Case Study V: Exploring the χ^2 Landscape

Adam Stone
Department of Materials Science and Engineering, Lehigh University, Bethlehem, PA 18015, USA
(Dated: July 20, 2011)

The χ^2 landscape was plotted for a model function with multiple minima in χ^2, and the behavior of different fitting algorithms was examined. Starting guess values were important in determining which minimum would be found by the fitting algorithm, or whether it would get stuck without finding one. The Levenberg-Marquardt Method, Gauss-Newton Method, and Steepest Descent method were all calculated manually in order to observe the path the algorithm takes through parameter space in minimizing χ^2.

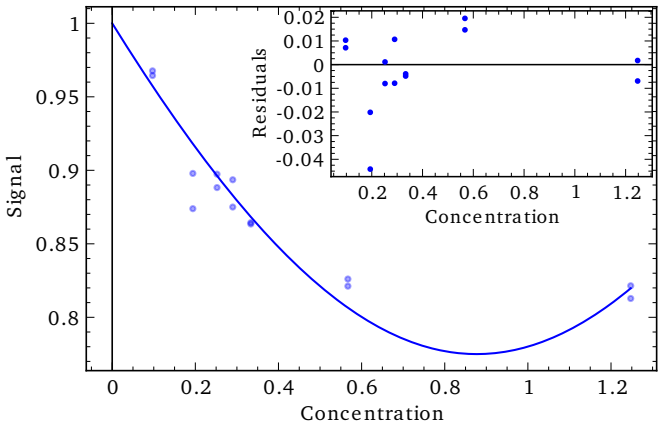

FIG. 1. Initial fit with residuals inset.

Data was provided which is known to follow a parabolic model based on two parameters:

$$f(x, a_1, a_2) = \sqrt{(1 + a_1 x)^2 + (a_2 x)^2} \tag{1}$$

Initial guess values of (-1, 1) were found to converge to the fit shown in Fig. 1. Plotting the model directly with various parameter values shows that a negative value of a_1 is necessary to obtain the initial downward slope. For a_2, only the magnitude affects the fit because this parameter is squared. This suggests that two solutions should be possible for minimizing χ^2. Fitting algorithms attempt to find the minimum of the *chi*^{2} landscape, but with functions like this there is not one unique solution. It is of interest to see how various fitting algorithms resolve this issue.

The data is quite noisy and looks like it may contain errors in both x and y. However, no information about errors was given, and there are advantages to assuming only errors in y. If we make this assumption, the equation for χ^2 depends on only two terms, a_1 and a_2, and the χ^2 landscape can be fully visualized in a single plot. When experimental error weights are unknown, χ^2 is defined by:

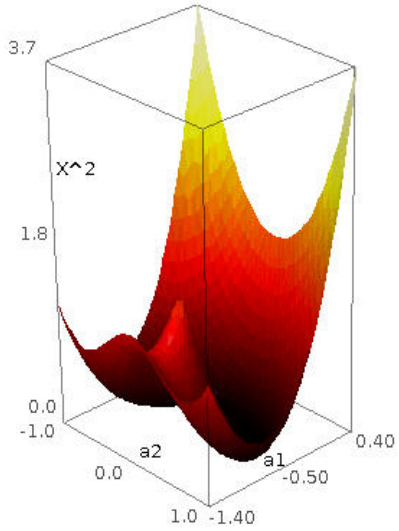

FIG. 2. χ^2 landscape plotted as 3D surface.

$$\chi^2(a_1, a_2) = \sum_{i=1}^{n} [y_i - f(x_i, a_1, a_2)]^2 \tag{2}$$

This can be plotted as a 3D surface (Fig. 2) or a contour plot (Fig. 3) versus a_1 and a_2, revealing a deep trench in the shape of a half-circle. The trench is symmetric about $a_2 = 0$ due to a_2 being squared, so there are two absolute minima, one on either side.

Initial Guesses and Multiple Minima

When multiple minima occur in the χ^2 landscape, choice of initial guess values of the fitting parameters can determine which minimum the minimization algorithm finds. In this case the two minima have perfectly equivalent χ^2 values and represent identical fits with the only difference being the sign of a_2. However, other cases are possible (such as the transmission spectrum of a thick film considered in Case 2) in which many local minima occur that are not equivalent in which the fitting algorithm

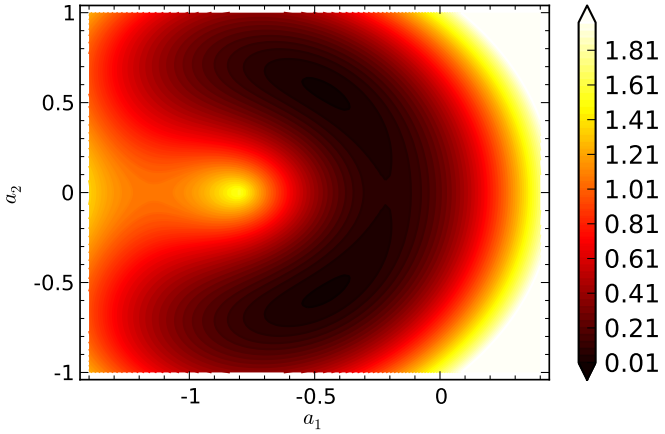

FIG. 3. χ^2 landscape as a contour plot.

may become trapped. The thick film analysis was more complicated because many parameters were involved, but we can use this simpler case to inspect how choice of guess values affects which minimum is found by the algorithm.

Table I shows the fitting results for various different initial guesses of (a_1, a_2) with the Levenberg-Marquardt method and the number of iterations completed. Since no experimental errors are known, the indicated parameter errors were estimated by multiplying the covariance matrix by the reduced χ^2 (χ^2 divided by the number of data points minus the number of model parameters). It seems that every convergent fit except one has an a_1 value of -0.456 and an a_2 value of ± 0.559. Initial guesses of $a_2 \geq 0$ find the positive a_2 basin, while initial guesses of $a_2 < 0$ find the negative basin. The one exception (indicated by *) gives a much different result for the two parameters, and if that result is fed back through the fitting algorithm, it no longer converges. Parameter errors could also not be estimated for this fit. Plotting this point on the χ^2 landscape (Fig. 4) shows that it lies very close to a saddle point (a minimum along the a_1 axis, but a peak in a_2), and at a much higher χ^2 value than the two minima. Thus, this result appears to be erroneous; a failure of this implementation of Levenberg-Marquardt to either find the minimum or to indicate that it has failed to do so, perhaps indicating the position of the final iteration before failure. Starting guesses near the saddle point along $a_2 = 0$ seem most likely to fail.

Fitting with QtiPlot

The previous results were obtained using Sage and the SciPy python module's *leastsq* function, which itself is simply a wrapper around MINPACK's *lmdif* and *lmder* algorithms. The fits in Table I were repeated using QtiPlot, a Linux-based alternative to Origin. The results are given in Table II.

TABLE I. Initial guesses and the resulting fits with Sage.

Guess (a_1, a_2)	Fit (a_1, a_2)	Iterations
(0.000, 0.000)	returns guess values	16
(1.00, 0.000)	(-0.456 ± 0.020 , 0.559 ± 0.018)	27
(10.0, 0.000)	(-0.456 ± 0.020 , 0.559 ± 0.018)	31
(0.000, 0.200)	(-0.456 ± 0.020 , 0.559 ± 0.018)	30
(0.000, 0.0100)	(-0.456 ± 0.020 , 0.559 ± 0.018)	54
(-1.00, 1.00)	(-0.456 ± 0.020 , 0.559 ± 0.018)	16
(-1.00, 0.000)	returns guess values	11
(-10.0, 0.000)	(-1.14, 7.50e-9)	18
(0.000, -0.200)	(-0.456 ± 0.020 , -0.559 ± 0.018)	30
(0.000, -0.0100)	(-0.456 ± 0.020 , -0.559 ± 0.018)	54
(-1.00, -1.00)	(-0.456 ± 0.020 , -0.559 ± 0.018)	16
(-2.00, 0.000)	returns guess values	11
(-1.14, 7.50e-9)*	returns guess values	4

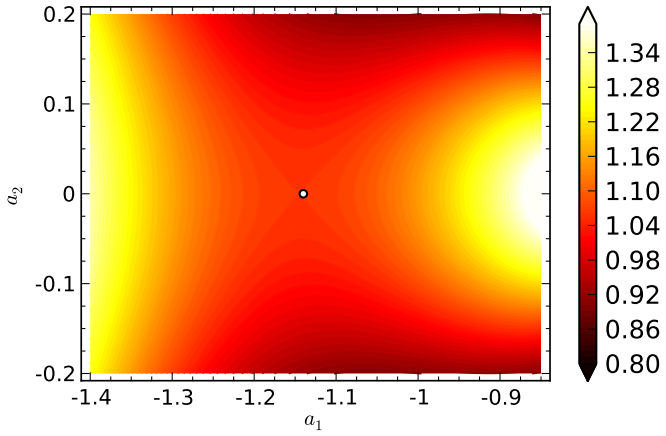

FIG. 4. χ^2 landscape saddle point and irregular fit.

TABLE II. Initial guesses and the resulting fits with QtiPlot.

Guess (a_1, a_2)	Fit (a_1, a_2)	Iterations
(0.000, 0.000)	(-0.210 ± 0.032 , 0 ± 0)	2
(1.00, 0.000)	(-0.210 ± 0.032 , 0 ± 0)	2
(10.0, 0.000)	(-0.210 ± 0.032 , 0 ± 0)	2
(0.000, 0.200)	(-0.458 ± 0.026 , 0.560 ± 0.023)	5
(0.000, 0.0100)	(-0.458 ± 0.026 , 0.560 ± 0.023)	6
(-1.00, 1.00)	(-0.458 ± 0.026 , 0.560 ± 0.023)	4
(-1.00, 0.000)	(-1.15 ± 0.14 , 0 ± 0)	2
(-10.0, 0.000)	(-1.15 ± 0.14 , 0 ± 0)	4
(0.000, -0.200)	(-0.458 ± 0.026 , -0.560 ± 0.023)	6
(0.000, -0.0100)	(-0.458 ± 0.026 , -0.560 ± 0.023)	6
(-1.00, -1.00)	(-0.458 ± 0.026 , -0.560 ± 0.023)	4
(-2.00, 0.000)	(-1.15 ± 0.14 , 0 ± 0)	3
(-1.14, 7.50e-9)*	(-1.15 ± 0.14 , 7.50e-9 ± 0)	2

Comparing the two, it seems that SciPy was more often able to find one of the χ^2 minima. The obtained parameter values and errors are slightly different, despite both programs supposedly using the same algorithm and estimating errors using the reduced χ^2. QtiPlot also used much fewer iterations. Initial guess values that included $a_2 = 0$ always yielded a fit with $a_2 = 0 \pm 0$, although a_1 always found a minimum along the $a_2 = 0$ line. In some cases this minimum was the saddle point shown in Fig. 4, otherwise it was the deeper trench which forms the half-circle discussed previously. In contrast, while SciPy often could not converge when $a_2 = 0$ was guessed, it could still sometimes find one of the two global minima. It thus appears that the choice of fitting software can have some effect on fitting results, particularly when multiple minima are involved and initial guesses include zero values or are relatively far from a χ^2 minimum.

Monte Carlo Interpolation

Solutions along the half-circle trench between the two absolute minima would also give very low χ^2 values, suggesting these would give nearly as good fits as the absolute minima and thus could also be reasonable solutions–especially considering the large noise in the data. If the noise of the data had been slightly different, the best-fit parameters would be slightly different but would likely lie along this arc. This can be investigated by interpolating the initial data, applying a random error to each point in order to produce artificial data sets and attempt to fit these.

We have seen that the choice of initial guess will affect the result with this system, so it will also affect the Monte Carlo results. Most of the guess values in Table I produce best-fit parameters that cluster around one or the other basin, particularly those guesses which begin at a_1 values lower than -0.456 and a_2 near zero. Guesses that begin at higher a_1 values than -0.456 and a_2 near zero show more interesting behavior, with best-fits divided between the two basins (Fig. 5). This is interesting since the model is fundamentally symmetric, even if the noise in the data varies. Occasionally fits are also seen in the middle of the half-circle trench where a saddle point occurs, but since they all lie along the $a_2 = 0$ line, this is likely due to a failure of the fitting algorithm (as in the * case in Table I) rather than a true minimum here for those data sets.

Manual Evaluation by Different Methods

Levenberg-Marquardt (LM) is only one possible method for minimizing χ^2, and other methods may behave differently when navigating the χ^2 landscape. The steepest-descent (SD) method involves calculating the gradient of the χ^2 expression and moving in steps down

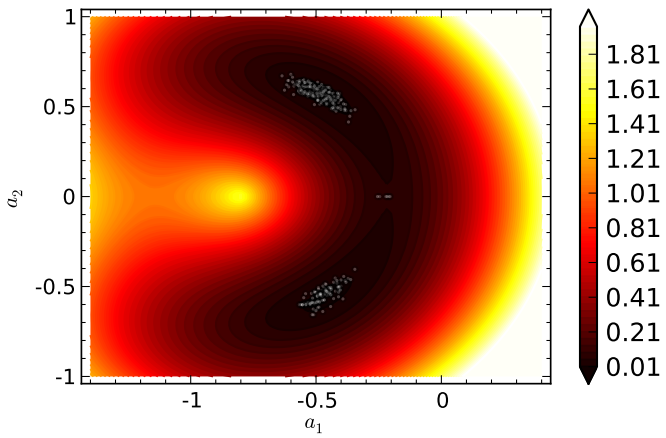

FIG. 5. χ^2 landscape with Monte Carlo interpolated fits, assuming Gaussian error with $\sigma = 0.05$ and guess parameters of $a_1 = 0, a_2 = 0.01$.

the gradient until a minimum is found. Mathematically, the gradient of a scalar function $f(a_1, a_2)$ is the vector $\nabla f = \left(\frac{\partial f}{\partial a_1}, \frac{\partial f}{\partial a_2}\right)$. The partial derivatives of χ^2 for this model function are given by:

$$\frac{\partial \chi^2}{\partial a_1} = -2\sum_{i=1}^{n} \frac{\left(y_i - \sqrt{a_2^2 x_i^2 + (a_1 x_i + 1)^2}\right)(a_1 x_i + 1)x_i}{\sqrt{a_2^2 x_i^2 + (a_1 x_i + 1)^2}} \tag{3}$$

$$\frac{\partial \chi^2}{\partial a_2} = -2\sum_{i=1}^{n} \frac{\left(y_i - \sqrt{a_2^2 x_i^2 + (a_1 x_i + 1)^2}\right)a_2 x_i^2}{\sqrt{a_2^2 x_i^2 + (a_1 x_i + 1)^2}} \tag{4}$$

The steepest descent algorithm iteratively evaluates the following expression:

$$(a_1, a_2)_i = (a_1, a_2)_{i-1} - \epsilon \left(\frac{\partial \chi^2}{\partial a_1}, \frac{\partial \chi^2}{\partial a_2}\right)_{i-1} \tag{5}$$

where ϵ is some step size. The gradient ensures that the direction of the step is along the steepest slope, the minus ensures that the algorithm moves down rather than up the gradient, and the value of ϵ determines how far the algorithm moves in the direction of the gradient in the parameter space.

Applying this method to the guess values in Table III yields Fig. 6. Since the method itself simply seeks to always move down in slope, it should be easy to become trapped in local minima if these exist. Thus, caution is needed to ensure that the starting guess is within a basin that contains a global minimum. It is also clear that when the slope along one axis is zero, the algorithm will

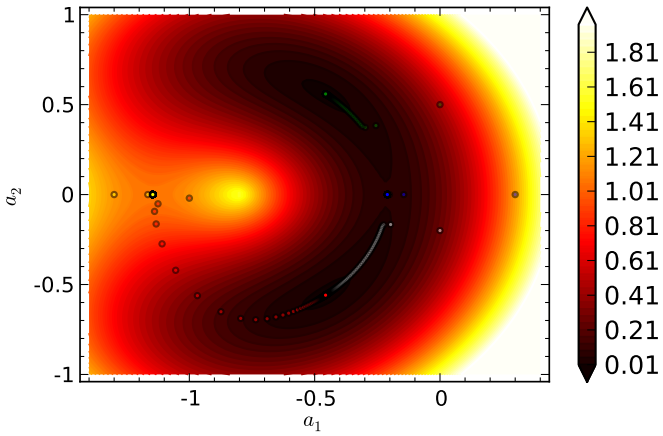

FIG. 6. Steepest descent method with $\epsilon = 0.1$, for the initial guesses in Table III.

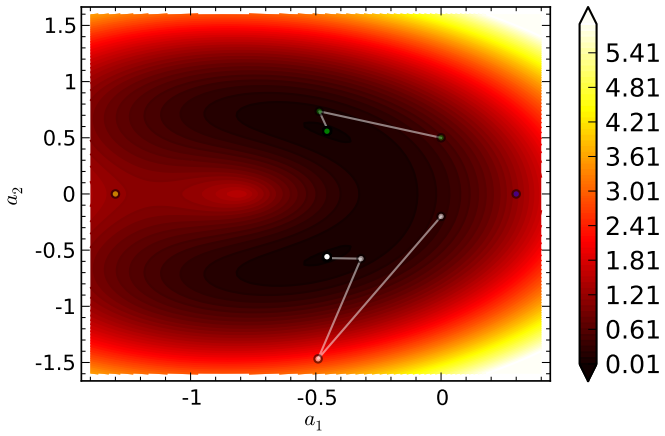

FIG. 7. Gauss-Newton method for initial guesses given in Table III.

TABLE III. Initial guesses and the plotted colors for the three fitting methods.

Guess	SD Fit	GN Fit	LM Fit	Colors
(0, 0.5)	(-0.46, 0.56)	(-0.46, 0.56)	(-0.46, 0.56)	green
(-1, -0.02)	(-0.46, -0.56)	(-0.46, -0.56)	(-0.46, -0.56)	red
(-1.3, 0)	(-1.15, 0)	None	(-1.14, 0)	yellow
(0, -0.2)	(-0.46, -0.56)	(-0.46, -0.56)	(-0.46, -0.56)	white
(0.3, 0)	(-0.21, 0)	None	(-0.21, 0)	blue

not move along that axis. In this case, because of the symmetry across $a_2 = 0$, every point with $a_2 = 0$ will have a slope of zero in the a_2 direction regardless of the value of a_1. It is no surprise that starting guesses with $a_2 = 0$ remain along the $a_2 = 0$ line and only find saddle points, as these are minima along a_1.

Another option is the Gauss-Newton (GN) method, which is applicable when the function to be minimized is a sum of squares. This method first requires computing the Jacobian matrix $\mathbf{J}$, which is the matrix of first partial derivatives of each of the residuals (as opposed to the derivatives of the sum of the squares of the residuals used in the previous case). Since the given data has 14 data points, and each residual is effectively a function of a_1 and a_2, $\mathbf{J}$ is a 14x2 matrix in this case. Each column corresponds to the residual of a single data point, while the two values are the partial derivatives vs a_1 and a_2. The Hessian matrix, the matrix of second partial derivatives, is approximated by $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ (which is reasonable if residuals are assumed to be small). One must then solve $\mathbf{H}\Delta = \mathbf{J}^T\mathbf{r}$ for Δ, where $\mathbf{r}$ is the matrix of residuals and Δ is the parameter step for that iteration. Thus in this method, the step distance as well as direction is obtained from the analysis itself. Solving for Δ requires that $\mathbf{H}^{-1}$ exists:

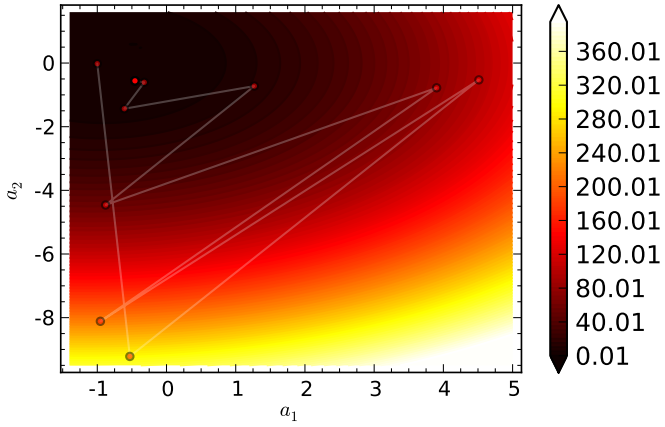

FIG. 8. Gauss-Newton method for initial guess (-1, -0.02).

$$\Delta = \mathbf{H}^{-1}\mathbf{J}^T\mathbf{r} \tag{6}$$

When attempting to apply this method to the same set of initial guesses as the steepest descent case, it was found that guesses where $a_2 = 0$ encountered singular (non-invertible) Hessian matrices, and thus could not be solved. Other guesses did converge to one of the two minima, as seen in Fig. 8. Not shown in this plot is the guess starting at (-1, -0.01), which wandered far away from the minima before eventually converging. The path traversed by this fit is shown in Fig. **??**. Comparing these two methods, it seems that Gauss-Newton converges in fewer iterations, but its path is less direct with a zig-zag character and can take it far away from its eventual destination. Both have difficulty with guesses on ridges and saddle points.

The Levenberg-Marquardt algorithm is much like the Gauss-Newton method, but with a small modification. Rather than solving $(\mathbf{J}^T\mathbf{J})\Delta = \mathbf{J}^T\mathbf{r}$ for Δ, one solves

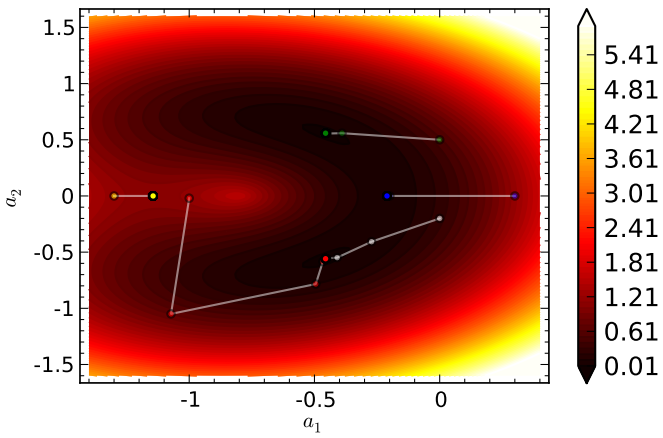

FIG. 9. Manually applied Levenberg-Marquardt method with guess values from Table III.

$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\Delta = \mathbf{J}^T\mathbf{r}$ where λ is some positive scalar and $\mathbf{I}$ is the identity matrix. The solution for the step taken in each iteration is then given by:

$$\Delta = (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{r} \tag{7}$$

Choosing a lambda value of 0.1, the guesses in Table III were fit manually according to this equation, yielding Fig. 9. The small offset introduced by λ was enough to avoid singular matrices and find a solution for Δ in Eq. 7, but starting guesses with $a_2 = 0$ still only were able to find saddle points rather than true minima, converging to approximately the same solutions as the steepest descent method. Reviewing the results in Table III, all three methods either found the same minimum for a given guess, or failed to find either minimum. Differences are only evident if the paths traversed by the algorithms are compared. Steepest descent takes a more direct route down the gradient, while Gauss-Newton moves erratically and Levenberg-Marquardt takes a fairly direct route and arrives after only a few iterations. In the limit where λ goes to zero, Levenberg-Marquardt becomes equivalent to Gauss-Newton. As the values of λ is increased, the distance between steps becomes smaller and the path increasingly resembles that of steepest descent.

Conclusion

Fitting often simply involves providing a model and a set of guess values and receiving an answer, but sometimes it is useful to visualize the solution to the χ^2 equation for a range of parameter values. This can reveal sub-optimal local minima where algorithm may become trapped, multiple global minima, sensible ranges of guess values, and the path that the algorithm takes in obtaining the solution. The different algorithms appear to encounter similar difficulties with certain guess values and produce similar results if acceptable guesses are provided.