



RESEARCH ARTICLE

10.1002/2014EA000090

Key Points:

- Simple code to calculate uncertainties on a reduced chi-square fit is provided
- Ensures covariance matrix diagonal is positive, fewer fitting iterations needed
- A fitting routine of your choice may be used, then uncertainties calculated

Correspondence to:

R. J. Wilson,
Rob.Wilson@lasp.colorado.edu

Citation:

Wilson, R. J. (2015), Error analysis for numerical estimates of space plasma parameters, *Earth and Space Science*, 2, 201–222, doi:10.1002/2014EA000090.

Received 2 JAN 2015

Accepted 17 APR 2015

Accepted article online 27 APR 2015

Published online 10 JUN 2015

Error analysis for numerical estimates of space plasma parameters

R. J. Wilson¹
¹Laboratory for Atmospheric and Space Physics, University of Colorado Boulder, Boulder, Colorado, USA

Abstract Many papers estimate space plasma parameters from instrumentation data via a fitting method, such as reduced chi-square minimization of a model to the data; however, it is currently rare to see uncertainties for those estimates given in the form of error bars or a covariance matrix. This paper seeks to address this issue by providing a simple method that will provide the covariance matrix and therefore uncertainties with little extra computation, no matter how complex the model. Using established “black box” minimization codes will provide a best fit to the data but may not provide the covariance matrix, while others may provide the covariance matrix (providing their settings are tuned appropriately) but tend to locate a best fit vector to near-machine precision first. Our method allows the fitting to a physically sensible number of decimal places for the instrument yet also provides the covariance matrix with far fewer iterations required to locate the best fit values, greatly decreasing code run time for the fit procedure itself—a great benefit when there are years of data, or multiple spacecraft, to analyze. While the underlying method utilizing the Hessian matrix is not new, the application is currently rarely applied to spacecraft data and this approach is simple to implement. This paper reviews the basic technique and application to data and ends with simple pseudocode that anyone may employ to calculate the covariance matrix for the fitted parameters.

1. Introduction

Most outer planetary missions have an instrument on board that measures the in situ thermal plasma. These include Pioneer 10 and 11 (Ames Research Center Plasma Analyzer [Wolfe *et al.*, 1974]), Voyager 1 and 2 (Plasma Experiment (PLS) [Bridge *et al.*, 1977]), Ulysses (Solar Wind Observations Over the Poles of the Sun (SWOOPS) [Bame *et al.*, 1992]), Galileo (Plasma Science Experiment (PLS) [Frank *et al.*, 1992]), Cassini (Cassini Plasma Spectrometer (CAPS) [Young *et al.*, 2004]), New Horizons (Solar Wind Around Pluto (SWAP) [McComas *et al.*, 2008]), and Juno (Jovian Auroral Distributions Experiment (JADE) [McComas *et al.*, 2013]). Each instrument is different, but they all fundamentally measure the number of incident ions/electrons that impact on them (units of counts, with the exception of the Voyagers that measure current due to the incoming plasma). Most instruments measure counts at a given energy per charge (often “singles” data or a coincidence counter), while some have time-of-flight capabilities that allow the mass and charge of those ions to be calculated.

While counting statistics are useful, scientists want plasma parameters such as densities, temperatures (isotropic or anisotropic), and bulk velocities of the plasma species. These values must be deduced from those counting statistics.

Statistical moments can often be calculated that provide these parameters and are essentially summations, where the r^{th} moment is given by:

$$M_r = \int v^r f d^3v. \quad (1)$$

where f is the phase space distribution function in velocity (v) space. Density is given by the zeroth moment, M_0 , while $M_1 = M_0 V$, such that the bulk velocity V is given by $V = \frac{M_1}{M_0}$. Higher-order moments provide other parameters. This is a standard technique and covered in many texts, such as Baumjohann and Treumann [1999] and Boyd and Sanderson [1969]. When coded, the integral becomes a summation of terms, and to calculate the uncertainty on its parameters, one only has to do standard propagation of errors, as found in standard text books.

©2015. The Authors.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

This standard technique requires fully knowing f for each plasma species, that is for the entire 4π steradians field of view and over all velocity space per ion species (for the purposes of this document we may consider electrons to be a negative ion species). Unfortunately, plasma instruments tend to measure total ion or electron counts, not f , and the conversion from counts to f is not always possible without additional assumptions.

For electrons there is only one “ion species” with a known mass and charge (ignoring the rare instances when negative ions enter the instrument, e.g., Titan’s ionosphere [Coates *et al.*, 2007]). Counts can then easily be converted to f with the instrument calibrations (geometric factor, sensor efficiency, and knowing what energy, E , and therefore v as $E = \frac{1}{2}mv^2$, where m is mass of the electron).

However, Cassini is a three-axis stabilized craft, so there is not the full 4π steradians viewing for their electron instrument (very occasionally there are spacecraft rolling intervals, with a spin period around 30 min, useful for calibration, but poor time resolution for science). Galileo had a spin platform with a period of ≈ 20 s, such that the plasma instruments could view all sky over one revolution; however, due to the antenna malfunction and the drastically reduced bandwidth, they only downlinked every third or fourth energy step (dependent on telemetry mode) of a subset of the full range per spin, providing gaps in velocity space through the plasma distributions of interest.

For ion detectors there are similar issues with a lack of all-sky viewing on Cassini or a lack of all velocity space for Galileo. However, there are added complications in that multiple ion species exist: in the solar wind H^+ and He^{++} ions are present, the inner Saturnian magnetosphere predominantly contains light ions (H^+ and H_2^+) and water group ions (O^+ , OH^+ , H_2O^+ , and H_3O^+), while at Jupiter H^+ , O^+ , O^{++} , S^+ , S^{++} , and S^{+++} ions are predominantly observed. Recall that the detector measures ion counts and the most common quantities are “ion singles,” which is the number of particles detected within an energy-per-charge bin but contains no information about the ion species. This means that if a given energy step counted 100 ions, we must first divide the counts by v^4 [Paschmann and Daly, 1998; Wilson *et al.*, 2012a] in order to convert the measured quantities to f . Kinetic energy provides v such that $v^4 = 4E^2/m^2$. The mass squared term becomes an issue when you do not know what fraction of the 100 counts are from which species (e.g., it could potentially be 76 counts of species 1 and 24 counts of species 2, or vice versa, or any other combination). For instance, at Saturn, confusing a H^+ ion with an OH^+ ion would skew v^4 by a factor of $289 (= 17^2)$. At Jupiter, O^+ and S^{++} have the same mass to charge ratio and therefore the same energy to charge ratio, meaning both ion species should peak in counts at the same measured energy step. This is also true for O_2^+ and S^+ ions with identical mass to charge ratios. Even when the mass to charge ratios are not identical, to the energy resolution of the singles data set, we cannot trivially separate out similar ion distributions. While the water group ion species have distinct mass to charge ratios of 16, 17, 18, and 19, respectively, this is only useful in singles data if they are so cold that their distributions are very narrow such that one ion distribution does not overlap any part of another. This is not true at Saturn; each distribution is relatively wide such that all you can observe are the four distributions superposed to one smooth blended distribution. Without further time-of-flight data, they simply cannot be separated, as such a single mass is often assumed to represent the entire distribution.

Due to restrictions in viewing, velocity space sampling, and knowledge of proportions of plasma species, an alternative method of calculating plasma parameters is often applied—that of a forward model. In forward modeling we assume specific ion species and their distribution, typically a Maxwellian function [e.g., Wilson *et al.*, 2008; Elrod *et al.*, 2012; Wilson *et al.*, 2012b, 2013; Nicolaou *et al.*, 2014] or a Kappa distribution [e.g., Nicolaou *et al.*, 2014].

The following analysis assumes that your chosen method for calculating plasma parameters is via a forward model method and demonstrates how to calculate uncertainties on the fitted parameters. There may be complications in your model due to assumptions that had to be made, e.g., poorly determined parameters and difficulties in returning a unique solution. This paper does not focus on such issues; it assumes you have a model, and we concentrate on generating uncertainties for your fitted parameters to that model. Other typical methods, not covered here, assume that the data are isotropic such that a one-dimensional slice of data represents all 4π steradians and then use statistical moments (e.g., Lewis *et al.* [2008] for Cassini electron data). Another method, not covered here, assumes that you know how to partition total counts into the correct fraction per ion species and use the parts of the 3-D distributions you do have with symmetry arguments to fill in the missing phase space before calculating statistical moments [e.g., Thomsen *et al.*, 2010, 2013]. This paper is not intended to focus on one method of assigning plasma parameters to data but to provide a general error analysis technique that may be applied to any forward model.

Section 2 introduces the reduced chi-square form, while section 3 introduces the Hessian matrix, curvature matrix, and how to convert to the covariance matrix. Section 4 explains how to use this knowledge and ends with the simple equations required to calculate the covariance matrix after justifying their form, along with pseudocode that may be utilized to generate the alpha matrix (and therefore covariance matrix). We conclude the paper in section 5.

2. Cost Function and Reduced Chi-Square χ_r^2

The cost function, which we will denote as Z , is a scalar value that provides a comparison of a measured data set to a simulated data set (that may be generated from various free parameters). The value of Z is minimized in order to find the best fit set of free parameters that will generate a simulated data set that best matches the measured data set, where “best” is defined as giving the smallest cost function.

Usually, a cost function is based on a chi-square value, χ^2 , that is defined by:

$$\chi^2 = \sum_{i=1}^n \frac{(Obs_i - Sim_i)^2}{\sigma_i^2} \quad (2)$$

Here Obs is the measured (observed) data, Sim is the simulated data, and σ^2 is the variance that describes the uncertainty in the measured data. All three quantities, Obs , Sim , and σ^2 , are arrays of n elements.

The reduced chi-square value, χ_r^2 , is preferred, which normalizes χ^2 using n and the number of free parameters, ν .

$$\chi_r^2 = \frac{\chi^2}{n - \nu} = \frac{1}{n - \nu} \sum_{i=1}^n \frac{(Obs_i - Sim_i)^2}{\sigma_i^2} \quad (3)$$

The advantage of using the reduced chi-square function is that it is well understood and normalizes the result by the number of data points to aid comparison of different data intervals. In addition, since it is quadratic in nature, it makes the algebra required for covariance calculations manageable. The cost function then becomes a function of the free parameter vector, $\vec{p}$, that has ν elements. Note that in order to get a fit, you must have more data points than free parameters, so $n > \nu$, which ensures that the right-hand side of the equation is always positive; hence, Z must always be positive too.

$$Z(\vec{p}) = \frac{1}{n - \nu} \sum_{i=1}^n \frac{(Obs_i - Sim(\vec{p})_i)^2}{\sigma_i^2} \quad (4)$$

A quick warning about equation (4): σ_i is generally found from the observations using Poisson statistics. However, if any $\sigma_i = 0$, then this will result in a divide by zero (crashing code or giving an infinity such that $Z = \infty$ no matter what $\vec{p}$ is), so be sure to check for any cases where $\sigma_i = 0$ and replace them with a realistic nonzero estimate. If using Poisson statistics as typical for particle instruments, then it is reasonable to replace any case of $\sigma_i = 0$ with $\sigma_i = 1$.

Also note that a background term is not mentioned in equation (4), as this is the general form. If there is a background signal to be removed from the observation data, we will presume you have already accounted for that in your Obs_i 's and adjusted your σ_i 's accordingly using standard propagation of errors. For example, if it is a standard background from the measured data set, then both the signal and background will likely have uncertainties based on Poisson statistics, such that $Obs_i = Signal_i - Background_i$ and $\sigma_i^2 = Signal_i + Background_i$.

3. Uncertainty Equations for a Reduced Chi-Square Fit

In the determination of the uncertainties of fitted parameters from a reduced chi-square fit we only require three equations, which are standard text book equations (their proofs can be found in such books as *Bevington and Robinson* [2003]). The first equation is the Hessian matrix, $Hess$, which is a 2-D matrix of size ν by ν . The alpha matrix (also known as the curvature matrix) is then just the Hessian matrix with each element

divided by 2. Finally, the covariance matrix, S^2 , is the matrix inverse of the alpha matrix. So for a given best fit Z with the free parameter vector $\vec{p}$ those three equations are:

$$Hess = \begin{bmatrix} \frac{\partial^2(Z)}{\partial p_1^2} & \frac{\partial^2(Z)}{\partial p_1 \partial p_2} & \frac{\partial^2(Z)}{\partial p_1 \partial p_3} & \dots & \frac{\partial^2(Z)}{\partial p_1 \partial p_n} \\ \frac{\partial^2(Z)}{\partial p_2 \partial p_1} & \frac{\partial^2(Z)}{\partial p_2^2} & \frac{\partial^2(Z)}{\partial p_2 \partial p_3} & \dots & \frac{\partial^2(Z)}{\partial p_2 \partial p_n} \\ \frac{\partial^2(Z)}{\partial p_3 \partial p_1} & \frac{\partial^2(Z)}{\partial p_3 \partial p_2} & \frac{\partial^2(Z)}{\partial p_3^2} & \dots & \frac{\partial^2(Z)}{\partial p_3 \partial p_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2(Z)}{\partial p_n \partial p_1} & \frac{\partial^2(Z)}{\partial p_n \partial p_2} & \frac{\partial^2(Z)}{\partial p_n \partial p_3} & \dots & \frac{\partial^2(Z)}{\partial p_n^2} \end{bmatrix} \quad (5)$$

$$\alpha = \frac{Hess}{2} \quad (6)$$

$$S^2 = \alpha^{-1} \quad (7)$$

These matrices are symmetric down the major diagonal: $\frac{\partial^2(Z)}{\partial p_i \partial p_j} = \frac{\partial^2(Z)}{\partial p_j \partial p_i}$ and likewise $S_{ij}^2 = S_{ji}^2$.

4. Practically Using the Equations

Let us simplify this a little; let us say we only have two free parameters ($\nu = 2$) where the free parameter vector $\vec{p} = [x, y]$ (i.e., $p_1 = x$ and $p_2 = y$), such that Sim is a function of x and y , $Sim([x, y])$.

$$Z[x, y] = \frac{1}{n - \nu} \sum_{i=1}^n \frac{(Obs_i - Sim_i([x, y]))^2}{\sigma_i^2} \quad (8)$$

Clearly, n , ν , Obs_i , and σ_i are all constants in the equation; only Sim_i varies as a function of $\vec{p}$. As such, the Hessian matrix is now given by three unique second-order partial derivatives:

$$Hess = \begin{bmatrix} \frac{\partial^2(Z)}{\partial x^2} & \frac{\partial^2(Z)}{\partial x \partial y} \\ \frac{\partial^2(Z)}{\partial x \partial y} & \frac{\partial^2(Z)}{\partial y^2} \end{bmatrix} \quad (9)$$

The following equations work equally well when ν is greater than 2 as each element of the Hessian matrix only compares one or two of the free parameters at a time, holding any others fixed at their best fit value. The example here is a reduced chi-square function, but any function that is least squares in form will work.

In order to work out the uncertainty of the fitted parameters, we must know the "best fit" value as found by the minimization of Z . Let us call that $\vec{p}_0$ where $\vec{p}_0 = [p_{01}, p_{02}, \dots, p_{0n}]$, and in our 2 free parameter case, our best fits are x_0 and y_0 that give a best fit reduced chi-square value of Z_0 . Now all one needs to do is to use your simulation equation within the reduced chi-square equation and to analytically calculate the second-order partial differentiation terms of the Hessian matrix at your best fit location.

For the sake of an example, let us assume that our model is a simple 1-D isotropic Maxwellian for a single species of the simple form:

$$Sim_i = C_i x^{-\frac{3}{2}} e^{\frac{-(u_i - y)^2}{x}} \quad (10)$$

where C_i 's and u_i 's are constants and x and y are analogous to temperature and bulk velocity, respectively, and are our two free parameters. Substituting this into equation (8), we get:

$$Z[x, y] = \frac{1}{n - \nu} \sum_{i=1}^n \frac{(Obs_i - C_i x^{-\frac{3}{2}} e^{\frac{-(u_i - y)^2}{x}})^2}{\sigma_i^2} \quad (11)$$

4.1. Analytically Solving for the Uncertainties

We can use equation (11) to calculate the Hessian matrix elements analytically, and from there work out the alpha matrix and subsequently the covariance matrix. This just requires stepping through the mathematics of partial differentiation. In doing that, the result for this “simple” example for the elements of the Hessian matrix may be expressed as:

$$\frac{\partial^2 Z[x, y]}{\partial x^2} = \sum_{i=1}^n \frac{\frac{1}{2} C_i e^{\frac{2(u_i-y)^2}{-x}}}{(n-\nu) \sigma_i^2 x^6} \left(\left(8C_i - 5Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) (3x - 4(u_i - y)^2) + \frac{4}{x} \left(2C_i - Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) (u_i - y)^4 \right) \quad (12)$$

$$\frac{\partial^2 Z[x, y]}{\partial y^2} = \sum_{i=1}^n \frac{4C_i e^{\frac{2(u_i-y)^2}{-x}}}{(n-\nu) \sigma_i^2 x^4} \left(\frac{2}{x} \left(2C_i - Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) (u_i - y)^2 - C_i + Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) \quad (13)$$

$$\frac{\partial^2 Z[x, y]}{\partial x \partial y} = \frac{\partial^2 Z[x, y]}{\partial y \partial x} = \sum_{i=1}^n \frac{2C_i (u_i - y) e^{\frac{2(u_i-y)^2}{-x}}}{(n-\nu) \sigma_i^2 x^5} \left(\frac{2}{x} \left(2C_i - Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) (u_i - y)^2 - 8C_i + 5Obs_i x^{\frac{3}{2}} e^{\frac{(u_i-y)^2}{x}} \right) \quad (14)$$

To get actual numbers for the elements of the Hessian matrix, we solve the above three equations where $x = x_0$ and $y = y_0$. From here we use equations (6) and (7) to get the covariance matrix. If there are more than two free parameters, the process is the same, just build up the Hessian matrix pairs of free parameters at a time.

While the process is straightforward, the analytical partial differentiations get complicated quickly and have to be coded without introducing any errors. This is not trivial for this simple example of one species. Now imagine this same procedure when you have two species (so two distributions), or with three different velocity components (to give three dimensional flow), or with anisotropic Maxwellians requiring two different temperatures, or if your model includes a cross-talk smearing to match the data, adding up to as much as nine free parameters. This analytical approach becomes unfeasible quickly. We find that the following numerical technique is much easier.

Additionally, the analytical technique assumes that one knows the best fit parameters exactly, such that the first-order partial differential equations are zero. Your minimization code found the best fit to a predefined tolerance (examine the documentation for the minimization code you used) that may be extremely close to the analytical best fit but may not give zero first-order partial differential equations, and then your uncertainties calculated from the second-order partial differential equations may be skewed from their true value.

4.2. A Numerical Approach to Solving for the Uncertainties

It is easier for a user (both practically and in writing code) to calculate the elements of the Hessian matrix by numerical methods. The best fit reduced chi-square is where it is at a minimum or, in other words, the ν dimension hypersurface is at a minimum. This multidimensional well can be parameterized in quadratic terms near the best fit location, simply because this was a least squares best fit (via the chi-square method). The parameterization of the well can be done in simple terms; hence, the partial differentiations required for the Hessian matrix become trivial.

The following subsection explains this method of parameterizing this hypersurface well in greater detail. The essence is that three quadratic equations can be used to describe the hypersurface well two dimensions at a time (all other dimensions kept constant at their best fit values):

$$Z[x, y = y_0] = Ax^2 + Bx + C \quad (15)$$

$$Z[x = x_0, y] = Dy^2 + Ey + F \quad (16)$$

$$Z[x, y] = Gx^2 + Hx + I + Jy^2 + Ky + Lxy \quad (17)$$

The first equation describes the one-dimensional cut of x (while $y = y_0$), the second the one-dimensional cut of y (while $x = x_0$), and the third the two-dimensional well of x and y . Constants A to L are to be found numerically as shown below, although in the end, only A , D , and L are required to be known to calculate uncertainties.

4.3. Parameterize the Hypersurface to Generate the Covariance Matrix

The first equation for the cut through x may be rewritten after replacing B and C to leave only A unknown. C may be written in terms of A and B by introducing Z_0 , where $Z_0 = Z[x_0]$, and B may be replaced by knowing that the first-order differential at the best fit location must be zero. The details are straightforward and have been moved to Appendix A but result in the following equation:

$$Z[x] - Z_0 = A(x - x_0)^2 \quad \Rightarrow A > 0 \quad (18)$$

The left-hand side of the equation must be positive as $Z[x] \geq Z_0$, resulting in that A must be a nonzero positive value always. The Hessian matrix element is then the second-order differential, which can then be used to calculate the α matrix component, which is simply half the Hessian.

$$\frac{\partial^2 Z[x]}{\partial x^2} = 2A \quad \Rightarrow \alpha_{x,x} = A \quad (19)$$

The same procedure can be repeated for the cut through y , replacing E and F (again, see Appendix A for more details).

$$Z[y] - Z_0 = D(y - y_0)^2 \quad \Rightarrow D > 0 \quad (20)$$

$$\frac{\partial^2 Z[y]}{\partial y^2} = 2D \quad \Rightarrow \alpha_{y,y} = D \quad (21)$$

A similar technique can be used for the final two-dimensional cut (see Appendix B for details), but first let us consider the Hessian matrix element:

$$\frac{\partial^2 Z[x,y]}{\partial x \partial y} = \frac{\partial^2 Z[x,y]}{\partial y \partial x} = L \quad \Rightarrow \alpha_{x,y} = \alpha_{y,x} = \frac{L}{2} \quad (22)$$

Therefore, from our unknown coefficients G , H , I , J , K , and L , only L is required to be known. In the one-dimensional cases it is easy to get the computer to find the three coefficients (e.g., A , B , and C) with an often built-in polynomial solver. However, numerically finding the best fit of the six parameters, G – L , with limited data around the well does not always fit the hypersurface adequately. This may result in a poorly calculated L so that when we invert the α matrix, we find that the diagonal of the covariance matrix has negative values. This is unphysical as the diagonals of the covariance matrix ($\sigma_{i,i}^2$ where $i = j$) must be positive so their square root can be calculated to provide the nonindependent standard deviation of each free parameter (uncertainty of p_i is $\sigma_{i,i}$). The solution is to reduce the number of unknown coefficients as before, which is straightforward and described in Appendix B, but using similar techniques to before, and realizing that we have already fitted the two one-dimensional cases, the equation simplifies to:

$$Z[x,y] - Z_0 = A(x - x_0)^2 + D(y - y_0)^2 + L(x - x_0)(y - y_0) \quad (23)$$

Here the A and D are the same values as from the one-dimensional cases, and while both A and D must be positive nonzero values by definition, L could be either positive or negative. Since A and D are already known, only L has to be solved for, which is much simpler and much more robust than attempting to solve for all six coefficients G to L .

At this point the whole Hessian matrix can be assembled for this example case of two free parameters:

$$Hess = \begin{bmatrix} \frac{\partial^2(Z)}{\partial p_1^2} & \frac{\partial^2(Z)}{\partial p_1 \partial p_2} \\ \frac{\partial^2(Z)}{\partial p_2 \partial p_1} & \frac{\partial^2(Z)}{\partial p_2^2} \end{bmatrix} = \begin{bmatrix} 2A & L \\ L & 2D \end{bmatrix} \quad (24)$$

From which the conversion to the α matrix is just half that:

$$\alpha = \frac{Hess}{2} = \begin{bmatrix} A & L/2 \\ L/2 & D \end{bmatrix} \quad (25)$$

The matrix inverse of α will then provide the covariance matrix. For this simple two free parameter case, the matrix inversion is simple:

$$S^2 = \alpha^{-1} = \frac{1}{AD - L^2/4} \begin{bmatrix} D & -L/2 \\ -L/2 & A \end{bmatrix} = \begin{bmatrix} \frac{D}{AD-L^2/4} & \frac{-L/2}{AD-L^2/4} \\ \frac{-L/2}{AD-L^2/4} & \frac{A}{AD-L^2/4} \end{bmatrix} \quad (26)$$

Note here that to be able to computationally take the square root of the diagonal (without crashing code or generating complex numbers), they must be positive values.

Our best fit i^{th} free parameters then have an uncertainty of $\pm \sqrt{S^2_{i,i}}$. From our free parameter vector ($\vec{p} = [x, y]$) we have minimized the reduced chi-square to get a best fit parameter vector, $\vec{p}_0$:

$$\vec{p}_0 = \left[x_0 \pm \sqrt{\frac{D}{AD - L^2/4}}, y_0 \pm \sqrt{\frac{A}{AD - L^2/4}} \right] \quad (27)$$

This is for a simple case where there are only two free parameters; however, often the number of free parameters is greater. Yet the process is the same; build up the Hessian matrix diagonals one free parameter at a time, and the off-diagonal elements compare two free parameters at a time while holding the others fixed at their best fit value. From there the α matrix is trivially divided by 2. Inverting matrices of sizes larger than 2 by 2 quickly gets cumbersome, but there are many standard techniques/codes to numerically invert them, and higher-level programming languages often have their own built-in function for this job, which this author recommends you use.

4.4. Covariance Versus Variance

If each of the free parameters were independent from each other within the simulation model (rarely the case in space physics or Maxwellian models), then we do not need to create a full Hessian or α matrix as all nondiagonal terms ($Hess_{i,j}$ or $\alpha_{i,j}$ where $i \neq j$) would be zero. Matrix inversion in such a case retains all nondiagonal terms as zero, and the diagonal terms are simply the reciprocals of each element. That is, the variance of free parameter i , $S^2_{i,i}$, is just $1/\alpha_{i,i}$. (Note that some texts use σ^2 for variance or covariance; here we use S^2 for covariance to avoid confusion with σ used earlier in equation (2) to denote the uncertainty on the measured values (σ_i) that are being fitted via minimizing the chi-square.) In the above example, the standard deviation (calculated from the variance) for the best fit x is simply $\sqrt{\frac{1}{D}}$; likewise, for y the standard deviation (calculated from the variance) is $\sqrt{\frac{1}{A}}$. This highlights the importance of A and D being nonzero (else would crash many computer codes or generate infinity) and positive so that the square root computer command does not crash nor generate a complex number.

The covariance matrix assumes that each free parameter can affect the other free parameters within the simulation model; they are dependent, and the covariance matrix has nonzero nondiagonal terms. The covariance of free parameter i is taken as the i^{th} diagonal element ($S^2_{i,i}$) which, due to the matrix inversion, is now affected by all the other free parameters. The standard deviation of this best fit free parameter is the square root of $S^2_{i,i}$, which is always greater than the standard deviation found from the variance.

Confusion can clearly arise as the square root of the independent variance and dependent covariance are both described as the “standard deviation,” so it is important to state which matrix (variance or covariance) any standard deviations were derived from.

4.5. Robustly Calculating A , D , and L at the Best Fit Location

We may now apply linear regression in order to work out A , D , and L by sampling n points near the best fit locations, such that we have x_i and y_i giving $Z[x_i, y_i]$. The full algebra is in Appendix C which derives the following equations. For instance, our minimization code from earlier tells us that at $x = x_0$ and $y = y_0$, then $Z = Z_0$.

$$A = \frac{\sum_{i=1}^n (x_i - x_0)^2 (Z[x_i, y_0] - Z_0)}{\sum_{i=1}^n (x_i - x_0)^4} \quad (28)$$

$$D = \frac{\sum_{i=1}^n (y_i - y_0)^2 (Z[x_0, y_i] - Z_0)}{\sum_{i=1}^n (y_i - y_0)^4} \quad (29)$$

Then using A and D , L can be calculated:

$$L = \frac{\sum_{i=1}^n (x_i - x_0)(y_i - y_0)(Z[x_i, y_i] - Z_0 - A(x_i - x_0)^2 - D(y_i - y_0)^2)}{\sum_{i=1}^n (x_i - x_0)^2 (y_i - y_0)^2} \quad (30)$$

The above will work perfectly assuming that your best fit estimates are precisely at the best fit value (at the global minimum of the hypersurface). But what if they are slightly off?

4.6. Robustly Calculating A , D , and L Near the Best Fit Location With Quantized Parameters

It is a simple fact that computer minimization routines produce best fit results that are not the true best fit result. The minimization code ends when a particular threshold or tolerance (likely based on an internal step size) is met or the number of iterations has exceeded a certain limit. Rerunning the minimization a thousand times will often result in half a dozen similar best fit parameter vectors; although one set strictly has a smaller χ_r^2 than the others, all sets met the minimization code's threshold to finish and exit. These threshold values and internal step size they use are often tunable by the user and are worth checking prior to using the code.

If the best fit free parameter representing ion temperature is 5.123456 eV in one minimization run, 5.123458 eV on another, and 5.123453 eV on a third run, they are essentially all the same especially when practically we might only believe the values to a few significant figures; 5.123 eV is a perfectly believable result from a spacecraft instrument, while 5.123456 eV is not.

To speed up processing times, one could restrict the minimization routines to use quantized values, for example, each free parameter to three decimal places only, and the first line of the simulated model would truncate the values passed to it to three decimal places. In doing so, when the minimization code presents the best fit parameters on exit, one must apply the same truncation to the best fit values it presents. As such, the best fit temperature may be returned as 5.123 eV, whereas the true best fit may actually be 5.12342 eV, an offset of 0.00042 eV, although in practice the offset is unknown aside from being less than 0.001 if truncating to three decimal places, etc. This offset can have an effect on your calculated uncertainties, as such it is important when sampling near the best fit location to sample on both sides, both above and below the value in order to average this out. Ensuring you do two-sided sampling can greatly reduce the effect of this offset, while doing symmetric two-sided sampling can remove the effect completely from your uncertainties.

4.6.1. The Effect of Quantizing the Free Parameters for Calculating A

Earlier, we found A based on the returned best fit value of $x(= x_0)$ giving the returned best fit ($Z[x_0] = Z_0$). However, x_0 probably is offset from the true minimum due to the minimization routine reaching its tolerance and stopping before the true minimum (to full machine precision) was reached. We previously solved for:

$$Z[x] = A(x - x_0)^2 + Z[x_0] \quad (31)$$

This equation assumed that the first-order differential was zero in order to replace B as a multiple of A (see Appendix A and equation (A5)) which is only true if x_0 is at the true minimum, and as such this equation may not return the true value of A (and similarly for D for $y = y_0$). However, let us assume that the actual analytical best fit (with no rounding) is at $x_a(≠ x_0)$ such that:

$$Z[x] = a(x - x_a)^2 + Z[x_a] \quad (32)$$

where $Z[x_a]$ is the true best chi-square ($Z[x_a] < Z[x_0]$) and a is the actual polynomial term. How does a compare with A ?

Denoting the offset δ such that $x_a - \delta = x_0$ and setting $x = x_0$ in the above equation, one can find $Z[x_0]$ and substitute that in equation (31) (more explanation and intermediate steps for this section can be found in Appendix D).

$$Z[x] = A(x - x_0)^2 + (a\delta^2 + Z[x_a]) \quad (33)$$

From which we can work out A using linear regression as before.

$$\frac{A}{a} = \frac{\sum_{i=1}^n (x_i - x_0)^3 (x_i - x_0 - 2\delta)}{\sum_{i=1}^n (x_i - x_0)^4} \quad (34)$$

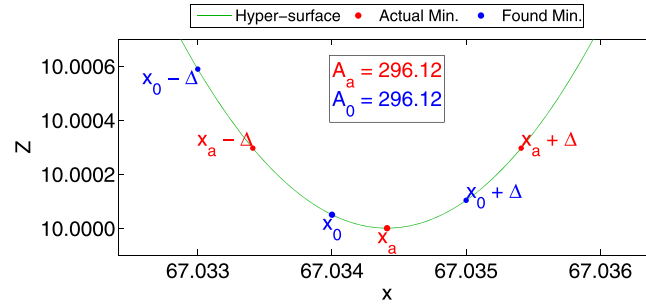


Figure 1. The hypersurface minimum in one dimension. The actual minimum is at $x_a = 67.03441$ (red), with nearest neighbors at a Δ of 0.001 away on each side. The blue x_0 found minimum at 67.034 (rounded to three decimal places) is clearly not at the true minimum; however, the nearest neighbors Δ away are greater than $Z[x_0]$ such that it is the minimum when stepping in intervals of Δ . Calculating A from either the three red or three blue data points gives the same value of 296.12.

If our sampling values for x_i are taken at equidistant intervals near the best fit location, then $x_i = x_0 + i\Delta$ (where $\Delta > \delta$) and i ranges from a to b ; then substituting in the previous equation, it simplifies greatly (see Appendix D for intermediate steps):

$$\frac{A}{a} = 1 - \frac{2\delta}{\Delta} \frac{\sum_{i=a}^b i^3}{\sum_{i=a}^b i^4} \quad (35)$$

If you were to only sample in a one-sided way (i.e., above the best fit value) i from $a = 1$ to $b = n$, the equation becomes:

$$\frac{A}{a} = 1 - \frac{\delta}{\Delta} \frac{15n(n+1)}{(2n+1)(3n^2+3n-1)} \quad (36)$$

This would underestimate A ($< a$) for the α matrix and overestimate the calculated variance. As n increases, A will get closer to a , but they will never be the same. If doing a one-sided calculation with just one sample ($n = 1$):

$$\frac{A}{a} = 1 - \frac{2\delta}{\Delta} \quad (37)$$

The superior approach is to do a symmetric two-sided calculation, where the summation is over i from $a = -n$ to $b = n$. As shown in Appendix D, the $\sum_{i=a}^b i^3$ in equation (35) now becomes zero, the last term vanishes leaving the result that $A = a$. Despite the found best fit not being the actual best fit, the uncertainty calculation was unaffected for the symmetric two-sided sampling case. Figure 1 shows an example of a one-dimensional hypersurface around the best fit location in green, with the actual minimum location in red at x_a and the “found” best fit location in blue at x_0 . By symmetrically sampling both sides at a distance Δ away, we may calculate A and find that both the red and blue values provide the same value.

Not only is the symmetric two-sided sampling case accurate but the value of A is also independent of the value of n used in the calculation. As such, there is no gain in using $n > 1$, meaning computation time is saved by only having to sample one value each side of the best fit value.

Equation (28) can now be rewritten to the final form equations to calculate A using $x_1 = x_0 - \Delta$ and $x_2 = x_0 + \Delta$:

$$\begin{aligned} A &= \frac{(x_0 - \Delta - x_0)^2(Z[x_1, y = y_0] - Z_0) + (x_0 + \Delta - x_0)^2(Z[x_2, y = y_0] - Z_0)}{(x_0 - \Delta - x_0)^4 + (x_0 + \Delta - x_0)^4} \\ &= \frac{\Delta^2(Z[x_1, y = y_0] - Z_0) + \Delta^2(Z[x_2, y = y_0] - Z_0)}{2\Delta^4} \\ &= \frac{(Z[x_1, y = y_0] - Z_0) + (Z[x_2, y = y_0] - Z_0)}{2\Delta^2} \\ &= \frac{Z[x_1, y = y_0] + Z[x_2, y = y_0] - 2Z_0}{2\Delta^2} \\ &= \frac{Z[x_0 - \Delta, y_0] + Z[x_0 + \Delta, y_0] - 2Z[x_0, y_0]}{2\Delta^2} \end{aligned} \quad (38)$$

Likewise, the final form equation to calculate D using $y_1 = y_0 - \Delta$ and $y_2 = y_0 + \Delta$:

$$D = \frac{Z[x_0, y_0 - \Delta] + Z[x_0, y_0 + \Delta] - 2Z[x_0, y_0]}{2\Delta^2} \quad (39)$$

It is best to keep Δ the same value for all free parameters. Not only does this simplify the algebra when it comes to calculating L but also it simplifies computer code.

4.6.2. The Effect of Quantizing the Free Parameters for Calculating L

Previously, we showed that sampling symmetrically on both sides of best fit of a one-dimensional case would result in perfect uncertainty calculations even if you were near the best fit location and not exactly at it. This is also true of the two-dimensional case utilizing all four diagonals around the best fit (to keep sampling symmetry): the uncertainties calculated via L with the following method will be correct even if your found best fit values are slightly offset from the true best fit values. Appendix E contains all the in-depth calculations that we will summarize in this section.

The equation for L sampling m values symmetrically on all sides in two dimensions reduces to the following equation, where x_0 and y_0 are our best fit values, i and j refer to the x and y dimensions, respectively, and Δ is a sampling step size that is the same for both dimensions:

$$L = \frac{\sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta]}{\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2} \quad (40)$$

Note here that this “all-sided” symmetric calculation is not dependent on the earlier one-dimensional values of A or D , as those canceled in this symmetric sense. Just as in the one-dimensional case, failing to do a symmetric sample around the best fit will result in an L value that is not accurate.

It can be shown (in Appendix E) that as long as x_0 and y_0 are close to the actual best fit values (x_a and y_a , where $x_a = x_0 + \delta_x$ and $y_a = y_0 + \delta_y$) then so long as $\delta_x < \Delta$ and $\delta_y < \Delta$, then the L found is the same value as if $x_0 = x_a$ and $y_0 = y_a$. It can also be shown that the value of m is irrelevant to calculating L (as long as $m \neq 0$), so for computational speed a value of $m = 1$ is the fastest with fewest calculations required. Clearly, when $i = 0$ or $j = 0$ (the horizontal and vertical cuts through the hypersurface that pass through the best fit location), then summand is also zero and does not contribute to the total sum in either the numerator nor denominator; therefore, of all the first ($m = 1$) nearest neighbors around the best fit locations, only the “diagonal” elements count toward the calculation for L .

Putting all this together for use in code and expanding the summation results in the final simple equation for L :

$$L = \frac{Z[x_0 + \Delta, y_0 + \Delta] + Z[x_0 - \Delta, y_0 - \Delta] - Z[x_0 - \Delta, y_0 + \Delta] - Z[x_0 + \Delta, y_0 - \Delta]}{4\Delta^2} \quad (41)$$

4.7. Example Calculation of Covariance Matrix

In this section we provide an example of how to calculate the covariance matrix using the methods presented earlier, that is, using nearest-neighbor symmetric sampling to calculate A , D , and L from equations (38), (39), and (41). Figure 2 shows two examples of the two-dimensional hypersurface around the best fit location to the nearest neighbor in Δ in each direction. The quadratic surface is for illustrative purposes only, where equation (23) was used, with the constants set as follows: $A = 296.12$, $D = 17037.34$, and $L = 337.02$. The actual minimum surface is when $x_a = 67.03441$ and $y_a = 4.10695$ giving $Z_a = 10.00000$. Figure 2 (left) shows a 3 by 3 grid showing the best fit value in the center (white cell) and the eight nearest-neighbor Z values on all sides for this perfect case, with $\Delta = 0.001$. To calculate A , we use equation (38) and the values in the green and center cells of the figure:

$$A = \frac{10.00029612 + 10.00029612 - 2 * 10.000000}{2 * 0.001^2} = 296.12 \quad (42)$$

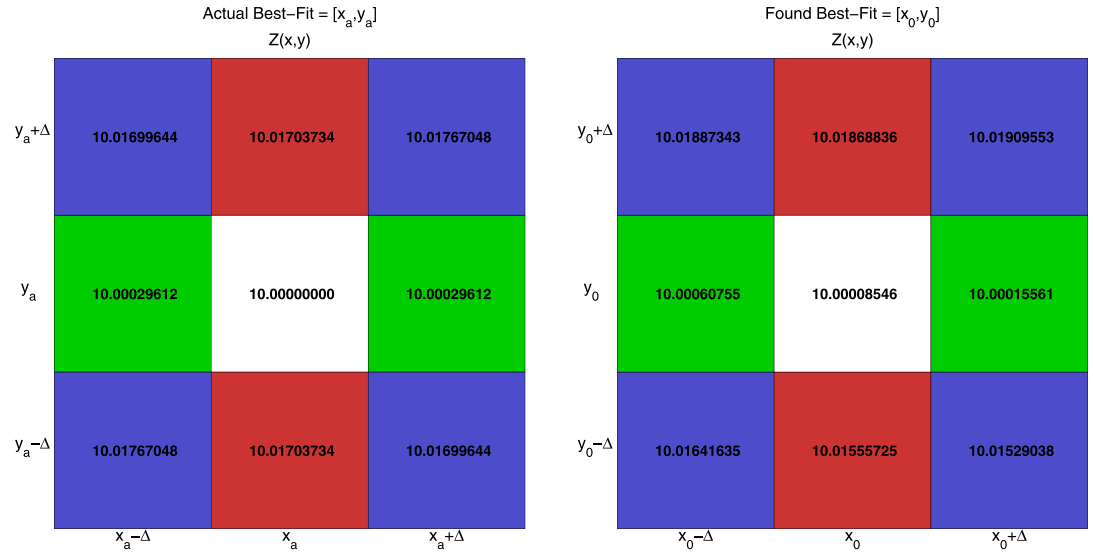


Figure 2. The two-dimensional hypersurface mapped at regular intervals of Δ to provide test values to show that the covariance matrix found from the true best fit vector and a near-best fit vector is the same. (left) The perfectly found “absolute” best fit. (right) The best fit to tolerance of code (truncated to three decimal places in this case), yet the covariance matrix is identical for both.

Likewise, to calculate D , we use equation (39) (essentially the same equation as for A but a different dimension) and the values in the red and center cells of the figure:

$$D = \frac{10.01703734 + 10.01703734 - 2 * 10.000000}{2 * 0.001^2} = 17037.34 \quad (43)$$

Finally, to calculate L , we use equation (41) and the values in the blue cells of the figure:

$$L = \frac{10.01767048 + 10.01767048 - 10.01699644 - 10.01699644}{4 * 0.001^2} = 337.02 \quad (44)$$

In the three equations above, A , D , and L all exactly match the known parameters the surface was generated from.

In Figure 2 (right) we show the same surface where a minimization routine has found the best fit values quantized to three decimal places to be $x_0 = 67.034$ and $y_0 = 4.107$ giving a $Z_0 = 10.00008546$. Clearly, this is not the perfect best fit as $Z_0 > Z_a$, $x_0 \neq x_a$, and $y_0 \neq y_a$ (but is practically close enough to be the same); instead, we have offsets in x and y of $\delta_x = 0.00041$ and $\delta_y = -0.00005$, respectively. This offset is clear in the nine Z values shown in the plot as the values are no longer symmetric. However, we can sample the eight nearest neighbors as before and do the same equations.

$$A = \frac{10.00060755 + 10.00015561 - 2 * 10.00008546}{2 * 0.001^2} = 296.12 \quad (45)$$

$$D = \frac{10.01868836 + 10.01555725 - 2 * 10.00008546}{2 * 0.001^2} = 17037.345 \approx 17037.34 \quad (46)$$

$$L = \frac{10.01909553 + 10.01641635 - 10.01887343 - 10.01529038}{4 * 0.001^2} = 337.0175 \approx 337.02 \quad (47)$$

In this case A is exactly as expected, while D and L are very close but not perfect. However, this is merely a rounding error to only printing values on the figure to eight decimal places; using the double-precision values for Z gave exactly 17037.34 and 337.02 for D and L .

This example shows that the method for finding A , D , and L in order to construct the covariance matrix is very simple and very accurate. Recall that the alpha matrix is made up from A and L ; the i^{th} diagonal element is A_i , and the nondiagonal element at (i, j) is $L_{ij}/2$. Using equation (25), we can substitute in the values for A , D , and $L/2$ in this two-dimensional case:

$$\alpha = \begin{bmatrix} 296.12 & 168.51 \\ 168.51 & 17037.34 \end{bmatrix} \quad (48)$$

The covariance matrix S^2 is just the matrix inverse of α :

$$S^2 = \begin{bmatrix} 0.003396 & -0.000034 \\ -0.000034 & 0.000059 \end{bmatrix} \quad (49)$$

Finally, the square root of the diagonal of the covariance matrix gives the uncertainty for each parameter. In this case, the uncertainty on x_0 is $\pm 0.058 (= \pm \sqrt{0.003396})$ and the uncertainty on y_0 is $\pm 0.008 (= \pm \sqrt{0.000059})$.

4.8. Example Pseudocode to Generate Uncertainties

So far, the examples shown have assumed that just two free parameters are to be fit; however, the same techniques apply for any number of free parameters as each element of the Hessian matrix never compares more than two parameters at once. The following pseudocode shows how to generate the covariance matrix, S^2 , and the standard deviations of the parameters, S , for any number of free parameters. Essentially, in each iteration it resets fp (the free parameter vector) to the best fit values, adjusts one or two elements of that vector by $\pm \Delta$, then works out the new Z for that vector. These are used to calculate the A or $L/2$ terms for each element of the α matrix, from which the covariance matrix can be calculated. This pseudocode assumes the first element of vector $p[1]$, not $p[0]$. This code could be rewritten in shorter form (e.g., instead of $Z1$, $Z2$, $Z3$, and $Z4$ being four separate scalars, it could be one array of size 4 that is looped to apply the correct $+$ or $- \Delta$) but is deliberately verbose here to make it clear, with spacing to highlight similar terms. The end result is a vector S of the standard deviations of the fitted parameters, such that the i^{th} element of p has a value of $p[i] \pm S[i]$.

```
/* Set up initial variables */
Delta = 0.001; /* Set up Delta with an appropriate value */
/* if fitting to 3 decimal places then Delta=0.001 is an obvious choice */
nu = {number of elements of vector p};
/* create 2-D array of size nu by nu to populate later */
alpha = zeros(nu,nu);
best = p; /* Best fit parameter vector */
Z0 = Z(best); /* Best fit Z */
/* Calculate A for the diagonal of the alpha matrix */
for i = 1:nu
    fp = best;
    fp(i) = fp(i) - Delta;
    Z1 = Z(fp);
    if Z1 <= Z0, then error('Was not at Global Minimum, A1');
    fp = best;
    fp(i) = fp(i) + Delta;
    Z2 = Z(fp);
    if Z2 <= Z0, then error('Was not at Global Minimum, A2');
    alpha(i,i) = ( Z1 + Z2 - 2 * Z0 ) / ( 2 * Delta * Delta );
end
/* Calculate L/2 for the off-diagonal terms of the alpha matrix */
for i = 1:(nu-1)
    for j = (i+1):nu
        fp = best;
        fp(i)=fp(i) + Delta;
        fp(j)=fp(j) + Delta;
        Z1 = Z(fp);
```

```

        if Z1 <= Z0, then error('Was not at Global Minimum, L1');
    fp = best;
    fp(i)=fp(i) - Delta;
    fp(j)=fp(j) - Delta;
    Z2 = Z(fp);
    if Z2 <= Z0, then error('Was not at Global Minimum, L2');
    fp = best;
    fp(i)=fp(i) - Delta;
    fp(j)=fp(j) + Delta;
    Z3 = Z(fp);
    if Z3 <= Z0, then error('Was not at Global Minimum, L3');
    fp = best;
    fp(i)=fp(i) + Delta;
    fp(j)=fp(j) - Delta;
    Z4 = Z(fp);
    if Z4 <= Z0, then error('Was not at Global Minimum, L4');
    alpha(i,j) = ( Z1 + Z2 - Z3 - Z4 ) / ( 8 * Delta * Delta);
    alpha(j,i) = alpha(i,j);
end
end
/* matrix inverse alpha to get covariance matrix */
/* Use numerical recipes if no built-in matrix inverse command */
S2 = inv(alpha);
/* sqrt of diagonal gives the standard deviation of each free parameter */
S = zeros(1,nu); /* create 1-D array to populate later */
for i = 1:nu
    if S2(i,i)<0
        error('Negative element in diagonal of covariance matrix!')
    end
    S(i) = sqrt( S2(i,i) ); /* calculate standard deviation of element i */
end

```

In the above code the line to initialize Z_0 may be set to use the output value from the fitting routine that gave the best fit, rather than requiring another call to your model. To then calculate all the Z_1 , Z_2 , Z_3 , and Z_4 values required in the pseudocode results in your model function being called a further $2\nu^2$ times ($\nu = \nu$ in the code). For instance, if $\nu = 4$, then only 32 calls to the model function are required in order to derive the uncertainties. This is computationally quicker than, for instance, using Monte Carlo techniques to estimate uncertainties that would require many more model function calls.

5. Discussion and Conclusion

The method of parameterizing the hypersurface in order to generate the Hessian matrix, α matrix, and then the covariance matrix works just as well if the best fit vector is found from a reduced chi-square running to full machine precision or if it truncates a little earlier. The specific code used to minimize the reduced chi-square is somewhat irrelevant. While some methods are faster than others (e.g., Nelder-Mead is quicker than differential evolution, DE, or adaptive simulated annealing, ASA, but is more dependent on the quality of the initial guess than DE or ASA are), they all perform the same function. Such routines typically return their best fit when a tolerance has been met or a maximum number of iterations has been reached. If the latter occurs, it is unlikely that the global minimum has been found, and it should be run again with a better guess value or its iteration limit increased. The tolerance can generally be tweaked by the user to adjust sensitivity and often refers to differences between the last iteration and current one as measured by the distance between vertices of the parameter vectors. However, this can lead to a few issues; failure to also tweak other minimization code specific parameters (i.e., the default Δ value) could create unrealistic uncertainties (i.e., negative covariance diagonals). For instance, if your model rounds the free parameter vector to three decimal places for use,

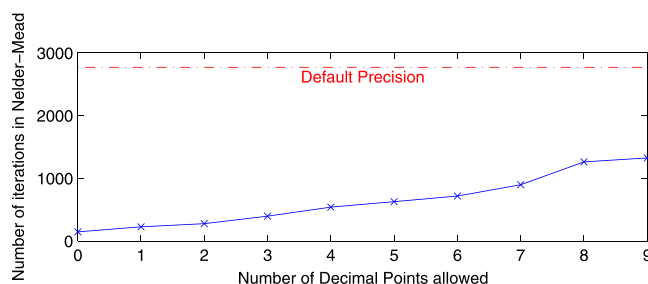


Figure 3. Code run time decreases with fewer decimal places per fit parameter. In the above experiment generating a best fit vector to three decimal places is 3 times faster than to eight decimal places and around 7 times faster than generating a best fit vector to machine precision.

a poorly chosen/forgotten Δ of 0.000001 would effectively be a Δ of zero such that the two nearest neighbors have $Z = Z_0$; thus, the A term becomes 0, which is unphysical.

If the minimization code exits too early (perhaps due to too few decimal places being used), there is a risk that the global minimum has yet to be found. However, the act of calculating the covariance matrix as presented in the pseudocode samples every dimension on both sides of the best fit, such that it is easy to add a line of code to check that returned neighbors of best fit do not have a smaller Z than the minimization code identified Z_0 .

Rather than adjust the “black box” minimizing code to truncate all iterations of the free parameters to n decimal places, it is much easier to have this rounding in the first line of your model function. When the black box code exits with its best fit parameters, they will still be to the full machine precision (as that is what was passed to the model function) as such the final output must be rounded in a likewise manner to achieve the actual parameters the model function used. While fewer decimal places speed up the number of iterations required, it must be set to a realistic number for the values you expect. For instance, space plasma densities are often plotted on a log scale in units of cm^{-3} as their range may cover many orders of magnitude. An ion density of 0.13 cm^{-3} to one decimal place would either fit to 0.1 cm^{-3} or 0.2 cm^{-3} ; both would have a model that would not match the data well. Worse still, fitting to zero decimal places would be either 0 cm^{-3} or 1 cm^{-3} which would never provide a good model match to the data or would crash the model. Alternatively, if the data are to be run in the Io torus at Jupiter, where densities are in the hundreds to thousands cm^{-3} , then fitting to zero decimal places would be fine, providing the other fit parameters (perhaps ones representing temperatures in eV and velocities in km/s) are also well represented to the same number of decimal places. For examining the plasma environments of Jupiter’s and Saturn’s magnetospheres with free parameters of the model being ion densities (cm^{-3}), temperatures (eV), and velocities (km/s), we found that the models that rounded the parameters to three decimal places were ample to match thermal plasma data from Galileo or Cassini, but two decimal places were too few to adequately match the data well in all environments.

If the minimization code used does have the option to provide covariance matrices (such as MPFIT [Markwardt, 2009]), then it is valid to use it providing the options are set up appropriately, such as ensuring two-sided sampling is carried out to generate the Hessian matrix (i.e., MPFIT uses single-sided techniques by default). However, it is always worth checking that the diagonals of the output covariance matrix are all positive values before use.

The trade-off from the method presented here to a black box code is that it will take substantially longer to carry out each fit, fine for an occasional fit, but a computational burden if there are many years of data. As shown in Figure 3 the speed increase in using fewer decimal places for each forward model fit within the model is significant; in this example test it was about 7 times faster to fit to three decimal places than to the predefined code tolerance. The fact that these truncated best fits are not the actual higher-precision best fit does not affect the calculated covariance matrix in any way, as shown in this paper.

In conclusion, five reasons to use the method of this paper to generate the uncertainty matrix are the following: (1) Any minimization code routine may be used to achieve your reduced chi-square fit. (2) It does not matter how complicated the model fit by the chi-square is. (3) By having your model round to a specified number of decimal places, it is computationally faster to find a fit, yet the covariance matrices generated are identical to those as if the minimization code ran longer to its full resolution. (4) The covariance matrix will

always have positive diagonal elements, ensuring that the standard deviations computed will neither crash your code nor generate complex numbers. (5) Generating the α matrix requires only a few simple lines of code, from which standard routines can be used to invert the matrix to get the covariance matrix.

A subsequent paper (in preparation) will describe the model these authors used to forward model Galileo PLS and Cassini CAPS data in order to provide thermal ion densities, temperatures, and flow speeds, along with the techniques that ensured that good fits were located (via limits, constraints, and data pruning). This is to be followed by two further papers showing global survey plasma parameters for each of these instruments.

Appendix A: One-Dimensional Variance Fits

Let us say that in one dimension x , the parameter space near the best fit can be approximated by a quadratic equation. This is reasonable as we have done a least squares fit. Constants in the following equations will now be capital letters (except Z that is not a constant as we have already defined that as the variable we have minimized), and we start with the normal quadratic equation:

$$Z[x] = Ax^2 + Bx + C \quad (\text{A1})$$

This is very easy to calculate numerically to find A , B , and C , but as we know the best fit is when $x = x_0$, giving $Z[x = x_0] = Z_0$ then:

$$C = Z_0 - Ax_0^2 - Bx_0 \quad (\text{A2})$$

Substituting this in the former equation gives:

$$Z[x] = A(x^2 - x_0^2) + B(x - x_0) + Z_0 \quad (\text{A3})$$

The first-order differential is a step toward the Hessian matrix:

$$\frac{\partial Z[x]}{\partial x} = 2Ax + B \quad (\text{A4})$$

At the best fit location we know we have a minimum at $x = x_0$, so the first-order differential equation (equation (A4)) must be zero, allowing us to relate B to A .

$$B = -2Ax_0 \quad (\text{A5})$$

This can now be substituted in to equation (A3) and rearranged:

$$Z[x] = A(x - x_0)^2 + Z_0 \quad (\text{A6})$$

Thus, of the original need to numerically calculate the best fits A , B , and C to parameterize the one-dimensional hypersurface well, we now only require finding A .

The Hessian matrix element is then the second-order differential:

$$\frac{\partial^2 Z[x]}{\partial x^2} = 2A \quad (\text{A7})$$

Now that we have parameterized the hypersurface well along dimension x , let us repeat with dimension y , following similar logic and repeating the process but now with constants D , E , and F :

$$Z[y] = Dy^2 + Ey + F \quad (\text{A8})$$

$$F = Z_0 - Dy_0^2 - Ey_0 \quad (\text{A9})$$

$$E = -2Dy_0 \quad (\text{A10})$$

$$Z[y] = D(y - y_0)^2 + Z_0 \quad (\text{A11})$$

$$\frac{\partial^2 Z[y]}{\partial y^2} = 2D \quad (\text{A12})$$

Appendix B: Two-Dimensional Covariance Fits

Working out the covariance requires parameterizing the fit so we can first partial differentiate on x then on y (or vice versa, same end result). So we have our 2-D hypersurface well in Z , with the best fit (and minimum Z at Z_0) at the center of the well at $x = x_0$ and $y = y_0$.

Let us assume that the surface can be described by the following six-term second-order polynomial equation. This is reasonable as Z is a best fit from a least squares method.

$$Z[x, y] = Gx^2 + Hx + I + Jy^2 + Ky + Lxy \quad (B1)$$

The first-order partial differentials for this are:

$$\frac{\partial Z[x, y]}{\partial x} = 2Gx + H + Ly \quad (B2)$$

$$\frac{\partial Z[x, y]}{\partial y} = 2Jy + K + Lx \quad (B3)$$

The second-order differential (from either of the first-order differentials) gives the term required for the Hessian matrix:

$$\frac{\partial^2 Z[x, y]}{\partial x \partial y} = L \quad (B4)$$

Therefore, the alpha value at the best fit location is:

$$\alpha_{x,y} = \alpha_{y,x} = \frac{L}{2} \quad (B5)$$

Currently, constants G to L are unknown; however, we may use the earlier 1-D fits to remove some of the unknowns in the 2-D case. Using equation (A6) and rearranging equation (B1) with $y = y_0$, we list both equations:

$$Z[x] = Ax^2 + (-2Ax_0)x + (Ax_0^2 + Z_0) \quad (B6)$$

$$Z[x] = Gx^2 + (Ly_0 + H)x + (Jy_0^2 + Ky_0 + I) \quad (B7)$$

We can then compare the coefficient terms of each power of x .

$$A = G \quad (B8)$$

$$-2Ax_0 = Ly_0 + H \quad (B9)$$

$$Ax_0^2 + Z_0 = Jy_0^2 + Ky_0 + I \quad (B10)$$

Likewise, using the equation (A11) and rearranging equation (B1) with $x = x_0$, we get:

$$Z[y] = Dy^2 + (-2Dy_0)y + (Dy_0^2 + Z_0) \quad (B11)$$

$$Z[y] = Jy^2 + (Lx_0 + K)y + (Gx_0^2 + Hx_0 + I) \quad (B12)$$

Again, we can then compare the coefficient terms of each power of y .

$$D = J \quad (B13)$$

$$-2Dy_0 = Lx_0 + K \quad (B14)$$

$$Dy_0^2 + Z_0 = Gx_0^2 + Hx_0 + I \quad (B15)$$

These give us G, H, J, K , and I in terms of known values and L :

$$G = A \quad (B16)$$

$$H = -2Ax_0 - Ly_0 \quad (B17)$$

$$J = D \quad (B18)$$

$$K = -2Dy_0 - Lx_0 \quad (B19)$$

$$Ax_0^2 + Z_0 = Dy_0^2 + (-2Dy_0 - Lx_0)y_0 + I \quad (B20)$$

$$\Rightarrow I = Ax_0^2 + Z_0 + Dy_0^2 + Lx_0y_0 \quad (B21)$$

This can now all be put in equation (B1):

$$Z[x, y] = Ax^2 + (-2Ax_0 - Ly_0)x + (Ax_0^2 + Z_0 + Dy_0^2 + Lx_0y_0) + Dy^2 + (-2Dy_0 - Lx_0)y + Lxy \quad (B22)$$

$$\Rightarrow Z[x, y] - Z_0 = A(x^2 - 2x_0x + x_0^2) + D(y^2 - 2y_0y + y_0^2) + L(xy - x_0y - xy_0 + x_0y_0) \quad (B23)$$

$$\Rightarrow Z[x, y] - Z_0 = A(x - x_0)^2 + D(y - y_0)^2 + L(x - x_0)(y - y_0) \quad (B24)$$

This last equation allows a full 2-D parameterization of the hypersurface well surface with only three unknowns (A , D , and L), which can then be fed back into equation (25) in order to calculate the α matrix then subsequently the covariance matrix.

Appendix C: Robustly Calculating the Constants A , D , and L

Consider we have n data points in a coordinate system (u, v) on which we wish to fit a straight line with a fixed gradient m that also passes through the origin, such that $v = mu$. We can use simple linear regression:

$$m = \frac{\sum_{i=1}^n (u_i - \bar{u})(v_i - \bar{v})}{\sum_{i=1}^n (u_i - \bar{u})^2} \quad (C1)$$

Here a bar above a letter represents an average (i.e., $\bar{u} = \frac{1}{n} \sum_{i=1}^n u_i$), and this has the property that the line will go through the center of mass point $(\bar{u}, \bar{v})$. But it must also pass through the origin, and this is a problem for equations where both sides are positive, like equation (A6), such that the average cannot be zero.

This is easily solved by doubling the number of data points used for the regression; we still use the first n then repeat those n points but multiplying both u_i and v_i terms by -1 . This will now make it symmetric around the origin.

$$m = \frac{(\sum_{i=1}^n (u_i - \bar{u})(v_i - \bar{v})) + (\sum_{i=1}^n (-u_i - \bar{u})(-v_i - \bar{v}))}{(\sum_{i=1}^n (u_i - \bar{u})^2) + (\sum_{i=1}^n (-u_i - \bar{u})^2)} \quad (C2)$$

However, as means are now zero, this can be simplified to:

$$\begin{aligned} m &= \frac{(\sum_{i=1}^n (u_i)(v_i)) + (\sum_{i=1}^n (-u_i)(-v_i))}{(\sum_{i=1}^n (u_i)^2) + (\sum_{i=1}^n (-u_i)^2)} \\ \Rightarrow m &= \frac{(\sum_{i=1}^n (u_i)(v_i)) + (\sum_{i=1}^n (u_i)(v_i))}{(\sum_{i=1}^n (u_i)^2) + (\sum_{i=1}^n (u_i)^2)} \\ \Rightarrow m &= \frac{2 \sum_{i=1}^n u_i v_i}{2 \sum_{i=1}^n u_i^2} \\ \Rightarrow m &= \frac{\sum_{i=1}^n u_i v_i}{\sum_{i=1}^n u_i^2} \end{aligned} \quad (C3)$$

These equations guarantee that your analytical equation is a minimum at the best fit value.

By making substitutions and using our measured data, we can easily find values for our constants. $m = A$ when:

$$u_i = (x_i - x_0)^2 \quad (C4)$$

$$v_i = Z[x_i, y = y_0] - Z_0 \quad (C5)$$

$$\Rightarrow A = \frac{\sum_{i=1}^n (x_i - x_0)^2 (Z[x_i, y = y_0] - Z_0)}{\sum_{i=1}^n (x_i - x_0)^4} \quad (C6)$$

$m = D$ when:

$$u_i = (y_i - y_0)^2 \quad (C7)$$

$$v_i = Z[x = x_0, y_i] - Z_0 \quad (C8)$$

$$\Rightarrow D = \frac{\sum_{i=1}^n (y_i - y_0)^2 (Z[x = x_0, y_i] - Z_0)}{\sum_{i=1}^n (y_i - y_0)^4} \quad (C9)$$

Then using A and D , $m = L$ when:

$$u_i = (x_i - x_0)(y_i - y_0) \quad (C10)$$

$$v_i = Z[x_i, y_i] - Z_0 - A(x_i - x_0)^2 - D(y_i - y_0)^2 \quad (C11)$$

$$\Rightarrow L = \frac{\sum_{i=1}^n (x_i - x_0)(y_i - y_0)(Z[x_i, y_i] - Z_0 - A(x_i - x_0)^2 - D(y_i - y_0)^2)}{\sum_{i=1}^n (x_i - x_0)^2 (y_i - y_0)^2} \quad (C12)$$

Appendix D: The 1-D Case When Not Exactly at the Best Fit Location

In this paper we are finding A based on the best fit value of $x(= x_0)$ giving the best fit ($Z[x_0] = Z_0$) where x_0 probably has some offset from the true minimum due to the minimization routine you used reaching its tolerance and stopping (and similarly for D for $y = y_0$). We have solved for:

$$Z[x] = A(x - x_0)^2 + Z[x_0] \quad (D1)$$

However, let us assume that the actual analytical best fit (with no rounding) is at x_a such that:

$$Z[x] = a(x - x_a)^2 + Z[x_a] \quad (D2)$$

where $Z[x_a]$ is the true best chi-square ($Z[x_a] < Z[x_0]$) and a is the actual polynomial term; how does that compare with A ?

Using this, we can find $Z[x_0]$ and substitute in equation (D1).

$$\begin{aligned} Z[x_0] &= a(x_0 - x_a)^2 + Z[x_a] \\ \Rightarrow Z[x] &= A(x - x_0)^2 + a(x_0 - x_a)^2 + Z[x_a] \end{aligned} \quad (D3)$$

Let us say our best fit is offset from the true best fit by value δ such that $x_a - \delta = x_0$; it then follows that we can remove the x_a term:

$$\begin{aligned} Z[x] &= A(x - x_0)^2 + a(x_a - \delta - x_a)^2 + Z[x_a] \\ \Rightarrow Z[x] &= A(x - x_0)^2 + a\delta^2 + Z[x_a] \end{aligned} \quad (D4)$$

To calculate A , we use the linear regression where by making substitutions and using our measured data, we can easily find values for our constants. From equation (C6):

$$A = \frac{\sum_{i=1}^n u_i v_i}{\sum_{i=1}^n u_i^2} \quad (D5)$$

where

$$\begin{aligned} u_i &= (x_i - x_0)^2 \\ v_i &= Z[x_i] - Z[x_0] \\ &= (a(x_i - x_a)^2 + Z[x_a]) - (a(x_0 - x_a)^2 + Z[x_a]) \\ &= (a(x_i - x_0 - \delta)^2 + Z[x_a]) - (a\delta^2 + Z[x_a]) \\ &= a((x_i - x_0 - \delta)^2 - \delta^2) \\ &= a(x_i - x_0)(x_i - x_0 - 2\delta) \end{aligned} \quad (D6)$$

Putting this all together in the linear regression term, we can calculate the ratio A/a :

$$A = \frac{\sum_{i=1}^n (x_i - x_0)^2 a (x_i - x_0) (x_i - x_0 - 2\delta)}{\sum_{i=1}^n (x_i - x_0)^4}$$

$$\Rightarrow \frac{A}{a} = \frac{\sum_{i=1}^n (x_i - x_0)^3 (x_i - x_0 - 2\delta)}{\sum_{i=1}^n (x_i - x_0)^4} \quad (D7)$$

Let us assume we take equidistant samples of hypersurface around the best fit location where $x_i = x_0 + i\Delta$ (where $\Delta > \delta$) and i ranges from a to b :

$$\begin{aligned} \frac{A}{a} &= \frac{\sum_{i=a}^b (x_0 + i\Delta - x_0)^3 (x_0 + i\Delta - x_0 - 2\delta)}{\sum_{i=a}^b (x_0 + i\Delta - x_0)^4} \\ &= \frac{\sum_{i=a}^b (i\Delta)^3 (i\Delta - 2\delta)}{\sum_{i=a}^b (i\Delta)^4} \\ &= \frac{\sum_{i=a}^b (i\Delta)^4 - 2(i\Delta)^3 \delta}{\sum_{i=a}^b (i\Delta)^4} \\ &= \frac{(\sum_{i=a}^b (i\Delta)^4) - (\sum_{i=a}^b 2(i\Delta)^3 \delta)}{\sum_{i=a}^b (i\Delta)^4} \\ &= \frac{(\sum_{i=a}^b (i\Delta)^4) - (2\delta \sum_{i=a}^b (i\Delta)^3)}{\sum_{i=a}^b (i\Delta)^4} \\ &= 1 - \frac{2\delta \sum_{i=a}^b i^3}{\Delta \sum_{i=a}^b i^4} \quad (D8) \end{aligned}$$

For a symmetric two-sided calculation the summation is over i from $a = -n$ to $b = n$ (giving $2n + 1$ samples). Let us consider just the $\sum i^3$ term:

$$\begin{aligned} \sum_{i=-n}^n i^3 &= \left(\sum_{i=-n}^{-1} i^3 \right) + \left(\sum_{i=0}^0 i^3 \right) + \left(\sum_{i=1}^n i^3 \right) \\ &= \left(\sum_{i=1}^n (-i)^3 \right) + 0 + \left(\sum_{i=1}^n i^3 \right) \\ &= - \left(\sum_{i=1}^n i^3 \right) + \left(\sum_{i=1}^n i^3 \right) \\ &= 0 \quad (D9) \end{aligned}$$

Clearly, this would make the second term on the right-hand side of equation (D7) zero, so $A = a$. As long as we do the linear regression by sampling hypersurface symmetrically on both sides of the best fit position the δ term cancels out, it does not matter for the uncertainty calculation that our best fit is not the true best fit.

If, however, we just use a one-sided sampling in equation (D7) where i ranges from 1 to n , we can then use some summation identities:

$$\begin{aligned} \frac{A}{a} &= 1 - \frac{\delta}{\Delta} \frac{2 \sum_{i=1}^n i^3}{\sum_{i=1}^n i^4} \\ &= 1 - \frac{\delta}{\Delta} \frac{15n(n+1)}{(2n+1)(3n^2+3n-1)} \quad (D10) \end{aligned}$$

Clearly, for one-sided sampling, for nonzero δ it must follow that $A \neq a$, with the difference getting smaller as n increases.

Appendix E: The 2-D Case When Not Exactly at the Best Fit Location

Previously, equation (30) was used to calculate L by sampling n points near the best fit location $Z[x_i, y_i]$ where $1 \leq i \leq n$. Appendix D previously showed that using a symmetric two-sided sampling method for one-dimensional fits would provide accurate uncertainties even if the best fit value was not the true value (provided $\delta < \Delta$). Would an equivalent “symmetric four-sided” fit work for the two-dimensional case? Here we must split up the sum over i from 1 to n in two summations over i and j , both going from $-m$ to $+m$, where $x_i = x_0 + i\Delta$ and $y_j = y_0 + j\Delta$ (keeping Δ the same for both x and y) such that the equation for L becomes:

$$\begin{aligned} L &= \frac{\sum_{i=-m}^m \sum_{j=-m}^m (x_i - x_0)(y_j - y_0)(Z[x_0 + i\Delta, y_0 + j\Delta] - Z[x_0, y_0] - A(x_i - x_0)^2 - D(y_j - y_0)^2)}{\sum_{i=-m}^m \sum_{j=-m}^m (x_i - x_0)^2 (y_j - y_0)^2} \\ &= \frac{\sum_{i=-m}^m \sum_{j=-m}^m (i\Delta)(j\Delta)(Z[x_0 + i\Delta, y_0 + j\Delta] - Z[x_0, y_0] - A(i\Delta)^2 - D(j\Delta)^2)}{\sum_{i=-m}^m \sum_{j=-m}^m (i\Delta)^2 (j\Delta)^2} \\ &= \frac{\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m ij(Z[x_0 + i\Delta, y_0 + j\Delta] - Z[x_0, y_0] - A(i\Delta)^2 - D(j\Delta)^2)}{\Delta^4 \sum_{i=-m}^m \sum_{j=-m}^m i^2 j^2} \\ &= \frac{\sum_{i=-m}^m \sum_{j=-m}^m ij(Z[x_0 + i\Delta, y_0 + j\Delta] - Z[x_0, y_0] - A(i\Delta)^2 - D(j\Delta)^2)}{\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2 j^2} \end{aligned} \quad (E1)$$

When i or j equals zero, both the summands in the numerator and denominator go to zero and as such do not increase the summation; therefore, there is no reason to include these in the sampling. So for practical purposes the summation should be from $-m$ to -1 then 1 to m but never $m = 0$ (this was also true in the one-dimensional case in Appendix D).

The numerator can be split into four separate terms and rearranged using standard summation identities, e.g., $\sum_i C f(i) = C \sum_i f(i)$ and $\sum_i \sum_j f(i, j) = \sum_j \sum_i f(i, j)$:

$$+ \left(\sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta] \right) - \left(Z[x_0, y_0] \sum_{i=-m}^m i \sum_{j=-m}^m j \right) - \left(A \Delta^2 \sum_{i=-m}^m i^3 \sum_{j=-m}^m j \right) - \left(D \Delta^2 \sum_{j=-m}^m j^3 \sum_{i=-m}^m i \right) \quad (E2)$$

It is now clear that the final three numerator terms are all zero as $\sum_{i=-m}^m i = \sum_{j=-m}^m j = 0$. This greatly simplifies the equation and also removed any dependence on A and D :

$$L = \frac{\sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta]}{\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2 j^2} \quad (E3)$$

As before, this L is calculated on the found best fits (x_0 and y_0) and not the actual best fits ($x_a = x_0 + \delta_x$ and $y_a = y_0 + \delta_y$), where the offsets from the true value are δ_x and δ_y , respectively. From earlier we know the actual parameterized equation and found that the equation for Z is the following:

$$Z[x_i, y_j] = a(x_i - x_a)^2 + d(y_j - y_a)^2 + l(x_i - x_a)(y_j - y_a) + Z[x_a, y_a] \quad (E4)$$

where l is the true L and a and d are the true one-dimensional values, where Appendix D confirmed that for symmetric samples $a = A$ and $d = D$:

$$Z[x_i, y_j] = A(x_i - x_a)^2 + D(y_j - y_a)^2 + l(x_i - x_a)(y_j - y_a) + Z[x_a, y_a] \quad (E5)$$

From here we can substitute in for $x_i, y_j, x_a,$ and y_a , all in terms of x_0 or y_0 :

$$Z[x_0 + i\Delta, y_0 + j\Delta] = A(i\Delta - \delta_x)^2 + D(j\Delta - \delta_y)^2 + l(i\Delta - \delta_x)(j\Delta - \delta_y) + Z[x_a, y_a] \quad (E6)$$

This term can be substituted into the numerator on the right-hand side of equation (E3):

$$\sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta] = \sum_{i=-m}^m \sum_{j=-m}^m ij (A(i\Delta - \delta_x)^2 + D(j\Delta - \delta_y)^2 + l(i\Delta - \delta_x)(j\Delta - \delta_y) + Z[x_a, y_a]) \quad (E7)$$

As before, the right-hand side of this can be split out into separate terms and rearranged using standard summation identities:

$$\begin{aligned}
 \sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta] &= + \left(A \sum_{i=-m}^m i(i\Delta - \delta_x)^2 \sum_{j=-m}^m j \right) + \left(D \sum_{j=-m}^m j(j\Delta - \delta_y)^2 \sum_{i=-m}^m i \right) \\
 &+ \left(I \sum_{i=-m}^m \sum_{j=-m}^m ij(i\Delta - \delta_x)(j\Delta - \delta_y) \right) + \left(Z[x_a, y_a] \sum_{i=-m}^m i \sum_{j=-m}^m j \right) \\
 &= 0 + 0 + \left(\sum_{i=-m}^m \sum_{j=-m}^m ijI(i\Delta - \delta_x)(j\Delta - \delta_y) \right) + 0 \\
 &= I \sum_{i=-m}^m \sum_{j=-m}^m ij(i\Delta - \delta_x)(j\Delta - \delta_y) \tag{E8}
 \end{aligned}$$

Three of the four terms above go to zero, again because $\sum_{i=-m}^m i = \sum_{j=-m}^m j = 0$. The final stage is to expand the parentheses in this equation and again separate out the summations:

$$\begin{aligned}
 \sum_{i=-m}^m \sum_{j=-m}^m ijZ[x_0 + i\Delta, y_0 + j\Delta] &= I \sum_{i=-m}^m \sum_{j=-m}^m (i^2j^2\Delta^2 - i^2j\Delta\delta_y - ij^2\Delta\delta_x + ij\delta_x\delta_y) \\
 &= +I \left(\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2 \right) - I \left(\Delta\delta_y \sum_{i=-m}^m i^2 \sum_{j=-m}^m j \right) \\
 &\quad - I \left(\Delta\delta_x \sum_{j=-m}^m j^2 \sum_{i=-m}^m i \right) + I \left(\delta_x\delta_y \sum_{i=-m}^m i \sum_{j=-m}^m j \right) \\
 &= I \left(\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2 \right) - 0 - 0 + 0 \\
 &= I\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2 \tag{E9}
 \end{aligned}$$

This term can be substituted into the numerator on the right-hand side of equation (E3):

$$L = \frac{I\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2}{\Delta^2 \sum_{i=-m}^m \sum_{j=-m}^m i^2j^2} = I \tag{E10}$$

Therefore, providing you sample symmetrically on both x and y , the L found is accurate for x_0 and y_0 even if $x_0 \neq x_a$ and $y_0 \neq y_a$, providing $\delta < \Delta$. Also, this equation is independent of m , as such there is no computational reason to use anything more than $m = 1$.

Finally, we may return to equation (E3), substitute for $m = 1$, and expand the summations:

$$L = \frac{Z[x_0 + \Delta, y_0 + \Delta] + Z[x_0 - \Delta, y_0 - \Delta] - Z[x_0 - \Delta, y_0 + \Delta] - Z[x_0 + \Delta, y_0 - \Delta]}{4\Delta^2} \tag{E11}$$

To summarize, we have established that as long as we calculate L symmetrically on all four corners, the δ_x , δ_y , and Δ values all cancel out, even when $x_0 \neq x_a$ and $y_0 \neq y_a$, and no prior knowledge of the one-dimensional coefficients (A and D) is required.

Acknowledgments

This study applies to any data; no particular data were used in this study. I thank Fran Bagenal for her support and fruitful comments and Graeme Wilkin for the many useful conversations that led to this paper.

References

- Bame, S. J., D. J. McComas, B. L. Barraclough, J. L. Phillips, K. J. Sofaly, J. C. Chavez, B. E. Goldstein, and R. K. Sakurai (1992), The Ulysses solar wind plasma experiment, *Astron. Astrophys.*, 92, 237–265.
- Baumjohann, W., and R. A. Treumann (1999), *Basic Space Plasma Physics*, Imperial College Press.
- Bridge, H. S., J. W. Belcher, R. J. Butler, A. J. Lazarus, A. M. Mavretic, J. D. Sullivan, G. L. Siscoe, and V. M. Vasyliunas (1977), The plasma experiment on the 1977 Voyager mission, *Space Sci. Rev.*, 21, 259–287, doi:10.1007/BF00211542.
- Beverington, P. R., and D. K. Robinson (2003), *Data Reduction and Error Analysis for the Physical Sciences*, 3rd ed., McGraw-Hill, Univ. of California.

- Boyd, T., and J. J. Sanderson (1969), *Plasma Dynamics (Applications of Mathematics)*, Thomas Nelson and Sons Ltd.
- Coates, A. J., F. J. Crary, G. R. Lewis, D. T. Young, J. H. Waite, and E. C. Sittler (2007), Discovery of heavy negative ions in Titan's ionosphere, *Geophys. Res. Lett.*, **34**, L22103, doi:10.1029/2007GL030978.
- Elrod, M. K., W.-L. Tseng, R. J. Wilson, and R. E. Johnson (2012), Seasonal variations in Saturn's plasma between the main rings and Enceladus, *J. Geophys. Res.*, **117**, A163207, doi:10.1029/2011JA017332.
- Frank, L. A., K. L. Ackerson, J. A. Lee, M. R. English, and G. L. Pickett (1992), The plasma instrumentation for the Galileo Mission, *Space Sci. Rev.*, **60**, 283–304, doi:10.1007/BF00216858.
- Lewis, G. R., N. André, C. S. Arridge, A. J. Coates, L. K. Gilbert, D. R. Linder, and A. M. Rymer (2008), Derivation of density and temperature from the Cassini Huygens CAPS electron spectrometer, *Planet. Space Sci.*, **56**, 901–912, doi:10.1016/j.pss.2007.12.017.
- Markwardt, C. B. (2009), Non-linear least squares fitting in IDL with MPFIT, in *Proceedings of Astronomical Data Analysis Software and Systems XVIII, Quebec, Canada, ASP Conference Series*, vol. 411, edited by D. Bohlender, P. Dowler, and D. Durand, pp. 251–254, Astron. Soc. of the Pacific, San Francisco, Calif.
- McComas, D., et al. (2008), The Solar Wind Around Pluto (SWAP) instrument aboard new horizons, *Space Sci. Rev.*, **140**, 261–313, doi:10.1007/s11214-007-9205-3.
- McComas, D. J., et al. (2013), The Jovian Auroral Distributions Experiment (JADE) on the Juno Mission to Jupiter, *Space Sci. Rev.*, doi:10.1007/s11214-013-9990-9.
- Nicolaou, G., D. J. McComas, F. Bagenal, and H. A. Elliott (2014), Properties of plasma ions in the distant Jovian magnetosheath using Solar Wind Around Pluto data on New Horizons, *J. Geophys. Res. Space Physics*, **119**, 3463–3479, doi:10.1002/2013JA019665.
- Paschmann, G., and P. W. Daly (1998), *Analysis Methods for Multi-Spacecraft Data*, vol. 1, ISSI Scientific Reports Series SR-001, ESA/ISSI.
- Thomsen, M. F., D. B. Reisenfeld, D. M. Delapp, R. L. Tokar, D. T. Young, F. J. Crary, E. C. Sittler, M. A. McGraw, and J. D. Williams (2010), Survey of ion plasma parameters in Saturn's magnetosphere, *J. Geophys. Res.*, **115**, A10220, doi:10.1029/2010JA015267.
- Thomsen, M. F., R. J. Wilson, R. L. Tokar, D. B. Reisenfeld, and C. M. Jackman (2013), Cassini/CAPS observations of duskside tail dynamics at Saturn, *J. Geophys. Res. Space Physics*, **118**, 5767–5781, doi:10.1002/jgra.50552.
- Wilson, R. J., R. L. Tokar, M. G. Henderson, T. W. Hill, M. F. Thomsen, and D. H. Pontius (2008), Cassini plasma spectrometer thermal ion measurements in Saturn's inner magnetosphere, *J. Geophys. Res.*, **113**, A12218, doi:10.1029/2008JA013486.
- Wilson, R. J., et al. (2012a), *PDS User's Guide for Cassini Plasma Spectrometer (CAPS)*, Planetary Data System. [Available at <http://ppi.pds.nasa.gov/>.]
- Wilson, R. J., P. A. Delamere, F. Bagenal, and A. Masters (2012b), Kelvin-Helmholtz instability at Saturn's magnetopause: Cassini ion data analysis, *J. Geophys. Res.*, **117**, A03212, doi:10.1029/2011JA016723.
- Wilson, R. J., F. Bagenal, P. A. Delamere, M. Desroche, B. L. Fleshman, and V. Dols (2013), Evidence from radial velocity measurements of a global electric field in Saturn's inner magnetosphere, *J. Geophys. Res. Space Physics*, **118**, 2122–2132, doi:10.1002/jgra.50251.
- Wolfe, J. H., J. D. Mihalov, H. R. Collard, D. D. McKibbin, L. A. Frank, and D. S. Intriligator (1974), Pioneer 10 observations of the solar wind interaction with Jupiter, *J. Geophys. Res.*, **79**, 3489–3500, doi:10.1029/JA079i025p03489.
- Young, D. T., et al. (2004), Cassini plasma spectrometer investigation, *Space Sci. Rev.*, **114**, 1–112, doi:10.1007/s11214-004-1406-4.