# Locating Error in Dynamical Systems

Carter Koehler

Advisor: Matthew Plumlee

03/19/2022

**Abstract**

# 1 Notation

- $d$: Number of dimensions of state vector

- $N$: Number of time points

- $t_i$: Time at point $i$

- $x_i^* \in \mathbb{R}^d$, $i = 1, \ldots, N$: State vector of the "true" model at time $t_i$.

- $x_i(\eta) \in \mathbb{R}^d$, $i = 1, \ldots, N$: State vector of the approximate model at time $t_i$.

- $f_0 : \mathbb{R}^d \to \mathbb{R}^d$: Basic model, which is known in principle.

- $m^*(x, \eta) : \mathbb{R}^d \to \mathbb{R}^d$: Unmodelled terms in a system, which we will call *supplemental terms*. In principle not known.

- $m_i(x)$: Basis functions for approximating $m^*$. Write $m(x) = \sum_{i=1}^p \eta_i m_i(x)$.

- $f^*(x) = f_0(x) + m^*(x)$: True full model, including

- $f(x, \eta) = f_0(x) + m(x, \eta)$: Predicted model

- $p$: Number of basis functions we search over when looking for unmodelled terms.

- $\eta \in \mathbb{R}^{p \times d}$: Parameters associated with the functions in our search basis.

# 2 Introduction

## 2.1 Direction

Dynamical Systems models have been used to great effect in a variety of different fields, and though analysis of these models is well-worn, some issues remain when it comes to comparing predictions by the model to real-world data.

First and foremost is the issue of parameter estimation. Though frameworks exist for estimating the constant parameters of a differential equation, such frameworks run into a variety of problems, including...

If we look for an approximation $f$ that approximates $f^*$ optimally under some loss $L(f^*, f)$ (further details on the loss later), we can identify a few main sources of loss that we expect to contribute

The problem we will focus on primarily is the issue of determining the sources of error in dynamical systems. If a proposed model gives us a certain quantity of error, how do we know what causes that error? Is it the result of unmodelled effects in the right-hand sides of our equations? If so, can we learn those effects given a search basis $\{m_i\}_{i=1}^p$? How much error is coming from discretizing the time steps, and can we estimate that and try to account for it when making predictions? And last, how much of the error we see can we attribute to "true" noise or measurement error? Many attempts have been made, with more or less success, to address these issues, but few have attempted to look at these problems in a unified way and attempted to tackle the problem of uncertainty in dynamical systems as a single problem.

These are very large issues, and we won't try to solve them all in this limited research span, but we think that approaching them as a unified problem will provide useful ways of thinking as the field progresses.

The dynamical systems we will look at concern the general system

$$\dot{x}(t, \eta) = f(x; \eta).$$

# 3 Background

## 3.1 Parameter Estimation

The first problem we are interested in is estimating $\eta$. That is, given $f(x; \eta)$ and a set of observations $y_1, \ldots, y_N$, can we find $\eta$ to within a reasonable

error margin?

Gelman, et al. (1996) solved a particular instance of this problem, related to pharmacokinetics, using a hierarchical Bayesian approach. They began with a Bayesian prior describing the parameters of their model, under which they numerically solved the dynamical systems and then used the resulting error distributions to update their paramter distributions using fairly standard Monte Carlo methods. One advantage this gave them in their particular case was that it allowed them to jointly construct a population model and individual models by exploiting the hierarchical nature of Bayesian models.[1] However, for a general problems their method has a litany of problems, including that there are not always obvious choices of prior, and the computation of posterior is not always tractable.

More recently, the Bayesian approach for parameter estimation has fallen out of favor and been largely replaced by collocation methods, like those used in Ramsay, et al. (2007), which treats the problem as a nested optimization, which considers the solution of the system as a sum of basis functions. They optimize in their outer loop over the parameters, checking for fit against observed values, while their inner loop finds the coefficients of the basis functions which give the optimally-fitting function to the system for a given set of parameters.[2] This generally gives good estimates to the parameters in question, but it gives almost no indication of how good they are.

Even more recently, Levine and Stuart (2021) uses techniques from machine learning to solve not only the problem of parameter estimation, but also the issue of discovering model error, which is broadly defined as errors resulting from a misspecified or underspecified model. For example, if one sets $\dot{x} \propto x$, but in reality $\dot{x} \propto x^2$, there is no way to recover the "real" model through simple parameter estimation. Their methods are remarkably powerful and show additionally that starting with an educated guess of the true model can cause RNNs to perform much better than they would on their own. However, there are a variety of questions that go unanswered. For example, how much uncertainty can we expect in these estimations in general? And how can we ensure that the parameters we discover, as well as our starting model, are uniquely meaningful, and that we don't just absorb the "known" model in our discovered model?

## 3.2 Uncertainty Quantification in Numerical Solutions of ODEs

In addition to finding the parameters of the dynamical systems we are interested in, we would also like to know the uncertainty associated with our estimates. This is mainly for the purpose of determining when the solution is "found" to within a specified tolerance. Techniques exist for determining worst-case error, but in practice the error estimates never reach below desired thresholds, leading to the need for other heuristics to determine when enough data has been collected, among other controlling factors.[citation needed]However, precise knowledge of uncertainty can also used to better estimate the solution and parameters of the system when working under a Bayesian framework, as proposed in Chkrebtii, et al. (2016). However, this naturally requires the use of Bayesian methods, including Markov Chain Monte Carlo methods, which have a variety of problems that can arise, not the least of which that they are not guaranteed to converge, and that it can be difficult to tell when they have converged.

These problems are difficult to avoid, however, when quantifying uncertainty. Most current methods of uncertainty analysis require the use of Bayesian methods of solution, primarily because the posterior distribution of an estimated parameter typically comes packaged with a measure of variance. For example, Cockayne, et al. (2019) lays out a set of reasonable conditions under which the solution to a Dynamical Systems attained using Bayesian methods can give the average-case error[3].

# 4  Current Project

Of course, solving all of the problems in the field of Numerical Solutions to Dynamical Systems, while something we would like to do, is a bit of a tall ask for a rather limited research period, and we include the above sections mostly to give a sense of the broad direction we would like to go in our research. For the time being, however, we will be content to reproduce some of the results found in the papers listed above, and to provide a limited proof of concept for some of our own methods.

Our work here will follow most closely from Levine and Stuart (2021) and will focus primarily on cases in which the main model–including parameters––is known, or at least hypothesized, but misses extra terms in the right-hand

side of the system. In particular, if we consider the system describing the evolution of the state $x \in \mathbb{R}^m$ according to the system

$$\dot{x} = f(x; \eta).$$

where we further divide the right-hand side into the known part and the unknown part, say $f^*(x; \eta) = f_0(x) + m^*(x; \eta^*)$, where $f_0$ is assumed known, while $m^*$ and $\eta^*$ are assumed fixed. For the present project, we assume that $m_j^*(x) = \sum_{i=1}^d \eta_{ij} m_i(x)$ is a superposition of known, elementary functions of $x$, scaled by the parameters in question, $\eta^*$, though we expect this to generalize to cases where the form of $m^*$ is more complicated and we have to search over a set of basis functions, such as the spline functions that approximate well any function over the domain in question. Our goal is to learn the values of all the $\eta_{ij}^*$ through observing data generated by the perturbed system. We will write our approximations to the added model as $m(x, \eta)$, where for the current project $m$ can be assumed to be identical to $m^*$, and each $\eta_{ij}$ approximates the true $\eta_{ij}^*$.

We should note here, in order to justify our referring to $m^*$ as a perturbation, that we will assume $||f_0|| \gg ||m^*||$.

# 5    Methodology

We have already laid out several approaches to solving this problem, most of which have been Bayesian. We will take an approach designed to leverage more recent developments in the fields of machine learning and numerical optimization, in particular the *autograd* functionality of *torch*, which allows us to backpropagate error through nearly any operation we can think of.

## 5.1    Backpropagation-Based Optimization

Our main approach to learning the small parameters $\eta_{ij}$ of our model involves fairly straightforward backpropagation. We choose a suitable norm $\| \cdot \|$ and a suitable way to make predictions, the details of both of which are discussed further in (5.2). Then we let *torch* handle the details of computing the gradients with respect to $\eta$ of each step in the computation of our predictions, which will produce an update for each value of $\eta$. We can also check, for processes that are reasonably approximated by linear functions, that the

updates to $\eta$ are reasonable by computing simple finite differences, along the lines of

$$\delta\eta_{ij} = \frac{\|x(\eta + \varepsilon p_{ij})\| - \|x(\eta - \varepsilon p_{ij})\|}{2\varepsilon},$$

where $p_i j$ is chosen as the such that of its $ij^{th}$ element is 1 and all others are 0.

## 5.2 Choice of Evaluation Criteria

For any such backpropagation method to work, we need both a way to evaluate how good our current value of $\eta$ is and, if we need to update it, which directions we should look in. This requires a method of generating predictions and a loss function to evaluate performance. We will look at two choices.

One way of determining predictions and loss is to numerically solve the ODE for the current value of $\eta$ over the entire time domain and take the 2-norm of the difference with the given data, which in our case will be the solution of the system with the known value of $\eta^*$. We will label this $L : \mathbb{R}^{N \times d} \times \mathbb{R}^n \to \mathbb{R}$, defined by

$$L(x^*, \eta) = \|x^* - x_\eta\|,$$

where $x^*$ solves $\dot{x} = f_0(x) + m^*(x, \eta^*)$ and $x$ solves $\dot{x} = f_0(x) + m(x, \eta)$.

Another choice of evaluation criterion we wish to look into, which has several notable advantages over this method is what we will refer as the Diff Loss. The main idea behind Diff Loss is that, when we solve an entire system, errors in the solution will propagate forward, so that, for $\ell \gg 1$, we expect $\|x *_\ell - x_\ell\|$ to be dominated by accumulated error from preveious time points. Thus, most of our error would be noise, either the result of discretization error or from earlier $x_i$s.

We propose to fix this by learning over "diffs" of $x^*$, which we will write as $y^*$, defined as $y_i^* = x_{i+1} - x_i$. We will compute our predicted values $y_i$ based on $x^*$ by simply computing the right-hand side of the ODEs for our current $\eta$. More precisely,

$$y_i = f_0(x_i^*) + m(x_i^*, \eta), \, i = 0, \ldots, N - 1.$$

Then we compute a loss by merely taking the 2-norm over the true diff and the predicted diff, so that we might write $L : \mathbb{R}^{N \times d} \times \mathbb{R}^n \to \mathbb{R}$ as

$$L(x^*, \eta) = \|y^* - y_\eta\|.$$

# 6 Sample Problems

We mostly concern ourselves with two dynamical systems, which we perturb in known ways. We choose these two primarily because Fitzhugh-Nagumo is very well-behaved and should be easy to learn $\eta$ on, whereas Lorenz-63 is likely to be more difficult, due to its high sensitivity to changes in the model.

## 6.1 Fitzhugh-Nagumo

The Fitzhugh-Nagumo equations are defined as follows:

$$\dot{V} = c\left(V - \frac{V^3}{3} + R\right)$$
$$\dot{R} = -\frac{1}{c}\left(V - a - bR\right)$$

We won't go too deep into the theory of this system. Its specific history and a more complete description of its behavior of it are given in [2]. For our purposes, we will take $a = 0.2$, $b = 0.2$, $c = 3.0$, which produces oscillations in the state vector, where each variable changes between times of steep increase and times of relatively little change, as shown in Figure 1.
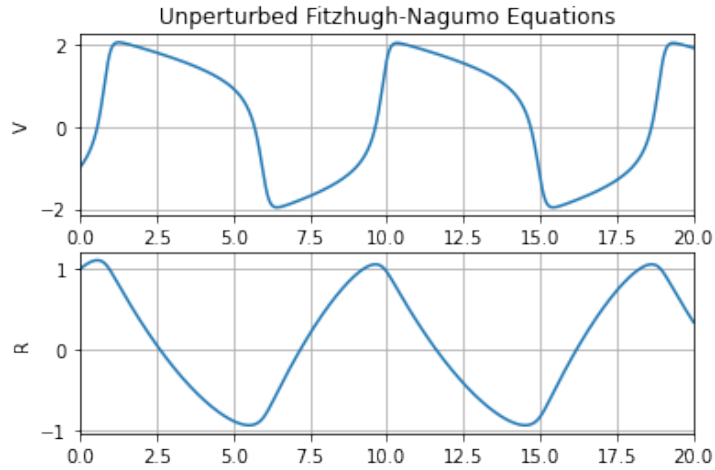


Figure 1: The Fitzhugh-Nagumo Equations with no perturbation.

## 6.2 Lorenz 63

The Lorenz-63 System is given as the following:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z$$

For choices of parameters $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, this produces a chaotic attractor which should look familiar to those inside and outside the mathematical community. The Lorenz attractor is particularly noted because of its sensitivity to small changes in initial conditions and scalar parameters.

# 7 Results

# 8 Future Directions

# References

[1] Andrew Gelman, Frederic Bois, and Jiming Jiang. Physiological pharmacokinetic analysis using population modeling and informative prior distributions. *Journal of the American Statistical Association*, 91(436):1400–1412, 1996.

[2] J. O. Ramsay, G. Hooker, D. Campbell, and J. Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[3] Jon Cockayne, Chris J. Oates, T. J. Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM Review*, 61(4):756–789, 2019.

[4] Oksana A. Chkrebtii, David A. Campbell, Ben Calderhead, and Mark A. Girolami. Bayesian Solution Uncertainty Quantification for Differential Equations. *Bayesian Analysis*, 11(4):1239 – 1267, 2016.
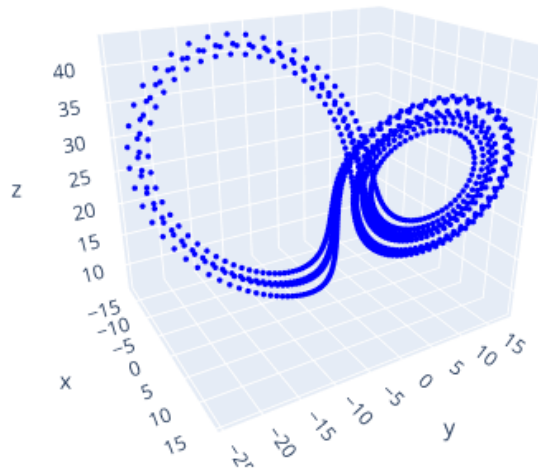
Unperturbed Lorenz-63



Figure 2: The classical Lorenz-63 System

[5] Matthew E. Levine and Andrew M. Stuart. A framework for machine learning of model error in dynamical systems, 2021.