Parameter Estimation in Differential Equations: A Numerical Study of Shooting Methods

Franz Hamilton
Faculty Advisor: Dr. Timothy Sauer

January 5, 2011

Abstract

Differential equation modeling is central to applications of mathematics to science and engineering. When a particular system of equations is used in an application, it is often important to determine unknown parameters. We compare the traditional shooting method to versions of multiple shooting methods in chaotic systems with noise added and conduct numerical experiments as to the reliability and accuracy of both methods.

1 Introduction

Modeling by differential equations is important in applications. It is used in a variety of fields to accurately depict the physical state of a system. Accurately describing the system allows for future behavior to be predicted. This is particularly useful when looking at applications in chemistry, engineering, physics and other disciplines.

A differential equation model is designed to capture as much of the system behavior as possible. These differential equations often have parameters. Some parameters may be calculated from first principles or known from literature. However, it is extremely common that other parameters need to be fitted from observed data.

In this article, we will restrict ourselves to problems that arise in chaotic dynamics. In chaotic systems, the trajectories are sensitive to small changes in the initial conditions as well as the parameters, making parameter estimation even more difficult. Due to these sensitivities, there is always the danger of diverging from the correct trajectory.

For our study we assume a system of n coupled ordinary differential equations in variables $x_1, \ldots, x_n$. In real world problems, it is rare to be able to simultaneously measure

the values of all the dynamical variables at all time points and sometimes these dynamical variables are not known at all. For this reason we will assume in our analysis the value of only one variable, say x_1, is available for observation. To each equation we add a white noise term. From the observed x_1 variable we want to determine the parameters in the systems as well as the initial conditions of $x_1, \ldots, x_n$. We will investigate the use of shooting methods in fitting parameters with a particular interest in the method of multiple shooting.

The traditional shooting method (details for example can be found in [9]) can be used to solve boundary value problems in ordinary differential equations. Given a set of boundary values, the initial condition of the system is guessed and the resulting solution is compared with the boundary values. If the boundary values are not met, then the initial condition is adjusted, according to some predetermined iterative protocol, until the solution converges to the correct boundary values. A bisection or Newton-type iteration is often used for this purpose.

For parameter estimation problems, applying the traditional shooting method requires a slightly different approach. Now we are not only trying to adjust initial conditions but also parameters to match the boundary value. Due to the added number of unknowns, we may need to observe the trajectory at more points to aid in the parameter fitting.

The single shooting method applied to parameter estimation uses a single initial condition to produce a trajectory that attempts to fit the noisy data points. This can be particularly problematic when considering the dynamics of chaotic systems, which are by definition sensitive to initial data. The idea of multiple shooting was introduced by von Domselaar and Hemker [1], further developed by Bock [4] and Baake et al. [2] and discussed in an interesting review by Voss et al. [5]. Multiple shooting takes the traditional boundary value problem and considers it as a multi-point boundary value problem. This allows for multiple initial values to be produced along the desired time interval, lessening the impact of trajectory divergence for chaotic dynamics. Once we apply the multiple shooting approach to our problem, we see that considering it as a multi-point boundary value problem simplifies the process of data fitting, particularly when considering chaotic data.

Adjusting initial conditions and parameters to fit either one or multiple trajectories to data leads to a large nonlinear least squares problem. It becomes a nonlinear least squares problem due to the complicated dependence of trajectories on parameters. Therefore, iterative methods are needed to solve this problem and in our instance we will be using the Gauss-Newton and Levenberg-Marquardt iterative method.

In Section 2 of the paper we will discuss parameter fitting with both a single shooting and multiple shooting approach. We will demonstrate how to use the Gauss-Newton

method to solve the nonlinear least squares problem in both single and multiple shooting instances. In Section 3 we will discuss parameter fitting for two example chaotic systems, giving the appropriate results. We will address the use of both Gauss-Newton as well as Levenberg-Marquardt. In Section 4 we will make some concluding remarks about parameter fitting for the two systems and about the overall effectiveness of single shooting and multiple shooting.

2 Shooting Methods for Parameter Estimation

We will reconstruct the value of the parameters as well as the respective initial values for each differential equation in the system. We initially generate noisy data points by setting all parameters and initial values in the system to what we define as the system's correct values. In attempting to reconstruct parameters from noisy data we investigate the use of a shooting method, which uses a Gauss-Newton or Levenberg-Marquardt iterative method to solve the nonlinear least squares problem that arises. We will compare two types of shooting methods: single shooting and multiple shooting.

2.1 Reconstruction with Single Shooting

Figure 1 shows the single shooting method schematically. The solution emanates from an initial value, or *shooting node*. For the single shooting method, one *solution segment* is generated. We would like to minimize the length of the vertical line segments which represent the difference between the values of the solution segment and the noisy data points represented by the red circles. We will assume throughout that the noise is Gaussian and therefore it will be convenient to minimize the differences in the least squares sense. The goal of a shooting method is to alter the values at the shooting nodes so that the values of the solution segment approximate the noisy data points as closely as possible.

We assume a parametrized system of the form

$$\begin{aligned} \dot{x} &= f_1(x, y, z, \vec{p}) \\ \dot{y} &= f_2(x, y, z, \vec{p}) \\ \dot{z} &= f_3(x, y, z, \vec{p}) \end{aligned}$$

where only one of the three equations is observable. Without loss of generality, we assume only the x variable is observable. Furthermore, the observations are noisy. In particular the initial conditions of $\dot{y}$ and $\dot{z}$ are not known at all. Denote by $d_0, d_1, \ldots, d_m$ the x coordinate of our noisy data points and denote by $\vec{p}$ a vector of unknown parameters.

Define the residual vector, r, as the following:

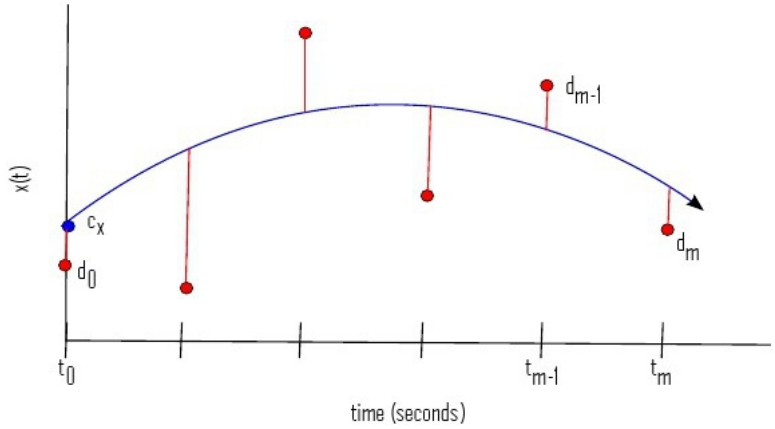

Figure 1 – Schematic of the single shooting method. The red circles represent noisy data points generated from a differential equation. The blue circle represents the shooting node and the blue curve is the solution segment resulting from the shooting node. The red lines represent the difference between the data points and the values of the solution segment.

$$r = \begin{bmatrix} x(t_0, \vec{c}, \vec{p}; t_1) - d_1 \\ x(t_0, \vec{c}, \vec{p}; t_2) - d_2 \\ \vdots \\ x(t_0, \vec{c}, \vec{p}; t_m) - d_m \\ c_x - d_0 \end{bmatrix} \tag{1}$$

Let $(x(t), y(t), z(t)) = (x(t_0, \vec{c}, \vec{p}; t), y(t_0, \vec{c}, \vec{p}; t), z(t_0, \vec{c}, \vec{p}; t))$ denote the solution of the system (calculated without noise) with initial value vector $\vec{c}$ at time t_0 with parameter vector $\vec{p}$. In our examples, $\vec{c}$ and $\vec{p}$ will each have three entries (c_x, c_y, c_z and p_1, p_2, p_3 respectively) since the two systems we will be considering both have three parameters and three initial values (one for each differential equation in the system). Since we are only observing the x variable, we consider only $x(t_0, \vec{c}, \vec{p}; t)$ in the formulation of r. The differences between the values of the solution of the system and the data points are represented in the upper part of r and the last entry in r represents the difference between the value of the shooting node c_x and the data point. Pictorially, r will contain the lengths of the red lines in Figure 1.

The goal is to solve

$$min_{\vec{c},\vec{p}} ||r||_2^2 \tag{2}$$

Since r depends non-linearly on $\vec{c}, \vec{p}$, we will solve (2) using the Gauss-Newton method first. The Gauss-Newton method [9] is an iterative method used to solve nonlinear least

squares problems. Implementing it will minimize the difference between the solution segment and the noisy data points.

We will use the Gauss-Newton method to choose parameters $\vec{p}$ and initial values $\vec{c}$ to minimize r. It is an iterative method whose i^{th} step is

$$\begin{aligned} J(w^i)^T J(w^i) v^i &= -J(w^i)^T r(w^i) \\ w^{i+1} &= w^i + v^i \end{aligned} \tag{3}$$

We will denote by w^i a vector of guesses at step i for the parameters and initial values we are trying to reconstruct and define $J = Dr$ to be the Jacobian of the residual vector, r. Equation (3) is the heart of the Gauss-Newton method. We solve for v^i which essentially is an adjustment vector for the values in w^i. We add v^i to w^i and generate the new iteration, w^{i+1}. To generate the next iteration we put the values in w^{i+1} into our system and generate a new solution via the shooting method. From this we re-define r and J and proceed to equation (3) to solve for w^{i+2}. This process is repeated until convergence to within 10^{-6} accuracy is reached.

As (3) shows, the Gauss-Newton iteration requires calculation of J, the Jacobian matrix of the vector r. Calculating the Jacobian requires extra, auxiliary differential equations known as variational equations [6] to be solved along with the original system. We use a numerical ODE solver, with appropriate step size, to follow the variational equations. Given a differential equation $\dot{u} = f(u, \vec{p})$ where $u \in \mathbb{R}^n$ and $\vec{p} \in \mathbb{R}^k$ define $F(\vec{c}, \vec{p}; T) = u(T)$ where $u(t)$ is the solution to the following:

$$\begin{aligned} \dot{u} &= f(\vec{c}, \vec{p}) \\ u(0) &= \vec{c} \end{aligned}$$

$F(\vec{c}, \vec{p}; t)$ satisfies the differential equation, therefore $\frac{d}{dt}F(\vec{c}, \vec{p}; t) = f(F(\vec{c}, \vec{p}; t), \vec{p})$ and $F(\vec{c}, \vec{p}; 0) = \vec{c}$. Using the chain rule:

$$\tfrac{d}{dt}[DF(\vec{c}, \vec{p}; t)] = Df \cdot D \begin{bmatrix} F \\ \vec{p} \end{bmatrix}$$

where D represents differentiation with respect to $\vec{c}$ and $\vec{p}$. This can then be written as the following:

$$\begin{bmatrix} \frac{\partial \dot{F}}{\partial c_x} & \frac{\partial \dot{F}}{\partial c_y} & \frac{\partial \dot{F}}{\partial c_z} & \frac{\partial \dot{F}}{\partial p_1} & \frac{\partial \dot{F}}{\partial p_2} & \frac{\partial \dot{F}}{\partial p_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial u_1} & \frac{\partial f}{\partial u_2} & \frac{\partial f}{\partial u_3} & \frac{\partial f}{\partial p_1} & \frac{\partial f}{\partial p_2} & \frac{\partial f}{\partial p_3} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial F}{\partial c_x} & \frac{\partial F}{\partial c_y} & \frac{\partial F}{\partial c_z} & \frac{\partial F}{\partial p_1} & \frac{\partial F}{\partial p_2} & \frac{\partial F}{\partial p_3} \\ 0 & \dots & 0 & 1 & & \\ \vdots & \ddots & & & \ddots & \\ 0 & & 0 & & & 1 \end{bmatrix}$$

Defining $z = \frac{\partial F}{\partial \vec{c}}$ and $q = \frac{\partial F}{\partial \vec{p}}$ we can write the above as:

$$\begin{bmatrix} \dot{z}_1 & \dot{z}_2 & \dot{z}_3 & \dot{q}_1 & \dot{q}_2 & \dot{q}_3 \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial \vec{u}} & \frac{\partial f}{\partial \vec{p}} \end{bmatrix} \cdot \begin{bmatrix} z_1 & z_2 & z_3 & q_1 & q_2 & q_3 \\ 0 & \dots & 0 & 1 & & \\ \vdots & \ddots & & & \ddots & \\ 0 & & 0 & & & 1 \end{bmatrix}$$

With the above we are able to construct the Jacobian matrix J needed for Gauss-Newton. Differentiating (1) with respect to the 6 variables x, y, z, p_1, p_2, p_3 yields the Jacobian matrix

$$\begin{bmatrix} z_1(t_1) & z_2(t_1) & z_3(t_1) & q_1(t_1) & q_2(t_1) & q_3(t_1) \\ z_1(t_2) & z_2(t_2) & z_3(t_2) & q_1(t_2) & q_2(t_2) & q_3(t_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_1(t_m) & z_2(t_m) & z_3(t_m) & q_1(t_m) & q_2(t_m) & q_3(t_m) \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which we can write as:

$$\begin{bmatrix} \frac{\partial F}{\partial c_x}(t_1) & \frac{\partial F}{\partial c_y}(t_1) & \frac{\partial F}{\partial c_z}(t_1) & \frac{\partial F}{\partial p_1}(t_1) & \frac{\partial F}{\partial p_2}(t_1) & \frac{\partial F}{\partial p_3}(t_1) \\ \frac{\partial F}{\partial c_x}(t_2) & \frac{\partial F}{\partial c_y}(t_2) & \frac{\partial F}{\partial c_z}(t_2) & \frac{\partial F}{\partial p_1}(t_2) & \frac{\partial F}{\partial p_2}(t_2) & \frac{\partial F}{\partial p_3}(t_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial F}{\partial c_x}(t_m) & \frac{\partial F}{\partial c_y}(t_m) & \frac{\partial F}{\partial c_z}(t_m) & \frac{\partial F}{\partial p_1}(t_m) & \frac{\partial F}{\partial p_2}(t_m) & \frac{\partial F}{\partial p_3}(t_m) \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where c_x, c_y, c_z represent our three initial values and p_1, p_2, p_3 represent our three parameters.

2.2 Multiple Shooting Approach

The single shooting approach is often numerically unstable. Slightly incorrect parameters can cause the estimated trajectory to diverge from the correct trajectory after a small amount of time. This is clearly seen when considering chaotic systems where systems can be extremely dynamic [10].

Multiple shooting uses multiple shooting nodes, rather than a single shooting node, along the time interval in an attempt to match the trajectory of the data points more

accurately. The trade-off for extra accuracy in the trajectories is several extra initial conditions added to the list of parameters to be determined. Figure 2 shows the idea behind multiple shooting. The blue circles represent our k shooting nodes and each shooting node has a solution segment to the next shooting node that spans $m+1$ data points. At each shooting node, our vector of initial values (denoted $\vec{c}_j$) will change to reflect the current shooting node. Once again $\vec{c}_j$ is a vector of initial values for our system and since we have three differential equations in our system, $\vec{c}_j$ will have three entries (c_{xj}, c_{yj}, c_{zj}). For k shooting nodes we will have k initial value vectors $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k$.

Denote by $d_0, d_1, \ldots, d_{km}$ the x coordinate of our noisy data points. For some data

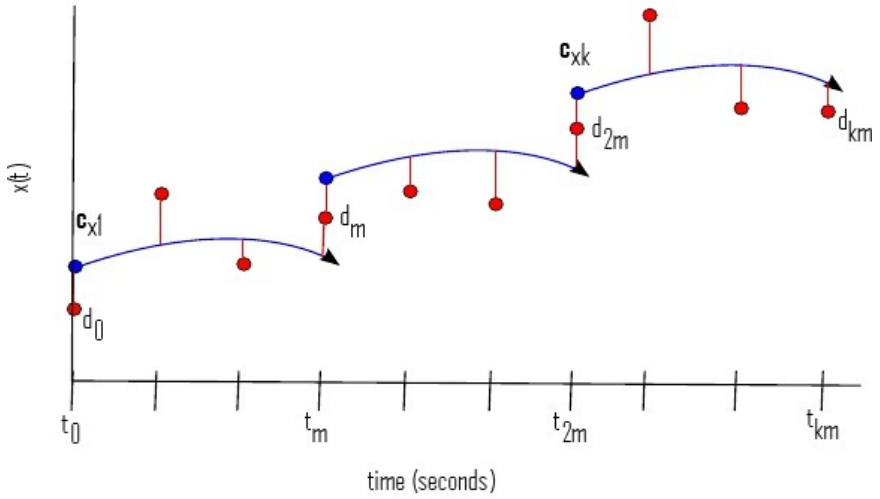

Figure 2 – Schematic of the multiple shooting method. The red circles represent noisy data points generated from a differential equation. The blue circles represent the shooting nodes and the blue curves are the solution segments resulting from the shooting nodes. The red lines represent difference between the data points and the values of the solution segments. In this schematic $k = 3$ and $m = 3$.

points, there are two differences (represented by the red lines) that need to be minimized: the difference between the value of the shooting node c_{xj} (since we are observing the x variable) and the data point and the difference between the values of the solution segment from the previous shooting node and the data point. Once again we use Gauss-Newton

to minimize these differences. For multiple shooting define the residual vector

$$r = \begin{bmatrix} x(t_0, \vec{c}_1, \vec{p}; t_1) - d_1 \\ x(t_0, \vec{c}_1, \vec{p}; t_2) - d_2 \\ \vdots \\ x(t_0, \vec{c}_1, \vec{p}; t_m) - d_m \\ x(t_m, \vec{c}_2, \vec{p}; t_{m+1}) - d_{m+1} \\ x(t_m, \vec{c}_2, \vec{p}; t_{m+2}) - d_{m+2} \\ \vdots \\ x(t_m, \vec{c}_2, \vec{p}; t_{2m}) - d_{2m} \\ x(t_{2m}, \vec{c}_3, \vec{p}; t_{2m+1}) - d_{2m+1} \\ x(t_{2m}, \vec{c}_3, \vec{p}; t_{2m+2}) - d_{2m+2} \\ \vdots \\ x(t_{(k-1)m}, \vec{c}_k, \vec{p}; t_{(k-1)m+1}) - d_{(k-1)m+1} \\ \vdots \\ x(t_{(k-1)m}, \vec{c}_k, \vec{p}; t_{km}) - d_{km} \\ c_{x1} - d_0 \\ c_{x2} - d_m \\ \vdots \\ c_{xk} - d_{(k-1)m} \end{bmatrix}$$

Once again we assume only the x variable is observable and $\vec{c_j}$ is an initial value vector and $\vec{p}$ is a parameter value vector. In our examples, $\vec{c_j}$ and $\vec{p}$ will each have three entries (c_{xj}, c_{yj}, c_{zj} and p_1, p_2, p_3 respectively) since the two systems we will be considering both have three parameters and three initial values (one for each differential equation in the system). The differences between the values of the solution segments and data points are represented in the upper part of r and the differences between the values of the shooting nodes and the data points are represented in the lower part of r where c_{xj} is the j^{th} shooting node. Pictorially, r will contain the lengths of the red lines in Figure 2. The Jacobian J is computed in a similar fashion to the discussion in Section 2.1.

3 Results

We compare the performance of single shooting to multiple shooting in two different systems: the Rössler system and the Lorenz system. For both systems we will reconstruct the three initial values in addition to the three parameters, though it is the reconstruction of the parameters that we are mainly interested in. Due to our use of the Gauss-Newton or Levenberg-Marquardt iterative method, there exists the possibility that our iteration could diverge. Often this is the case when our initial guesses are too distant from the actual value we are trying to reconstruct. Due to this, we evaluate success in two ways:

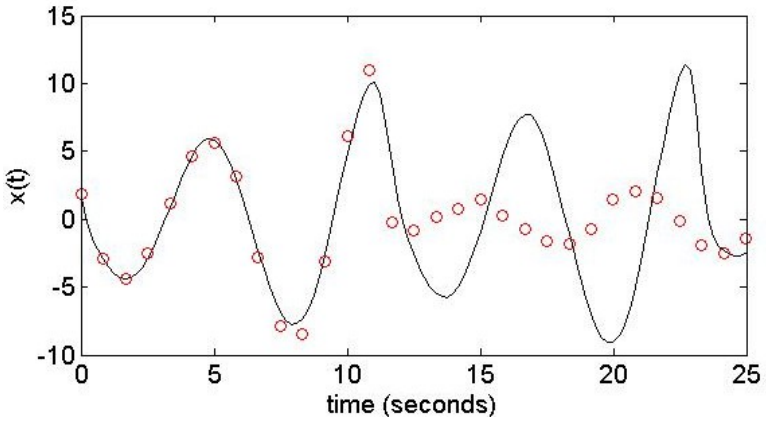

Figure 3 – Example of the Rössler system with noise. Curve represents true solution while circles represent noisy data points at a noise level of 0.6

First, whether or not the iteration converges to the neighborhood of correct values, and second, if they converge, the accuracy in finding the original values. We run 100 trials, find the number of times our method converges to a finite parameter set and then find the average and standard deviation of our results for the cases that converge. Of course, the results are highly dependent on the nearness of the initial guesses (initial guesses used to start Gauss-Newton and Levenberg-Marquardt are found in the captions of Figure 4 and Figure 5 for the Rössler and Lorenz systems respectively). However they serve the purpose of comparing the success of the algorithm among different number of shooting nodes.

3.1 Gauss-Newton

We first consider the results of the shooting methods when Gauss-Newton is used. When fitting the three parameters in both chaotic systems we needed to use an under-relaxation version of Gauss-Newton to aid convergence. The difference between Gauss-Newton and the under-relaxation version is that the second line of equation (3) becomes:

$$w^{i+1} = w^i + \alpha v^i$$

where $0 < \alpha \leq 1$. This is needed due to the complexity of fitting three parameters. When fitting one parameter, with the other two parameters fixed, then the under-relaxation version of Gauss-Newton is not needed. For the Rössler system we used $\alpha = 0.55$ and for the Lorenz system we used $\alpha = 0.1$. These values for α were chosen to be more or less optimal after numerous trial runs for both chaotic systems.

We initially apply our method to the Rössler equations [8], a system of three differ-

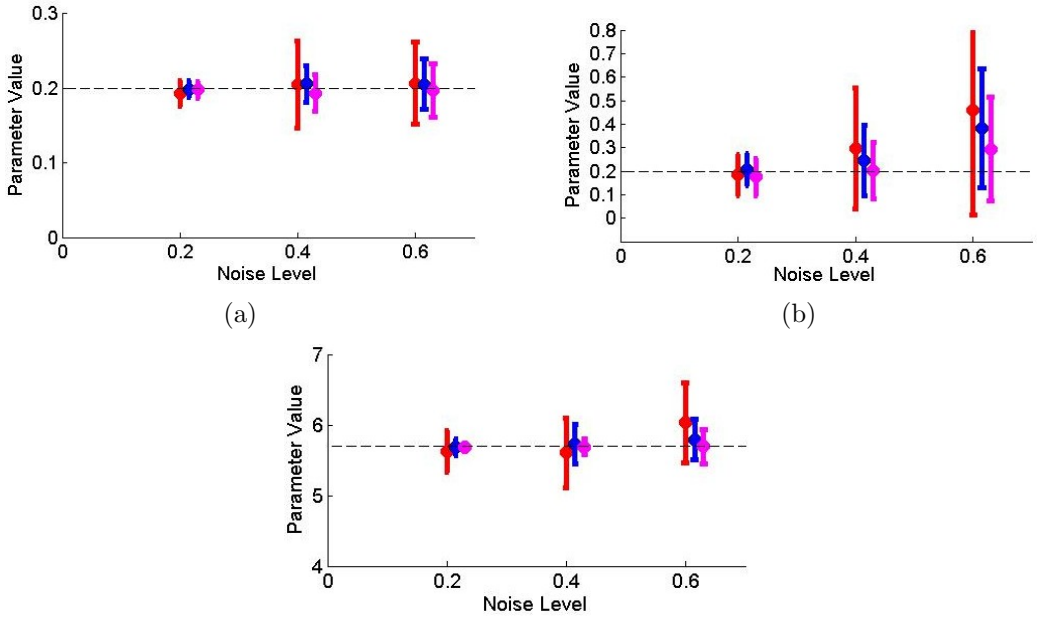

Figure 4 – Reconstruction of Rössler parameters a, b, c. We run 100 trials, find the number of times our method converges and then find the average and standard deviation (shown as error bars) of our results for the cases that converge. The correct parameters we are trying to reconstruct are $a = 0.2$, $b = 0.2$ and $c = 5.7$ and we start our method at $a = 0.05$, $b = 0.05$ and $c = 4$. We also attempt to to reconstruct initial values $c_{x_1} = 4.7820$, $c_{y_1} = -8.7335$ and $c_{z_1} = 0.0666$ and we start our method at $c_{x_1} = 4$, $c_{y_1} = -8$ and $c_{z_1} = 0$. Dotted line indicates true value and marker color indicates type of shooting used: red- single shooting, blue- multiple shooting with 3 shooting nodes, purple- multiple shooting with 6 shooting nodes. Results are slightly displaced horizontally for readability. (a) Approximation of parameter a. Correct value is $a = 0.2$. (b) Correct value $b = 0.2$. (c) Correct value $c = 5.7$.

ential equations with three parameters and three initial values of the form:

$$\begin{aligned} \dot{x} &= y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{4}$$

We will look at noisy data points generated by this system over a time interval of $[0, 25]$. Noise will be added by attaching a white noise term to the Rössler equations:

$$\begin{aligned} \dot{x} &= y - z + n\xi_1(t) \\ \dot{y} &= x + ay + n\xi_2(t) \\ \dot{z} &= b + z(x - c) + n\xi_3(t) \end{aligned} \tag{5}$$

where $\xi_i(t)$ represents standard white noise of unit variance, and n is the noise level to be chosen later. Figure 3 shows the influence of noise on the system. We will observe the x differential equation and apply the shooting methods described above. Our goal here is to reconstruct the three Rossler parameters: a, b, c.

Table 1 and Figure 4 show the results of our reconstruction of the three Rössler

Convergence Table for Parameters a, b, c		
Number of Shooting Nodes	Noise Level	Covergence (%)
1	0.2	56
	0.4	33
	0.6	29
3	0.2	74
	0.4	42
	0.6	36
6	0.2	92
	0.4	69
	0.6	44

Table 1 – The table above summarizes the probability of convergence of our trials for the Rössler equations with noise. The number of times the Gauss-Newton method converged, out of 100 trials, is listed.

parameters. The first thing we notice is that as intuition suggests, the error bars increase in size as the noise level increases for the various shooting types. In terms of performance advantage, we see that multiple shooting presents an advantage over single shooting. The first indicator comes from looking at the convergence percentage. An increase in the number of shooting nodes causes the number of times our Gauss-Newton iteration diverges to decrease. So in terms of convergence percentage, six shooting nodes performs the best, three shooting nodes the second best and one shooting node the worst. In looking at the accuracy of the parameter reconstruction, we see that multiple shooting once again performs better than single shooting. Increasing the number of shooting nodes improves the accuracy of the average values for the parameters as well as decreases the size of the error bars.

The Lorenz equations [3] form a system of three differential equations that have an importance in climate and weather predictions. A version of the Lorenz equations with white noise is

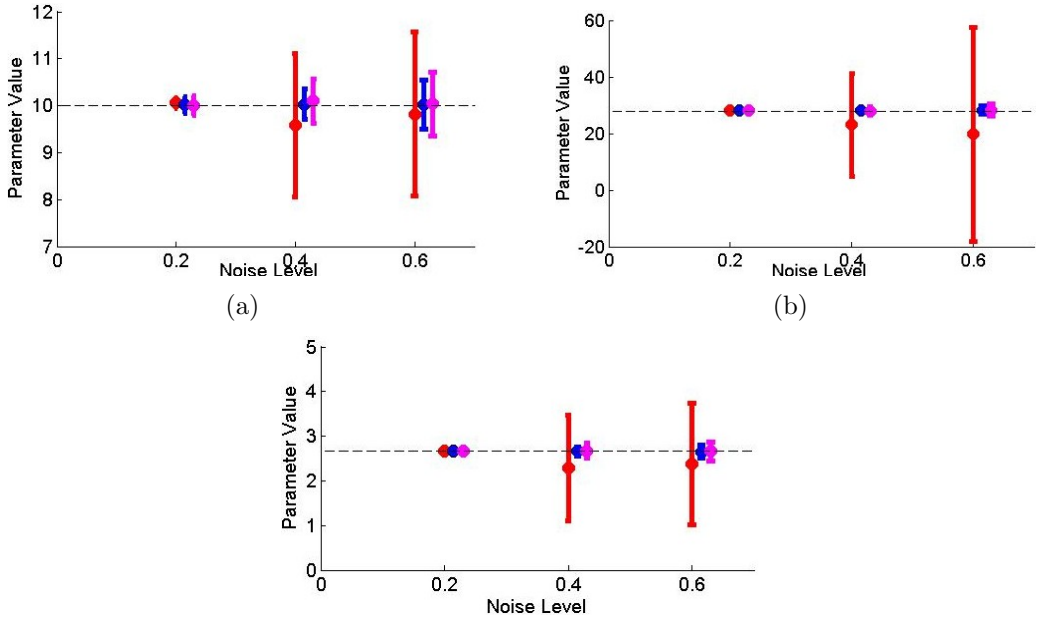

Figure 5 – Reconstruction of Lorenz parameters s, r, b. We run 100 trials, find the number of times our method converges and then find the average and standard deviation (shown as error bars) of our results for the cases that converge. The correct parameters we are trying to reconstruct are $s = 10$, $r = 28$ and $b = 8/3$ and we start our method at $s = 8$, $r = 26$ and $b = 5/3$. We also reconstruct initial values $c_{x_1} = -8.0969$, $c_{y_1} = -6.9108$ and $c_{z_1} = 28.0485$ and start our method at $c_{x_1} = -7.2$, $c_{y_1} = -6$ and $c_{z_1} = 27$. Dotted line indicates true value and marker color indicates type of shooting used: red- single shooting, blue- multiple shooting with 3 shooting nodes, purple- multiple shooting with 6 shooting nodes. Results are slightly displaced horizontally for readability. (a) Approximation of parameter s. Correct value is $s = 10$. (b) Correct value is $r = 28$. (c) Correct value is $b = 8/3$.

$$\begin{aligned} \dot{x} &= -sx + sy + n\xi_1(t) \\ \dot{y} &= -xz + rx - y + n\xi_2(t) \\ \dot{z} &= xy - bz + n\xi_3(t) \end{aligned} \tag{6}$$

where s, r, and b are the unknown parameters within the system. The noisy data points are generated from the x equation over a time interval of [0,5].

Table 2 and Figure 5 show the results of our reconstruction of the three Lorenz parameters. Once again, we see that multiple shooting offers an advantage over single shooting. Looking at the convergence percentage, an increase over one shooting node causes the number of times our Gauss-Newton iteration diverges to decrease. However, unlike in the Rössler example there is no observable advantage of using six shooting nodes over three shooting nodes. Looking at the parameter reconstruction, we yet again see that

Convergence Table for Parameters s, r, b		
Number of Shooting Nodes	Noise Level	Covergence (%)
1	0.2	39
	0.4	24
	0.6	21
3	0.2	100
	0.4	100
	0.6	100
6	0.2	100
	0.4	99
	0.6	94

Table 2 – The table above summarizes the probability of convergence of our trials for the Lorenz equations with noise. The number of times the Gauss-Newton method converged, out of 100 trials, is listed.

multiple shooting offers a performance advantage in the form of more accurate average values and smaller standard deviations. Amongst multiple shooting methods, we see that there is no improvement in the accuracy of the reconstruction when using six shooting nodes over three shooting nodes. One would assume that six shooting nodes would lead to a better performance than three shooting nodes. However, it could be that for the solution to the Lorenz system that we were looking at three shooting nodes was optimal.

3.2 Levenberg-Marquardt

Levenberg-Marquardt [7] is an alternative to the Gauss-Newton method that is often utilized when the Gauss-Newton approximation results in an ill-conditioned problem. Just like Gauss-Newton, it minimizes the difference between the value of the solution segment and the noisy data points.

We use the Levenberg-Marquardt method to choose parameters $\vec{p}$ and initial values $\vec{c}$ to minimize the residual vector r. It is an iterative method whose i^{th} step is

$$\begin{aligned} (J(w^i)^T J(w^i) + \lambda diag(J(w^i)^T J(w^i)))v^i &= -J(w^i)^T r(w^i) \\ w^{i+1} &= w^i + v^i \end{aligned} \tag{7}$$

Here λ is a damping parameter to be chosen. Notice that if $\lambda = 0$ then we have (3), the equation for the Gauss-Newton method. The setup for our problem is the same as defined in Section 2, we just replace equation (3) with equation (7).

Use of Levenberg-Marquardt though requires determining what value λ should be. λ

could be fixed to a value, but what is more likely to be done is that λ is adjusted based on the success of the Levenberg-Marquardt iteration. At each iteration, the norm of the current residual vector is compared with the norm of the previous iteration's residual vector. If the norm of the current residual vector is less than the norm of the previous residual vector, than you reduce λ by a factor of k where k is a number to be determined. Otherwise, λ is kept the same.

A large value of λ assures convergence of the Levenberg-Marquardt iteration but less accurate parameter reconstruction, while a smaller value of λ trades likelihood of convergence for accuracy in the parameter reconstruction. We took the approach to start λ large and progressively make it smaller at a rate determined by k. For both systems we set $\lambda = 10000$ and $k = 2$. The values for λ and k were chosen to be more or less optimal after numerous trial runs for both chaotic systems.

We find that use of Levenberg-Marquardt more or less generates the same plots as Figure 4 and Figure 5. This makes sense since as λ gets smaller and smaller Levenberg-Marquardt essentially becomes Gauss-Newton. Where you can see the real difference is in the convergence of the shooting methods. For the Lorenz system, use of Levenberg-Marquardt yields convergence every time for both three and six shooting nodes and significantly increases the convergence percentage of the single shooting method as compared to when Gauss-Newton was used. For the Rössler system, use of Levenberg-Marquardt increased the convergence percentage for both single shooting and three shooting nodes as compared to the Gauss-Newton results. We did not notice an increase in convergence percentage for six shooting nodes. It is still unclear why this is the case.

4 Conclusion

Our goal was to investigate the accurate reconstruction of unknown parameters in two systems of differential equations: the Rössler system and the Lorenz system. Given noisy data points observed from the x differential equation and the actual equations themselves, we utilized shooting methods to reconstruct the values of each system's three parameters at various noise levels.

We looked at a single shooting method and multiple shooting methods (consisting of three and six shooting nodes), both of which used the Gauss-Newton or Levenberg-Marquardt method to solve the nonlinear least squares problem that arose. We wanted to compare the performance of both shooting methods in the reconstruction of the three Lorenz and three Rossler parameters. This comparison was done in regards to two sets of results: the number of times our Gauss-Newton or Levenberg-Marquardt method converged in the shooting method and in the case of convergence the accuracy of the shooting

method in finding the correct values.

We first examined the shooting methods with Gauss-Newton. In the case of the Rössler system, Table 1 and Figure 4 illustrate the performance advantage of using multiple shooting over single shooting. The convergence percentage of our method increases with the use of multiple shooting. In terms of the actual reconstructed values, multiple shooting delivers more accurate average values for the parameters with a smaller standard deviation. Amongst multiple shooting methods, we see an increase in convergence percentage as well as an increase in the accuracy of the reconstructed values when we jump from use of three shooting nodes to six shooting nodes. The parameter reconstruction in the Lorenz system yields somewhat similar results to what we found in the Rössler system. Table 2 and Figure 5 show that multiple shooting once again has a performance advantage over single shooting in terms of convergence percentage and accuracy of the reconstructed values. We also considered the shooting methods with Levenberg-Marquardt instead of Gauss-Newton. What we found was that the parameter reconstruction was comparable to that done with Gauss-Newton. The main advantage in using Levenberg-Marquardt is that for the most part it increases the likelihood of convergence for the shooting methods.

We were able to conduct a numerical study on shooting methods in fitting parameters to chaotic data. In particular, we examined parameter fitting in both the Rössler and Lorenz systems. In both systems, we found that methods of multiple shooting perform better than the traditional single shooting method.

References

[1] B. von Domselaar, P. Hemker. Nonlinear parameter estimation in initial value problems. (1975).

[2] E. Baake, M. Baake, H. Bock, K. Briggs. Fitting ordinary differential equations to chaotic data. Phys. Rev. **45**A, 5524 (1992).

[3] E. Lorenz. Deterministic nonperiodic flow. Journal of Atmospheric Sceinces, **20**, (1962).

[4] H. Bock. Recent advances in parameter identification for ordinary differential equations. Progress in Scientific Computing, **2**, 95-121 (1983).

[5] H. Voss, J. Timmer, J. Kurths. Nonlinear dynamical system identification from uncertain and indirect measurements. Int. J. Bif. Chaos, **14**, 1905-1933 (2002).

[6] J. Riley, M. Bennett, E. McCormick. Numerical integration of variational equations. Mathematics of Computation, **21**, 12-17 (1967).

[7] M. Heath. Scientific Computing An Introductory Survey. McGraw-Hill (2002).

[8] O. Rossler. An equation for continuous chaos. Physics Letters, **57**A, 5 (1976).

[9] T. Sauer. Numerical Analysis. Pearson Addison-Wesley (2006).

[10] W. Horbelt, T. Muller, J. Timmer, W. Melzer, K. Winkler. Analysis of nonlinear differenatial equations: parameter estimation and model selection. Medical Data Analysis, 152-158 (2000).