given by

$$\langle N_{\text{approx}}\rangle = \frac{2}{1-\Delta} + \frac{\log\sigma/W_0}{\log\Delta} - 1. \tag{37}$$

In Table I, we compare the fraction of nonevaporated parameters with the estimates from Eqs. (35) and (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 nonevaporated 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. (27). We explicitly

TABLE I. The number of nonevaporated parameters $\langle N\rangle$ per total number of parameters N at the best fit, for an eight-parameter model of exponentials and amplitudes. As the noise of the data ensemble grows, the number of nonevaporated 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 nonevaporated parameters, due to the curvature (see Fig. 24). We estimate the expected number of nonevaporated parameters from both a flat manifold approximation [Eq. (35)] and a constant curvature approximation. For the constant curvature approximation, we show the result of the exact integral of the Gaussian over the gray region of Fig. 24(b) as well as our perturbative approximation, Eq. (37), using the parameters $W_0 = 6.1$, $\Delta = 0.11$, and $N = 8$. These approximations agree with the numerical results when the noise is small, but for very noisy data there are still several nonevaporated parameters even if the noise is much larger than any of the widths. Therefore, although our general caricature of the model manifold as a hypercylinder 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, to an increase in the curvature near the boundary, as in Fig. 24(c).

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	0.0006	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

calculate the curvature near the boundary and we find that this is in fact the case.

The calculation in Table I 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-effect curvatures, the parameter uncertainties estimated by the inverse Fisher information matrix will be accurate since the parametrization is constant over the range of uncertainty. It is important to note that for models with large numbers of parameters, either of these conditions requires extremely small, often unrealistically small, error bars. In general, it is more practical to focus on predictions made by ensembles of parameters with good fits rather than parameter values at the best fit as the latter will depend strongly on the noise in the data.

VIII. APPLICATIONS TO ALGORITHMS

We now consider how the results derived in previous sections can be applied to algorithms. We have stressed that fitting sloppy models to data consists 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.

TABLE II. The results of several algorithms applied to a test problem of fitting a sum of four exponential terms (varying both rates and amplitudes—eight 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 indicates how efficiently it can follow the canyon to the best fit. BFGS is a quasi-Newton scalar minimizer of Broyden, Fletcher, Goldfarb, and Shanno [62,63]. The traditional [34,44] and trust region [45] 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.

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

We begin by deriving two common algorithms, the modified Gauss-Newton method and the Levenberg-Marquardt algorithm from the geometric picture in Secs. VIII A and VIII B. We then suggest how they may be improved by applying what we call delayed gratification and an acceleration term in Secs. VIII C and VIII D.

We demonstrate that the suggested modifications can offer improvements to the algorithms by applying them to a few test problems in Sec. VIII E. In comparing the effectiveness of the 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 II 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 (Sec. VIII A) and other similar algorithms, such as the Levenberg-Marquardt algorithm (Sec. VIII B). 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 Sec. VIII C. Second, we suggest the inclusion of a geodesic acceleration term in Sec. VIII D. We end the discussion by comparing the efficiency of standard versions of algorithms to those with the suggested improvements in Sec. VIII E.

A. 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 Sec. VI. The coordinates illustrated in Fig. 11 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

$$\frac{d\theta^\mu}{d\tau}(\tau = 0) = -g^{\mu\nu}\partial_\nu C. \tag{38}$$

If we assume that the manifold is extrinsically flat (the necessary and sufficient condition to produce concentric circles in extended geodesic coordinates), then Eq. (26) tells us that the cost will be purely quadratic,

$$\frac{d^2C}{d\tau^2} = g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau} = \text{const}, \tag{39}$$

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

$$\frac{dC}{d\tau} = \left(g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}\right)\tau + \dot{C}(\tau = 0). \tag{40}$$

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

$$\tau_{\max} = -\frac{\dot{C}(\tau = 0)}{g^{\mu\nu}\frac{d\theta^\mu}{d\tau}\frac{d\theta^\nu}{d\tau}}. \tag{41}$$

We can calculate the missing piece of Eq. (41) from the chain rule and Eq. (38),

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

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

$$\delta\theta^\mu = \dot{\theta}^\mu\delta\tau = -g^{\mu\nu}\partial_\nu C, \tag{42}$$

since $\delta\tau = 1$. Iteratively updating the parameters according to Eq. (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. (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 [53].

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 to be competitive with other algorithms. In practice, these techniques are not as effective as the Levenberg-Marquardt algorithm, discussed in the next section.

B. Levenberg-Marquardt algorithm

The algorithm that steps according to Eq. (42) using the metric of the model graph, Eq. (17), is known as the Levenberg-Marquardt step:

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

If D is chosen to be the identity, then the algorithm is the Levenberg algorithm [43]. 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 [44]. As we mentioned in Sec. IV, 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 Sec. IV, 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 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 Ref. [34]. One picks an initial value, say $\lambda = 0.001$, 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 [45]. 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,

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

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 [54] and note that it was a key inspiration for the delayed gratification update scheme that we describe below in Sec. VIII C.

As we observed in Sec. V, 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 [55].

C. Delayed gratification

We have seen that parameter-effect 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 nonlinearities alone suggest and produce the concentric circles in Fig. 11. When considering only a single step on even a highly curved manifold, if the parameter-effect 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

$$r_1 = 1 - \theta_1, \quad r_2 = A\big(\theta_2 - \theta_1^2\big),$$

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

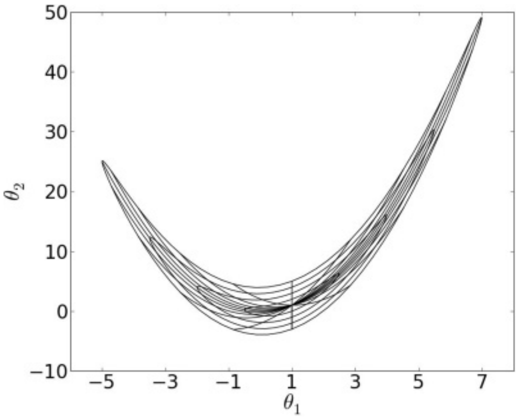

FIG. 25. Extended geodesic coordinates for the Rosenbrock function. The residuals are one choice of extended geodesic coordinates if the number of parameters equals the number of data points, as is the case for the Rosenbrock function. Because the Rosenbrock function is a simple quadratic, the coordinate transformation can be expressed analytically. Lines of constant ρ are equicost 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 equicost 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.

$$\theta_1 = 1 - r_1, \quad \theta_2 = \frac{r_2}{A} + (1 - r_1)^2.$$

If we change to polar coordinates,

$$r_1 = \rho \sin\phi, \quad r_2 = \rho\cos\phi,$$

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. 25.

Inspecting the geodesic paths that lead to the best fit in Fig. 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 Sec. VII F 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 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 equicost 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 [56]. 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 at successive iterations, slowing down the convergence rate, as illustrated in Fig. 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 because of how this strategy reduces the possibility of future steps.

D. 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 [57,58], to neural network training [59], and to the general problem of optimization on manifolds [33,60]. Here we apply it as a second-order correction to the Levenberg-Marquardt step.

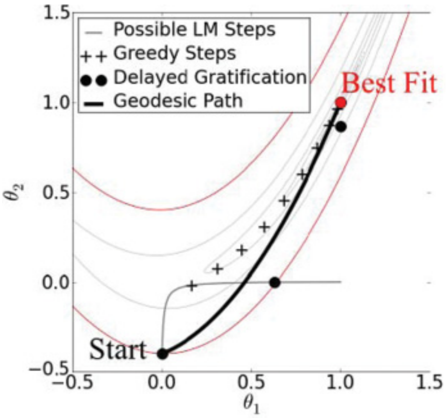

FIG. 26. (Color online) Greedy step and delayed gratification step criterion. In optimization problems for which there is a long narrow canyon, such as for the Rosenbrock function, choosing a delayed gratification step is important to optimize convergence. By varying the damping term, the algorithm may choose from several possible steps. A greedy step will lower the cost as much as possible, but by doing so it 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.

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 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

$$a^\mu = -\Gamma^\mu_{\alpha\beta} v^\alpha v^\beta, \tag{43}$$

as one can see by inspecting Eq. (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

$$\delta\theta^\mu = \dot{\theta}^\mu \delta\tau + \tfrac{1}{2}\ddot{\theta}^\mu \delta\tau^2. \tag{44}$$

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. 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. 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 antiparallel. 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 the velocity, as for the summing exponentials model in Fig. 28(a).

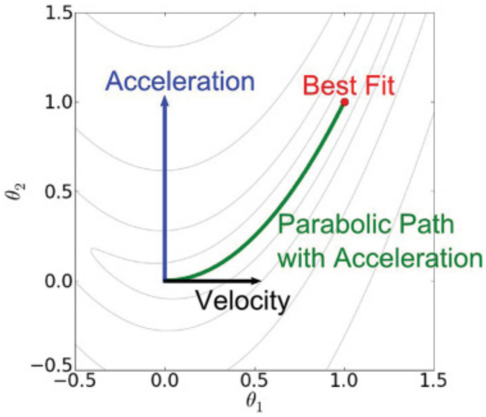

FIG. 27. (Color online) Geodesic acceleration in the Rosenbrock Valley. The Gauss-Newton direction, or velocity vector, gives the correct direction that one should move to approach the best fit while navigating a canyon. However, that direction quickly rotates, requiring an algorithm to take very small steps 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.

Although an acceleration that is antiparallel 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. 28(b).

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 Levenberg-Marquardt parameter until

$$\frac{\sqrt{\sum (a^\mu)^2}}{\sqrt{\sum (v^\mu)^2}} < \alpha, \tag{45}$$

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-effect curvature is 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

$$v^\mu = \frac{-g^{\mu\nu}\nabla_\nu C}{\sqrt{g^{\alpha\beta}\nabla_\alpha C\, \nabla_\beta C}}, \tag{46}$$

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. (46)

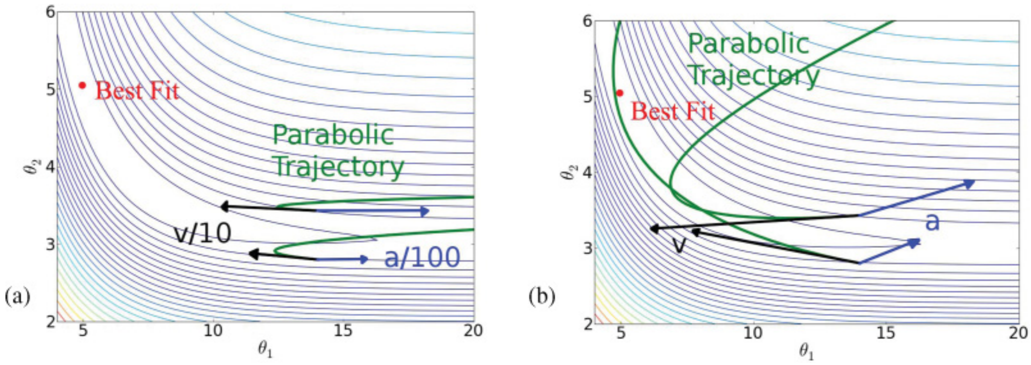

FIG. 28. (Color online) (a) Deacceleration when overstepping. Typically the velocity vector greatly overestimates the proper step size. (We have rescaled both velocity and acceleration to fit in the figure.) Algebraically, this is due to the factor of the inverse metric in the velocity, which has very large eigenvalues. The acceleration compensates for this by pointing antiparallel 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.

than by the geodesic equation, Eq. (24). In fact, Eq. (46) has been proposed as a neural network training algorithm by Amari *et al.* [42].

The second-order differential equation corresponding to Eq. (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 nonlinearities are parameter-effect curvature, the resulting differential equation is exactly the geodesic equation. By comparing step sizes with inverse curvatures in Fig. 23, we can see that over a distance of several steps, the approximation that all nonlinearities are parameter-effect curvature should be very good. In such a case, the deviation of Eq. (46) from Eq. (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. (5) with the full Hessian in Eq. (4). Many standard algorithms seek to efficiently find the actual Hessian, either by calculating it directly or by estimation [34,61]. One such algorithm, which we use for comparison in the next section, is a quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shannon (BFGS) [62], 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 Sec. VI 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 nonlinearities, 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.

E. 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,

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

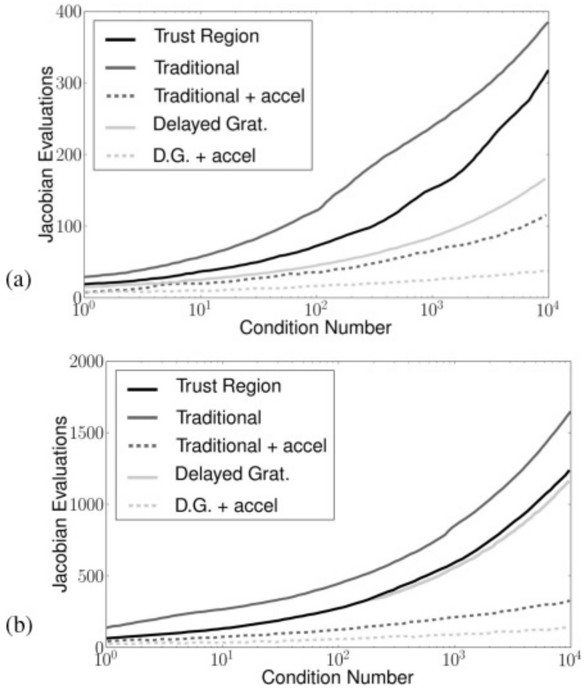

FIG. 29. Generalized Rosenbrock results for Levenberg-Marquardt variants. If the canyon that an algorithm must follow is very narrow (measured by the condition number of the metric at the best fit) or turns sharply, the algorithm will require more steps to arrive at the best fit. Those that use the geodesic acceleration term converge more quickly as the canyon narrows. As the parameter-effect 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.

where A and n are not optimizable parameters but set to control the difficulty of 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 that an algorithm requires to decrease the cost to 10^{-4}. The results for the cubic and quartic versions of the problem are given in Fig. 29 for several versions of the Levenberg-Marquardt algorithm.

We next consider a summing exponential problem; a summary of these results can be found in [22]. Here we expand it to include the delayed gratification algorithm outlined above in Sec. VIII C.

A surprising result from Table II 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. (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, the results of which will appear shortly [54].

IX. 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-effect 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-effect 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 [54].

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 we acknowledge support from NSF Grants No. DMR-0705167 and No. DMR-1005479.

APPENDIX A: 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,

$$I_{\mu\nu} = \left\langle -\frac{\partial^2 l}{\partial\theta^\mu \partial\theta^\nu}\right\rangle = -\int d\xi\;\, p(\xi,\theta)\frac{\partial^2 l}{\partial\theta^\mu \partial\theta^\nu}. \quad \text{(A1)}$$

It can be shown that the Fisher information can be written entirely in terms of first derivatives:

$$I_{\mu\nu} = \left\langle \frac{\partial l}{\partial\theta^\mu}\frac{\partial l}{\partial\theta^\nu}\right\rangle = \int d\xi\;\, p(\xi,\theta)\frac{\partial l}{\partial\theta^\mu}\frac{\partial l}{\partial\theta^\nu}. \quad \text{(A2)}$$

Equation (A2) makes it clear that the Fisher information is a symmetric, positive-definite matrix that 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 Sec. I, 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. (A1) or Eq. (A2) to calculate the information metric for least-squares problems. From Eq. (A1), we get

$$g_{\mu\nu} = \left\langle \frac{\partial^2 C}{\partial\theta^\mu \partial\theta^\nu}\right\rangle = \sum_m \langle \partial_\mu r_m \partial_\nu r_m + r_m \partial_\mu \partial_\nu r_m\rangle, \quad \text{(A3)}$$

where we have replaced I by g to indicate that we are now interpreting it as a metric.

Equation (A3), 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. (1) reveals that the derivatives of the residuals have no random element, and can therefore be treated as constant. The net result is

$$g_{\mu\nu} = \sum_m \partial_\mu r_m \partial_\nu r_m = (J^T J)_{\mu\nu}, \tag{A4}$$

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. (A2), 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. (A4), the same metric found in Sec. I.

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 [12]. They are given by the formula

$$\Gamma^{(\alpha)}_{\mu\nu,\epsilon} = \left\langle \partial_\epsilon l \partial_\mu \partial_\nu l + \left(\frac{1-\alpha}{2} \right) \partial_\epsilon l \partial_\mu l \partial_\nu l \right\rangle.$$

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

$$C = \sum_m |r_m|^p, \tag{A5}$$

which is naturally interpreted as the pth 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 $p = \infty$ [34]. 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

$$G_{mn} \propto \Delta\, d\vec{r}\, e^{-C} \frac{\partial C}{\partial r_m} \frac{\partial C}{\partial r_n}$$

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 function 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. (A5), corresponding to the L^p norm, $G_{mn} \propto \delta_{mn}$, so the metric of the 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 of 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 [12,13].

APPENDIX B: 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 [64].

Algorithm 1 Traditional Levenberg-Marquardt algorithm as described in [34,43,44]

```
1. Initialize values for the parameters, $x$, the Levenberg-Marquardt parameter $\lambda$, as well as $\lambda_{\text{up}}$ and $\lambda_{\text{down}}$ 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 $r_{\text{new}}$ at the point given by $x_{\text{new}} = x - g^{-1} \nabla C$, and calculate the cost at the new point, $C_{\text{new}} = \frac{1}{2} r_{\text{new}}^2$.
4. If $C_{\text{new}} < C$, accept the step, $x = x_{\text{new}}$, and set $r = r_{\text{new}}$ and $\lambda = \lambda / \lambda_{\text{down}}$. Otherwise, reject the step, keep the old parameter guess $x$ and the old residuals $r$, and adjust $\lambda = \lambda \times \lambda_{\text{up}}$.
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.
```

Algorithm 2 Geodesic acceleration in the traditional Levenberg-Marquardt algorithm

```
1. Initialize values for the parameters, $x$, the Levenberg-Marquardt parameter $\lambda$, as well as $\lambda_{\text{up}}$ and $\lambda_{\text{down}}$ 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, $v = -g^{-1} \nabla C$, and the geodesic acceleration of the residuals in the direction of the velocity, $a = -g^{-1} J^T (v^\mu v^\nu \partial_\mu \partial_\nu r)$.
4. Evaluate the new residuals $r_{\text{new}}$ at the point given by $x_{\text{new}} = x + v + \frac{1}{2} a$, and calculate the cost at the new point, $C_{\text{new}} = \frac{1}{2} r_{\text{new}}^2$.
5. If $C_{\text{new}} < C$ and $|a|/|v| < \alpha$, accept the step, $x = x_{\text{new}}$, and set $r = r_{\text{new}}$ and $\lambda = \lambda/\lambda_{\text{down}}$. Otherwise, reject the step, keep the old parameter guess $x$ and the old residuals $r$, and adjust $\lambda = \lambda \times \lambda_{\text{up}}$.
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.
```

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 Ref. [34] 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 Sec. VIII C modifies the traditional Levenberg-Marquardt algorithm to raise and lower the Levenberg-Marquardt term by differing amounts. The goal is to accept a step with the smallest value of the damping term that will produce a downhill step. This can typically be accomplished by choosing $\lambda_{\text{up}} = 2$ and $\lambda_{\text{down}} = 10$.

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 [54].

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 [63]. 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 BFGS, described in [62]. We also employ a limited memory variation (L-BFGS-B) described in [65] and a conjugate gradient (CG) method of Polak and Ribiere, also described in [62]. We also explored the downhill simplex algorithm of Nelder and Mead and a modification of Powells' method [63], neither of which make use of gradient information directly, and were not competitive with other algorithms.

[1] K. S. Brown and J. P. Sethna, Phys. Rev. E **68**, 021904 (2003).

[2] K. Brown, C. Hill, G. Calero, C. Myers, K. Lee, J. Sethna, and R. Cerione, Phys. Biol. **1**, 184 (2004).

[3] F. Casey, D. Baird, Q. Feng, R. Gutenkunst, J. Waterfall, C. Myers, K. Brown, R. Cerione, and J. Sethna, Syst. Biol. IET **1**, 190 (2007).

[4] B. Daniels, Y. Chen, J. Sethna, R. Gutenkunst, and C. Myers, Curr. Opin. Biotechnol. **19**, 389 (2008).

[5] R. Gutenkunst, F. Casey, J. Waterfall, C. Myers, and J. Sethna, Ann. N.Y. Acad. Sci. **1115**, 203 (2007).

[6] R. Gutenkunst, J. Waterfall, F. Casey, K. Brown, C. Myers, and J. Sethna, PLoS Comput. Biol. **3**, e189 (2007).

[7] R. Gutenkunst, Ph.D. thesis, Cornell University, 2008.

[8] J. J. Waterfall, F. P. Casey, R. N. Gutenkunst, K. S. Brown, C. R. Myers, P. W. Brouwer, V. Elser, and J. P. Sethna, Phys. Rev. Lett. **97**, 150601 (2006).

[9] H. Jeffreys, *Theory of Probability* (Oxford University Press, New York, NY, 1998).

[10] C. Rao, Bull. Calcutta Math. Soc. **37**, 81 (1945).

[11] C. Rao, Sankhya **9**, 246 (1949).

[12] S. Amari and H. Nagaoka, *Methods of Information Geometry* (American Mathematical Society, Providence, Rhode Island, 2007).

[13] M. Murray and J. Rice, *Differential Geometry and Statistics* (Chapman & Hall, New York, 1993).

[14] E. Beale, J. R. Stat. Soc. **22**, 41 (1960).

[15] D. Bates and D. Watts, J. R. Stat. Soc. **42**, 1 (1980).

[16] D. Bates and D. Watts, Ann. Stat. **9**, 1152 (1981).

[17] D. Bates, D. Hamilton, and D. Watts, Commun. Statist.-Simul. Comput. **12**, 469 (1983).

[18] D. Bates and D. Watts, *Nonlinear Regression Analysis and Its Applications* (Wiley, New York, NY, 1988).

[19] R. Cook and J. Witmer, Am. Stat. Assoc. **80**, 872 (1985).

[20] R. Cook and M. Goldberg, Ann. Stat. **14**, 1399 (1986).

[21] G. Clarke, J. Am. Stat. Assoc. **82**, 844 (1987).

[22] M. K. Transtrum, B. B. Machta, and J. P. Sethna, Phys. Rev. Lett. **104**, 060201 (2010).

[23] See supplemental material at [http://link.aps.org/supplemental/10.1103/PhysRevE.83.036701] for an animation of this figure.

[24] O. Barndorff-Nielsen, D. Cox, and N. Reid, Int. Stat. Rev. **54**, 83 (1986).
[25] D. Gabay, J. Optim. Theory Appl. **37**, 177 (1982).
[26] R. Mahony, Ph.D. thesis, Australian National University, 1994.
[27] R. Mahony and J. Manton, J. Global Optim. **23**, 309 (2002).
[28] R. 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).
[29] S. Smith, Ph.D. thesis, Harvard University, Cambridge, MA, 1993.
[30] S. Smith, Hamiltonian Gradient Flows: Algorithms Control **3**, 113 (1994).
[31] C. Udriste, *Convex Functions and Optimization Methods on Riemannian Manifolds* (Kluwer, Dordrecht, 1994).
[32] Y. Yang, J. Optim. Theory Appl. **132**, 245 (2007).
[33] P. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds* (Princeton University Press, Princeton, NJ, 2008).
[34] W. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing* (Cambridge University Press, New York, NY, 2007).
[35] C. Misner, K. Thorne, and J. Wheeler, *Gravitation* (Freeman, San Francisco, 1973).
[36] M. Spivak, *Differential Geometry* (Publish or Perish, Berkeley, CA, 1979).
[37] L. Eisenhart, *Riemannian Geometry* (Princeton University Press, Princeton, NJ, 1997).
[38] T. Ivancevic, *Applied Differential Geometry: A Modern Introduction* (World Scientific, Singapore, 2007).
[39] J. Stoer, R. Bulirsch, W. Gautschi, and C. Witzgall, *Introduction to Numerical Analysis* (Springer Verlag, Berlin, 2002).
[40] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation* (Westview, Redwood City, California, 1991).
[41] S. L. Frederiksen, K. W. Jacobsen, K. S. Brown, and J. P. Sethna, Phys. Rev. Lett. **93**, 165501 (2004).
[42] S. Amari, H. Park, and T. Ozeki, Neural Comput. **18**, 1007 (2006).
[43] K. Levenberg, Q. Appl. Math. **2**, 164 (1944).
[44] D. Marquardt, J. Soc. Ind. Appl. Math. **11**, 431 (1963).
[45] J. More, Lect. Notes Math. **630**, 105 (1977).
[46] R. Kass, J. R. Stat. Soc. Ser. B (Methodol.) **46**, 86 (1984).
[47] D. Hamilton, D. Watts, and D. Bates, Ann. Stat. **10**, 393 (1982).
[48] J. Donaldson and R. Schnabel, Technometrics **29**, 67 (1987).
[49] B. Wei, Aust. New Zeal. J. Stat. **36**, 327 (1994).
[50] L. Haines, T. O. Brien, and G. Clarke, Stat. Sin. **14**, 547 (2004).
[51] E. Demidenko, Comput. Stat. Data Anal. **51**, 1739 (2006).
[52] D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination* (American Mathematical Society, New York, NY, 1999).
[53] H. Hartley, Technometrics **3**, 269 (1961).
[54] M. K. Transtrum, B. B. Machta, C. Umrigar, P. Nightingale, and J. P. Sethna (unpublished).
[55] A. Bakushinskii, Comput. Math. Math. Phys. **32**, 1353 (1992).
[56] M. Lampton, Comput. Phys. **11**, 110 (1997).
[57] D. Luenberger, Manag. Sci. **18**, 620 (1972).
[58] A. Pázman, J. Stat. Planning Infer. **103**, 401 (2002).
[59] C. Igel, M. Toussaint, and W. Weishui, *Trends and Applications in Constructive Approximation*, International Series of Numerical Mathematics, Vol. 151 (Birkhauser, Basel, Switzerland, 2005).
[60] Y. Nishimori and S. Akaho, Neurocomputing **67**, 106(2005).
[61] P. Gill and W. Murray, SIAM J. Numer. Anal. **15** 977 (1978).
[62] J. Nocedal and S. Wright, *Numerical Optimization* (Springer, New York, NY, 1999).
[63] E. Jones, T. Oliphant, and P. Peterson *et al.*, [http://www.scipy.org] (2001).
[64] J. Moré, B. Garbow, and K. Hillstrom, *User Guide for MINPACK-1* (Argonne National Laboratory, Argonne, Illinois, 1980).
[65] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, SIAM J. Sci. Comput. **16**, 1190 (1995).
[66] K. S. Brown, Ph.D. thesis, Cornell University, 2003.
[67] G. Golub and V. Pereyra, SIAM J. Numer. Anal. **10**, 413 (1973).
[68] L. Kaufman, BIT Numer. Math. **15**, 49 (1975).
[69] G. Golub and V. Pereyra, Inverse Probl. **19**, R1 (2003).