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

- $f(x, \eta) = f_0(x) + m(x, \eta)$: Predicted model, including known terms and estimated supplemental
- $\eta \in \mathbb{R}^{p \times d}$: 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 parameters of a differential equation, the problem of minimizing 2-norm between a model's predictions and a given dataset is, in principle, ill-posed[1]. Thus, the problem of finding such a model still has challenges, even moreso for the problem of finding a model that is interpretable.

We look for an approximation $f$ that approximates $f^*$ optimally under some loss $L(f^*, f)$ (further details on the loss later), there are a few main sources of loss. The sources we want to focus on going forward are discretization error, model error, and "true" stochastic noise. In this paper, we will address the issue of model error, which is defined broadly to be the error in the predicted outcomes of a dynamical system due to misspecifying or underspecifying the dynamics governing the change in the state vector. For example, if one sets $\dot{x} = kx$, but in reality $\dot{x} = kx^2$, there is little hope to recover the "real" model through simply estimating the parameters of the chosen model.

That is, consider a state vector $x$, which evolves according to

$$\dot{x}(t, \eta^*) = f(x, \eta^*) \equiv f_0(x) + m^*(x, \eta^*),$$

and a given dynamical systems model

$$\dot{x}(t) = f_0(x),$$

for which there is some *a priori* reason to believe is a good model for the time-evolution of $x$. Then we wish to find

$$f(x, \eta) = f_0(x) + m(x, \eta),$$

such that $L(f^*(x, \eta^*), f(x, \eta))$ is minimized.

One of the advantages of this approach in the context of a general problem is the ability to make use of both domain knowledge–through incorporation of the initial model $f_0$–and data by way of learning $m(x)$[2].

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

# 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 data $x_1^*, \ldots, x_N^*$, can an estimate of $\eta^*$ be obtained?

One such approach to this problem involves Bayesian statistical models, as in Gelman, et al. (1996). This approach begins with a prior distribution describing model parameters. The differential equations are then solved numerically under the given parameter values, and the prior distributions are updated according to a Bayesian update procedure. This approach has several advantages, in particular that it is able to account for hierarchical effects. In the particular problem considered, there are both population effects and individual effects, which can be considered at the same time with their hierarchical structure intact.[3] However, this method also requires performing MCMC, which often fails to converge, and its convergence can almost never be verified.[4] Primarily for this reason, we will favor methods that do not involve performing MCMC.

More recently, the Bayesian approach for parameter estimation has fallen out of favor and been largely replaced by methods which rely on basis function expansions. Ramsay, et al. (2007), use one such method which treats the problem as a nested optimization. In an outer problem, they update the parameters, optimizing the Euclidean distance between their model predictions and observed data. The inner problem finds the coefficients for the basis functions which solve the differential equations for the current parameters.[5]

Even more recently, Levine and Stuart (2021) use techniques from machine learning to predict unobserved states that may affect the observed state vector. Their methods are remarkably powerful and show additionally that starting with an educated guess of the true model can cause RNN predic-

tions to reach optimality much faster than when they are given only data.[2] However, there are several questions that the authors do not answer. For example, it is not clear that these parameters are uniquely interpretable. The method might have terms that cancel each other, which could lead to very high parameter values, and the given model may be absorbed.

### 3.2 Uncertainty Quantification in Numerical Solutions of ODEs

While estimating the parameters of the ODEs themselves is a large part of this problem, it may also be useful to know how precise those estimates are. For example, Chkrebtii, et al. (2016) use a Bayesian method that conditions on model uncertainty in order to solve the system itself. They also use this model information to prove convergence properties of their algorithm, which could be challenging to show for other methods.[6] However, the use of Bayesian methods, especially ones which rely on MCMC, can lead to a variety of practical problems, as noted above.

These problems are difficult to avoid, however, when quantifying uncertainty. Most current methods of uncertainty analysis require the use of Bayesian methods, largely 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 of the parameter estimates.[7]

## 4 Current Project

While the above is meant to give context for this project, the scope of this work is limited to reproducing some of the literature results and providing a proof of concept for future work.

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 fixed with $m^*$ small relative to $f_0$. Further 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, $\eta^*$ (future work will consider cases when $m^*$ is more complicated). Our goal is to learn the values of all the $\eta_{ij}^*$ through observing data generated by the perturbed system. 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 $\eta_{ij}^*$.

# 5 Methodology

Our approach to the problem of estimating $\eta$ will leverage recent developments in the fields of machine learning and numerical optimization, in particular the `autograd` functionality of `torch`, which allows automatic computation of derivatives with respect to the parameters.[8] Those derivatives can then be used to perform backpropagation in order to learn $\eta^*$ through iterated approximations $\eta$.

## 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

Any such backpropagation method relies upon a loss, which measures the fidelity of the estimated model to the true model. We will look at two

choices.

One such loss function considers the full numerical solution to the ODE with $\eta$ given and compares it to the full numerical solution of the true model. Define

$$L_1(f) = \|x(\eta^*) - x(\eta)\|_2^2,$$

where $x(\eta) \in \mathbb{R}^N$ is the numerical solution to the model given $\eta$. This loss is intuitive but might not provide useful information when the proposed model's solution differs greatly from that of the true model.

One loss which might provide better information compares the *diffs* $x^*$ to the per-time-step changes in the proposed model. Define $y^* \in \mathbb{R}^{N-1}$ such that

$$y_i^* = x_{i+1}^* - x_i^*,\ i = 1, \ldots, N-1.$$

These are meant to capture the step-by-step change in the true solution. Then let the new loss

$$L_2(f) = \|y^* - m(x_i^*, \eta)\|_2^2.$$

This has a few advantages over $L_1$, including that it remains useful even if the predicted solution and the true solution behave very differently, as it only considers the change in state over short periods of time. It also does not require ever solving the proposed model, which makes the algorithm converge much faster.

A third loss which we will consider resembles $L_2$ in that it depends upon diffs but considers the full solution of the problem for $\eta$ given. Define $y(\eta) \in \mathbb{R}^{N-1}$ according to $y(\eta)_i = f_0(x_i^*) + m(x_i^*, \eta)$. Then denote

$$L_3(f) = \|y^* - y(\eta)\|_2^2$$

# 6 Sample Problems

We will focus on two systems primarily: the Fitzhugh-Nagumo system, which generally has stable behavior with respect to its parameters, and Lorenz-63 which exhibits chaotic behavior, which may make the task of learning $\eta$ much harder.

## 6.1 Fitzhugh-Nagumo

The Fitzhugh-Nagumo equations are defined as follows:

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

The specific history of this system, as well as a description of its behavior of it are given in [5]. Take $a = 0.2$, $b = 0.2$, $c = 3.0$, which produces oscillations in the state vector.

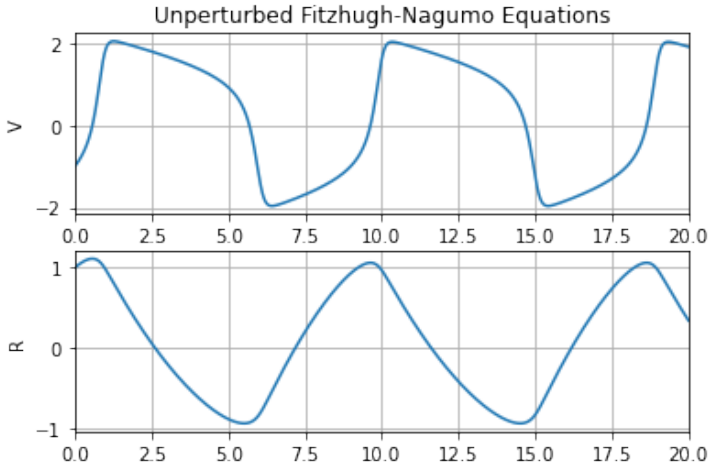


Figure 1: The Fitzhugh-Nagumo Equations with no perturbation.

## 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. Due to the chaotic behavior of the Lorenz attractor, even small changes in the right-hand side of its equations can cause significant change in the trajectory of the solution.

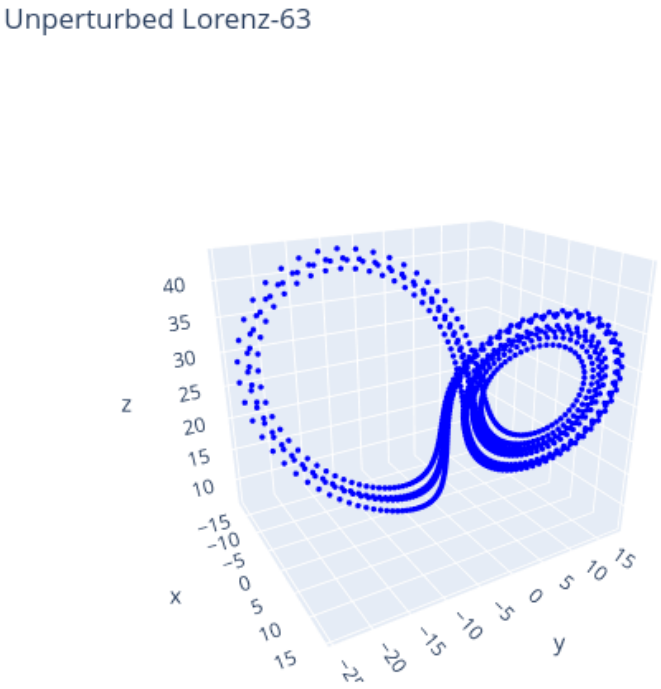


Figure 2: The Lorenz-63 System

# 7 Results

# 8 Additional Information

The code used to obtain the results above can be found at `https://github.com/carterkoehler/dynamical-systems-learning`

# References


[1] Matthias Chung, Mickaël Binois, Robert B. Gramacy, Johnathan M. Bardsley, David J. Moquin, Amanda P. Smith, and Amber M. Smith. Parameter and uncertainty esitmation for dynamical systems using surrogate staochastic processes. *SIAM Journal of Scientific Computing*, 41(4):A2212 – A2238, 2019.

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

[3] 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.

[4] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Brkner. Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667 – 718, 2021.

[5] 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.

[6] 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.

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

[8] Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson, and Laurence Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124(1):171–190, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.