frontiers | Frontiers in Ecology and Evolution

TYPE Methods
PUBLISHED 04 November 2022
DOI 10.3389/fevo.2022.1010278

OPEN ACCESS

EDITED BY
György Barabás,
Linköping University, Sweden

REVIEWED BY
Jurg Spaak,
University of Namur, Belgium
Jeremy Van Cleve,
University of Kentucky, United States

*CORRESPONDENCE
Steven A. Frank
saf@stevefrank.org

SPECIALTY SECTION
This article was submitted to Models in Ecology and Evolution, a section of the journal Frontiers in Ecology and Evolution

RECEIVED 02 August 2022
ACCEPTED 04 November 2022
PUBLISHED 04 November 2022

CITATION
Frank SA (2022) Automatic differentiation and the optimization of differential equation models in biology. *Front. Ecol. Evol.* 10:1010278. doi: 10.3389/fevo.2022.1010278

Automatic differentiation and the optimization of differential equation models in biology

Steven A. Frank*

Department of Ecology and Evolutionary Biology, University of California, Irvine, Irvine, CA, United States

A computational revolution unleashed the power of artificial neural networks. At the heart of that revolution is automatic differentiation, which calculates the derivative of a performance measure relative to a large number of parameters. Differentiation enhances the discovery of improved performance in large models, an achievement that was previously difficult or impossible. Recently, a second computational advance optimizes the temporal trajectories traced by differential equations. Optimization requires differentiating a measure of performance over a trajectory, such as the closeness of tracking the environment, with respect to the parameters of the differential equations. Because model trajectories are usually calculated numerically by multistep algorithms, such as Runge-Kutta, the automatic differentiation must be passed through the numerical algorithm. This article explains how such automatic differentiation of trajectories is achieved. It also discusses why such computational breakthroughs are likely to advance theoretical and statistical studies of biological problems, in which one can consider variables as dynamic paths over time and space. Many common problems arise between improving success in computational learning models over performance landscapes, improving evolutionary fitness over adaptive landscapes, and improving statistical fits to data over information landscapes.

KEYWORDS

automatic differentiation, optimization, fitness landscape, statistical inference, differential equation modeling, ecological dynamics, evolutionary dynamics

1. Introduction

Theoretical studies often analyze improvement on a performance surface. What phenotypes enhance fitness? What differential equation model best describes ecological or biochemical dynamics? What interventions improve the health of organisms or ecosystems?

Similarly, inductive methods of statistics and machine learning optimize prediction, minimize error, or maximize the use of information (McElreath, 2015; Goodfellow et al., 2016). Natural selection can itself be thought of as nature's inductive process of improving the fit of organisms to their environment (Frank, 2009; Frank and Fox, 2020).

Most of these problems reduce to finding a path that enhances performance (Floudas and Pardalos, 2008). In practice, that often means changing model parameter values in the direction along which the slope of performance tilts most strongly toward

improvement. Ideally, that best slope is found by differentiating the performance surface with respect to the parameters of the model or the traits of the organism.

Taking the derivative of a performance function with respect to its parameters is, in principle, a simple process. Performance functions typically arise by composition of basic mathematical operations such as addition, multiplication, and exponentiation. Standard procedures yield the derivative for each basic operation. The overall derivative follows by the chain rule for differentiation.

Two aspects hinder the calculation of derivatives. First, performance measures often follow from long chains of operations embedded in complex algorithms. For example, performance may depend on a measure of the distance between some optimal trajectory and the trajectory of a system of differential equations specified by a model with many parameters.

In practice, one often calculates the model trajectory for a given parameter set by a numerical algorithm. The derivative of the distance measure between the optimal and realized trajectories must be calculated through the operations of the numerical algorithm, which is typically embedded in the code of a computer program. Efficiently updating the parameters to reduce the distance of the model's trajectory from the optimal trajectory benefits from an automatic method to calculate the derivative of that distance with respect to the parameters (Griewank and Walther, 2008; Baydin et al., 2018; Margossian, 2019).

The second difficulty for calculation arises because performance functions often depend on large numbers of parameters. For example, the modern artificial intelligence revolution arose in part from expanding neural networks to multiple deeply connected layers with millions of parameters. It only became possible to calculate an improving pathway of parameter values with sufficient speed after the development of efficient methods for automatic differentiation of performance functions embedded within the calculations of complex computer code (Goodfellow et al., 2016).

Conceptually, automatic differentiation provides insight into the topography of performance surfaces. In biology, many problems of evolutionary dynamics turn on the adaptive topography that maps phenotypes to fitness (Stadler, 2002; Malan, 2021). The close similarity of evolutionary dynamics and the many other problems that depend on the topography of performance surfaces provides insight into how natural selection designs organisms and into the prospects and limits of optimization models in biology. In other words, automatic differentiation will help us to learn more about the shape of optimization problems in both natural history and in models of evolutionary biology.

This article focuses on the benefits of automatic differentiation for differential equation modeling and for broader problems in theoretical biology. Highlights include the opportunities for new kinds of theoretical approaches, a better understanding of current developments in statistics and machine learning, and some possible directions for future research motivated by improved understanding of multidimensional performance surfaces.

I start with an overview of differentiation in optimization. I next turn to the techniques and advantages of automatic differentiation for optimizing differential equation models (Chen et al., 2018; Rackauckas et al., 2020). I then present two examples to illustrate how the optimization of differential equations by automatic differentiation may enhance the future study of various topics in biology.

The first example fits various differential equation models to the classic data on predator-prey population cycles of lynx and hare. The second example searches for transcription factor network designs that solve the challenge of maintaining an internal circadian rhythm. The internal rhythm must buffer against the intrinsic stochasticity of biochemical dynamics, entrain to an erratic external circadian signal when available, and maintain the internal rhythm when the external signal is absent.

Slopes and curvatures calculated by differentiation also provide the basis for enhanced sensitivity analysis (Mester et al., 2022). In biology, sensitivity plays a key role in understanding robustness, variability, and evolutionary dynamics. In inference, sensitivity influences the degree of belief in conclusions drawn from models and data, often analyzed by Bayesian methods.

2. Differentiation in optimization

Optimizing a function L with respect to a parameter vector $\mathbf{p}$ sets a classic optimization problem. For example, what parameters for a system of differential equations minimize the distance between the system's trajectory and the observed trajectory of fluctuating lynx and hare populations? In this case, the total distance, L, might be the sum of the squared deviations between system's trajectory and the observed trajectory when measured at series of temporal intervals.

In general, it is relatively easy to find a local optimum near some initial point in parameter space and relatively difficult to find a global optimum over the full space of potential parameter values. Many optimization techniques exist. The best methods for a particular problem depend on the shape of the performance surface (Floudas and Pardalos, 2008; Ruder, 2017; Reddi et al., 2019).

For problems with continuous performance surfaces, using the derivative of L with respect to the parameters, $\mathbf{p}$, often greatly enhances the search for better parameters. The vector of partial derivatives of L with respect to the parameters is the gradient or slope of the performance surface, ∇L.

The gradient describes the change in performance with respect to the change in parameters. Thus, the gradient provides much information about how to update parameter values in

an iterative search for improving performance. Knowing the gradient does not by itself solve the problem of getting stuck in a local optimum. However, fast calculation of the gradient often transforms problems from hard and beyond reasonable study into ones that can be analyzed relatively easily (Goodfellow et al., 2016; Ruder, 2017; Reddi et al., 2019).

Fast calculation of derivatives by automatic methods may also allow calculation of the second derivatives, which describe the curvature of the performance surface. For a problem with n parameters and a single-valued performance measure, L, we write $\nabla^2 L$ for the $n \times n$ Hessian matrix of partial second derivatives. The Hessian matrix provides insight into many aspects of optimization.

3. Optimizing differential equations

Differential equations provide the primary modeling approach for many problems (Edelstein-Keshet, 1988; Ellner and Guckenheimer, 2006). Organismal traits often follow trajectories over time as individuals develop and respond to the environment. Systems of gene regulation, biochemistry, and physiology describe temporal changes in molecular concentrations and biological outputs. Continuous models of population genetics describe temporal changes in allelic and genotypic frequencies. Ecological dynamics describe temporal changes in populations and interacting species.

The differential equations describe a vector of values, $\mathbf{x}$, that depend on time, t, on a parameter vector, $\mathbf{p}$, on initial conditions, and on various other inputs. If we write the system values at time t as $\mathbf{x}_t$, then we can abbreviate the differential equations as

$$\mathbf{x}'_t = f(\mathbf{x}_t, \mathbf{p}),$$

in which the prime denotes differentiation with respect to t.

Many studies seek a parameter vector, $\mathbf{p}$, that optimizes a temporal trajectory of values, $\mathbf{x}_t$, over some time span. Analysis optimizes a performance function, $L(\mathbf{X})$ with respect to $\mathbf{p}$ for the system values at various times, $\mathbf{X} = \{\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots\}$.

This section illustrates automatic differentiation for the simple differential equation

$$x'_t = f(x_t, \mathbf{p}) = r\left(k - x_t\right) \tag{1}$$

with parameter vector $\mathbf{p} = \{r, k\}$. We use the performance function $L(x_T) = (x_T - c)^2$, the squared Euclidean distance between the system's value at particular time, T, and the target value, c, with the goal of minimizing the performance function.

Equation (1) has a simple explicit solution, which we could use to find optimum values of $\mathbf{p}$. However, in most applications, the system of differential equations must be evaluated numerically. Optimization must be done by choosing some parameters, numerically calculating the solution, x_t, and the loss function, L, and then updating the parameters to improve performance (Chen et al., 2018; Rackauckas et al., 2020). The optimization procedure consists of repeated rounds of calculation and parameter updating. The goal is to improve performance and, ideally, to find an optimal parameter vector that minimizes the loss function.

To illustrate the typical optimization cycle, we evaluate Equation (1) numerically by Euler's method

$$x_{t+h} \approx x_t + hf(x_t, \mathbf{p}) = x_t + hr\left(k - x_t\right), \tag{2}$$

in which h is the stepsize over which we update values. In practice, one uses better numerical methods to approximate the trajectory that x_t follows over time (LeVeque, 2007). However, the concepts by which one applies automatic differentiation do not depend on the numerical technique, so Euler's method is sufficient to illustrate how one uses automatic differentiation to aid in optimizing the parameters of differential equations.

4. Automatic differentiation

Automatic differentiation uses the chain rule to break long calculations into small pieces, each of which can be easily differentiated (Griewank and Walther, 2008; Baydin et al., 2018; Margossian, 2019). The chain rule tells us how to combine the differentiated pieces into the overall derivative. For example, given the function

$$f(p_1, p_2) = 2p_1p_2,$$

we can set $v = p_1p_2$ and obtain the partial derivative of f with respect to p_1 by the chain rule

$$\frac{\partial f}{\partial p_1} = \frac{\partial f}{\partial v}\frac{\partial v}{\partial p_1} = 2p_2.$$

The final value multiplies the two component derivatives, $\partial f/\partial v = 2$ and $\partial v/\partial p_1 = p_2$. To use this piece as part of a longer calculation, we substitute the numerical value of p_2 and store only that numerical value. For example, if $p_2 = 0.1$, then $\partial f/\partial p_1 = 0.2$.

In general, long calculations can be broken into a sequence of simple steps. At each step, we calculate the numerical value of the component derivative, combine that component's value with prior component values, and then store the resulting numerical value to use in subsequent steps. Thus, we need to store only a small amount of information as we sequentially combine the component values to obtain the final overall value. The calculation has the same theoretical exactness as a complete symbolic derivative but, by substituting numerical values as we go along, we can proceed faster and with much less computational storage space. Alternatively, many optimization methods estimate the derivatives numerically, which requires relatively little storage space but has much lower accuracy. The

TABLE 1 Calculations through the computational graph in Figure 1.

Forward value trace		Forward derivative trace ($\dot{z} = \partial z/\partial r$)		Reverse derivative trace ($\bar{z} = \partial L/\partial z$)	
$v_0 = x_0$	$= 0$	$\dot{v}_0 = \dot{x}_0$	$= 0$	$\bar{x}_0 = \bar{v}_0 = \bar{v}_5\,\partial v_5/\partial v_0$	$= -1.998$
$v_1 = r$	$= 0.1$	$\dot{v}_1 = \dot{r}$	$= 1$	$\bar{r} = \bar{v}_1 = \bar{v}_3\frac{\partial v_3}{\partial v_1} + \bar{v}_4\frac{\partial v_4}{\partial v_1}$	$= -0.01998$
$v_2 = k$	$= 1$	$\dot{v}_2 = \dot{k}$	$= 0$	$\bar{k} = \bar{v}_2 = \bar{v}_4\,\partial v_4/\partial v_2$	$= -0.001998$
$v_3 = hv_0v_1$	$= 0$	$\dot{v}_3 = h\,(v_0\dot{v}_1 + \dot{v}_0v_1)$	$= 0$	$\bar{v}_3 = \bar{v}_5\,\partial v_5/\partial v_3$	$= 1.998$
$v_4 = hv_1v_2$	$= 0.001$	$\dot{v}_4 = h\,(v_1\dot{v}_2 + \dot{v}_1v_2)$	$= 0.01$	$\bar{v}_4 = \bar{v}_6\,\partial v_6/\partial v_4$	$= -1.998$
$v_5 = v_0 - v_3$	$= 0$	$\dot{v}_5 = \dot{v}_0 - \dot{v}_3$	$= 0$	$\bar{v}_5 = \bar{v}_6\,\partial v_6/\partial v_5$	$= -1.998$
$v_6 = v_4 + v_5$	$= 0.001$	$\dot{v}_6 = \dot{v}_4 + \dot{v}_5$	$= 0.01$	$\bar{v}_6 = \partial L/\partial v_6 = -2(1 - v_6)$	$= -1.998$
$L = (1 - v_6)^2$	$= 0.998001$	$\dot{L} = -2\,(1 - v_6)\,\dot{v}_6$	$= -0.01998$	$\bar{L} = \partial L/\partial L$	$= 1$

The left box traces the calculation of values forward from the inputs to the output, yielding the final value for L. The center box traces the forward chain rule calculation of the derivatives with respect to the input parameter, r, yielding $\dot{L} = \partial L/\partial r$ at the bottom. The variable z represents the left side of each equation. A separate derivative trace is required for each input parameter. The right box traces the reverse chain rule calculation of the derivatives, starting from the bottom and proceeding to the top. Reverse mode has the advantage that a single trace calculates the partial derivatives of L with respect to all of the input parameters. A constant value $h = 0.01$ is used in calculations. Structure of table based on Table 2 of Baydin et al. (2018).

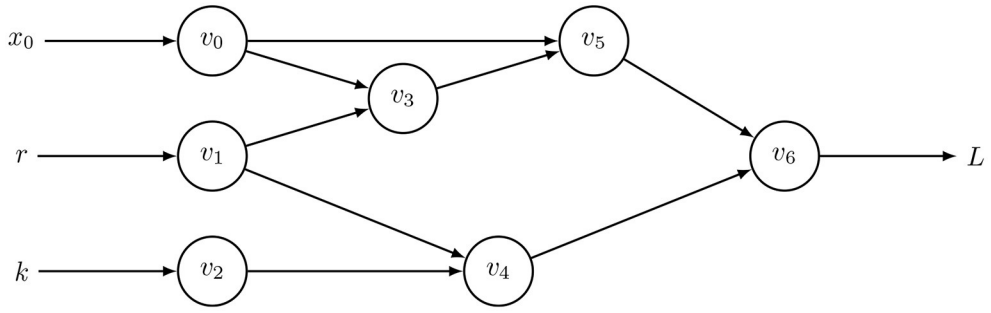

FIGURE 1
Computational graph to calculate $L(x_h) = (1 - x_h)^2$ for Equation (2) with $t = 0$ and $h = 0.01$, in which optimization seeks to minimize the loss function, L. Table 1 shows the intermediate results, v_i, with $v_6 = x_h$. This computational graph can be recursively expanded n times by using $v_6 = x_t$ as the starting point for additional applications of Equation (2) to calculate x_{nh} for any integer, n. The resulting graph quickly becomes too large to visualize or to trace the pathways of calculation without some automatic method. However, computer code readily expands such graphs and calculations.

errors in estimation impede optimization or cause optimization to fail.

The following subsections introduce two general methods of automatic differentiation and one special method for differentiating trajectories of differential equations. Those methods differ in how they apply the chain rule.

4.1. Forward method

The forward method begins with input parameters and then traces the chain of derivatives to the output. The center box of Table 1 illustrates the calculation for the computational graph of Equation (2), shown in Figure 1. That calculation yields the derivative of the output, L, with respect to the input parameter, r, written as $\dot{L} = \partial L/\partial r$.

Two early steps in the forward derivative trace show application of the chain rule. First, for $v_1 = r$, we obtain $\dot{v}_1 = 1$, in which the overdot denotes the partial derivative with respect to r. Second, for v_4, the chain rule expands to

$$\frac{\partial v_4}{\partial r} = \frac{\partial v_4}{\partial v_2}\frac{\partial v_2}{\partial r} + \frac{\partial v_4}{\partial v_1}\frac{\partial v_1}{\partial r} = h\,(v_1\dot{v}_2 + \dot{v}_1v_2) = 0.01,$$

using the constant $h = 0.01$ and substituting the other values given in the table. Continuing forward, we eventually arrive at the final value, $\dot{L} = -0.01998$.

The forward mode is simple and relatively easy to implement in computer code. A forward derivative trace can be done in parallel with the computations to calculate the performance value, L, for given inputs, as shown in the left box of Table 1.

Forward mode's primary disadvantage arises from the need to do a separate derivative trace for each input parameter. For large models with many parameters, full calculation of the gradient by forward automatic differentiation can be slow and limit application. Large optimization analyses often gain by using reverse mode automatic differentiation, which can calculate the gradient over all parameters in a single derivative trace.

4.2. Reverse method

The right box of Table 1 illustrates the reverse derivative trace for the same problem. Starting at the end, with L, and going backwards, the first step calculates $\bar{v}_6 = \partial L/\partial v_6$. Then, by the chain rule, $\bar{v}_5 = \partial L/\partial v_5 = \bar{v}_6\,\partial v_6/\partial v_5$. By this method, the calculation continues in the reverse direction. In the final steps, at the top of that box, one can calculate ∂L with respect to each input parameter, yielding the full gradient ∇L in one backward trace.

Modern neural network models often have millions of parameters. A single backward derivative trace calculates the full gradient of the performance function with respect to all parameters. That technique, often called back propagation in the neural network literature, provided a necessary advance in computation for the great recent breakthroughs in modern neural network modeling (Goodfellow et al., 2016).

The same method also enhances the potential to optimize large differential equation models. However, for differential equations, one typically needs to optimize through the numerical methods used to approximate the temporal trajectories of systems. To accomplish a practical method for reverse mode differentiation through the numerical approximation algorithm required another conceptual advance (Chen et al., 2018; Rackauckas et al., 2020).

4.3. Differentiating trajectories of differential equations

In essence, we apply the reverse method to the computational graph for the trajectory of the differential equation. This approach leads to a new differential equation that propagates the derivative of the performance function L backward in time, similarly to how we traced the derivative of L back to the parameters in the reverse mode approach of Table 1. Here, I describe the concepts in an approximate and intuitive way. I follow Chen et al. (2018), who provide a full explanation and complete derivation.

The goal is to calculate the derivative of L as a function of the location of the trajectory at various times. For simplicity, we consider a univariate differential equation with location at a single time, x_t, and associated performance, $L(x_t)$. We seek the gradient of L with respect to the parameters, $\mathbf{p}$, and the initial condition, x_0.

Start with the definition $a(t) = \mathrm{d}L/\mathrm{d}x_t$. We then consider how this value changes with time, t, creating a new differential equation, $a'(t)$, with the prime denoting differentiation with respect to time

$$a'(t) = \frac{1}{h}\left(\frac{\mathrm{d}L}{\mathrm{d}x_t} - \frac{\mathrm{d}L}{\mathrm{d}x_{t-h}}\right) = \frac{\mathrm{d}L}{h\mathrm{d}x_t}\left(1 - \frac{\partial x_t}{\partial x_{t-h}}\right),$$

for small h. The term $\partial x_t/\partial x_{t-h}$ is a backward chain rule expansion, like the steps in the reverse mode trace of Table 1. Noting from Equation (2) that $\partial x_t/\partial x_{t-h} = 1 + h\partial f/\partial x_{t-h}$, and letting h go to zero, we obtain

$$a'(t) = -a(t)\frac{\partial f}{\partial x_t}. \tag{3}$$

As in the reverse trace of Table 1, we can trace derivatives backwards, in this case tracing back in time for $\partial x_t, \partial x_{t-h}, \ldots, \partial x_h$ to calculate the trajectory of $a(t)$. In the final backward step, we apply the derivative of x_h relative to the input parameters, $\mathbf{p}$, as $\partial x_h/\partial \mathbf{p}$, in essence, allowing us to transform the last term of Equation (3) at time $t = h$ as

$$\frac{\partial f}{\partial x_h}\frac{\partial x_h}{\partial \mathbf{p}} = \frac{\partial f}{\partial \mathbf{p}}. \tag{4}$$

Using that method to analyze changes in L and f with respect to $\mathbf{p}$ instead of with respect to x_t, we can write the solution for Equation (3) as

$$\frac{\partial L}{\partial \mathbf{p}} = -\int_{t_1}^{t_0} a(t)\frac{\partial f}{\partial \mathbf{p}}\mathrm{d}t. \tag{5}$$

Here, the backward integral trace in Equation (5) equals the solution backward in time of the differential equation in Equation (3), modified by a final transformation that yields the derivative of the performance function, L, with respect to the parameters. Standard numerical methods to analyze differential equations can be used to find the solution for Equation (5).

The formal derivation of Equation (5) uses a more rigorous analysis to change $\partial f/\partial x_t$ in Equation (3) into $\partial f/\partial \mathbf{p}$ in Equation (5), arriving at the same conclusion (Chen et al., 2018).

5. Examples

5.1. Fitting data for temporal trajectories

One can fit differential equation models to time series data. For example, Figure 2 shows the dynamics of hare and lynx populations. The blue curve shows the fluctuation of the hare population, and the green curve shows the fluctuation of the lynx population. The gold curves show a Bayesian fit of a differential equation model to the initial 2/3 of the data and the model's prediction compared to the data for the final 1/3 of the time series.

Each model has n variables, two for the log transformation of hare and lynx population sizes and $n-2$ for dummy variables tracking unobserved factors. The models seek to match the data of hare and lynx population sizes observed over 91 years in a classic ecological study summarized by Odum and Barrett (1971)

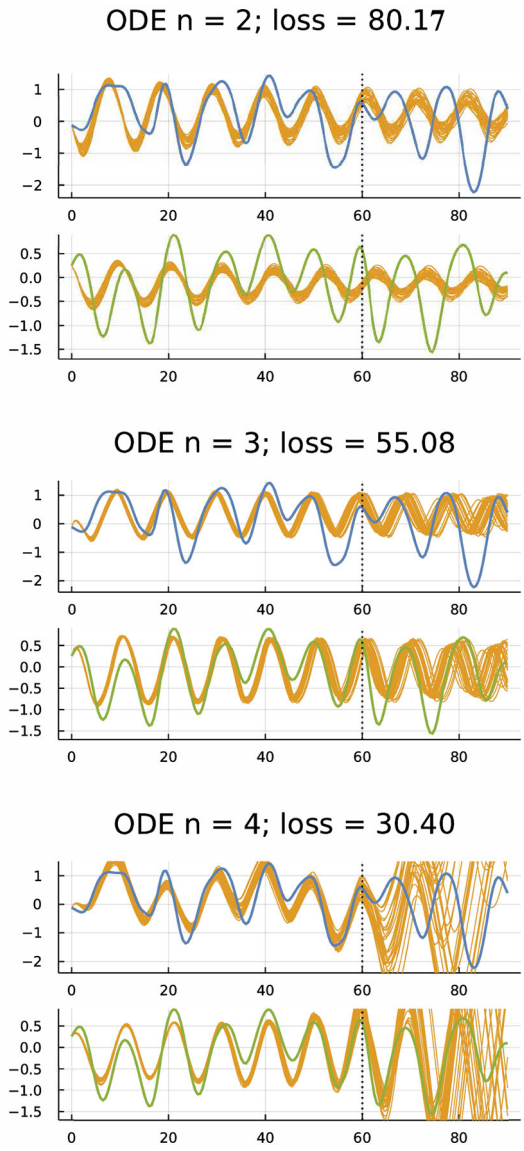

FIGURE 2
Dynamics of hare (blue) and lynx (green) populations (Odum and Barrett, 1971; Bonnaffé et al., 2021) and fitted models (gold) (Frank, 2022b). The ordinary differential equation (ODE) models were fit to the first 61 yearly observations. The gold curves past year 60 show predictions for subsequent dynamics. For $n = 2$, the model had one variable for hare and one for lynx. For $n = 3, 4$, the model had one or two extra variables to account for possible unobserved factors, providing more parameters to fit the data. Smaller loss means a better fit to the first 60 years. The model fitting was done by an approximate Bayesian method based on the gradient of the loss calculated by automatic differentiation (Li et al., 2015). For each model, gold trajectories arise from 30 randomly chosen parameter combinations of the Bayesian posterior distribution. From Figure 4 of Frank (2022b), which provides methods, a broader analysis of these data, and other fitting approaches such as neural ODEs that use artificial neural networks to fit differential equations to target trajectories.

and Bonnaffé et al. (2021). The differential equation for the vector of variables $\mathbf{u}$ in the ODE models has the form

$$\frac{d\mathbf{u}}{dt} = f(\mathbf{S}\mathbf{u} - \mathbf{b}), \tag{6}$$

in which the first two components of $\mathbf{u}$ are the log-transformed populations sizes, the n^2+n parameters are in the $n \times n$ matrix, $\mathbf{S}$, and the n vector, $\mathbf{b}$. The function f maps the n dimensional input to an n dimensional output, potentially inducing nonlinearity in the model. One typically selects f from the set of common activation functions used in neural network models. For all runs in Frank (2022b), I used $f = \tanh$ applied independently to each dimension. It would be easy to study alternative ODE forms. However, the analyses in Frank (2022b) focused on Equation (6).

The three sets of plots in Figure 2 correspond to models with $n = 2, 3, 4$ variables. For $n = 2$, the model has one variable to track the hare population and one to track the lynx population. That model roughly matches the frequency but not the amplitude of population fluctuations during the fitting period for the initial 2/3 of the data, corresponding to the first 60 years. The model fails to predict the frequency or amplitude accurately during the prediction period years 60–90.

For $n = 3$, the model adds another variable to account for unobserved factors. That model matched the fitting period better because it has an extra dummy variable and more parameters. The extra parameters determine the dynamics of the dummy variable and its interactions with the hare and lynx populations. The model's dynamics during the prediction phase roughly matches the frequency of population fluctuations. Although a bit off on amplitude, the fit is reasonably good given the very limited data and complex dynamics.

For $n = 4$, the model has two extra dummy variables and more parameters. The match to the fitting period is very good, with little variation among the sampled trajectories from the Bayesian posterior distribution of parameters. However, the match during the prediction phase is highly erratic.

Overall, the model with $n = 2$ appears underfit, with insufficient parameters and flexibility to match the observed pattern or predict future observations. The model with $n = 4$ appears overfit, with a close match to the fitted period but poor match for the predicted period. The model with $n = 3$ appears to be a good compromise, a conclusion supported by further analyses in Frank (2022b).

The Bayesian fitting procedure used automatic differentiation through a numerical differential equation solver. That approach provided a fast computational method to find a good fit to the data. The computational benefit does not alter the challenge of deciding what model to use or the criteria for deciding the relative success of different models. Instead, the method provides a simple and practical way

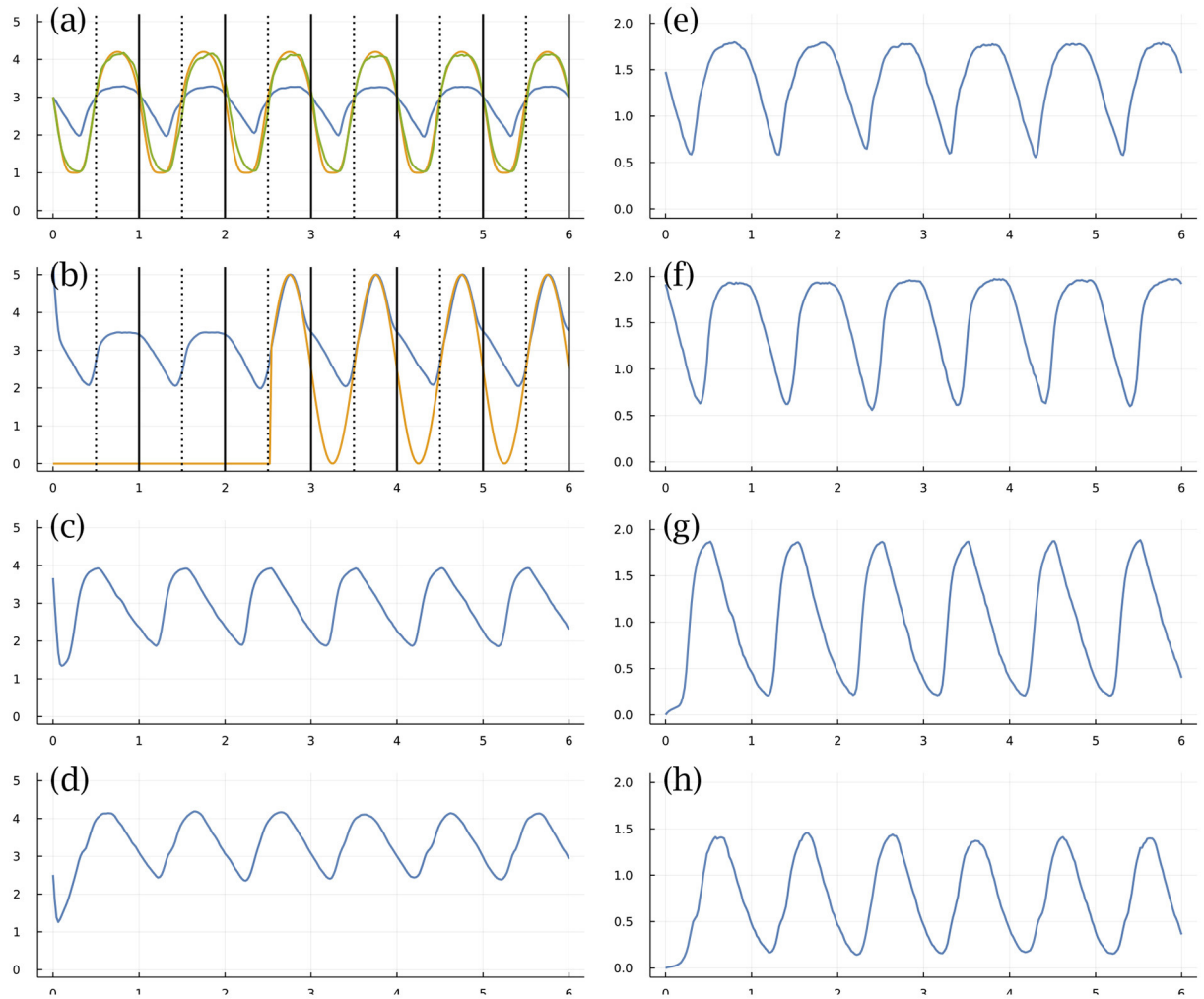

FIGURE 3
Circadian dynamics with stochastic fluctuations and random daylight signal. **(a–d)** Abundance of TFs on log10(1 + *y*) scale for *y* molecules per cell. **(e–h)** Abundance of mRNAs for associated TFs. See text for explanation. From Figure 1 of Frank (2022a).

of doing the computations that may be difficult to achieve with other methods. Technical innovation often leads to conceptual advance.

5.2. Analyzing adaptive traits

One can also fit a model to a theoretical challenge rather than to data. In particular, how can one build systems that produce good trajectories relative to some goal? How does natural selection design biological systems to solve environmental challenges and increase fitness?

Consider how cells control gene expression. Much of that control arises from transcription factors (TFs), which are proteins that bind to DNA and alter the rate at which nearby genes transcribe mRNA. Following the details in Frank (2022a), we can study a simple model.

Suppose we have n genes. Each gene expresses an mRNA, which makes a TF protein. Thus, our differential equation model has to track the temporal trajectory of the numbers n mRNAs within the cell and the associated n TF proteins. The expression level of each gene is influenced by the abundances of the n TFs. We can think of the n TFs as inputs to the TF computational network, which produces n outputs that influence the expression level of the n genes. That input-output function depends on parameters that describe the binding of the TFs to DNA and the transformation of binding events into expression levels (Bintu et al., 2005a,b; Marbach et al., 2010).

We seek parameters to match a target temporal trajectory of TF abundances. Suppose we measure success by how well the first TF, labeled TF 1, follows a circadian pattern (Frank, 2022a).

Figure 3 shows an example, in which there are $n = 4$ TFs (left) and matching mRNA (right) levels. Figure 3a shows the

abundance of TF 1 (blue curve) on a log10(1 + y) scale for molecules per cell, y, over a time period of 6 days. Each day divides by the dotted vertical line, which denotes entry into daytime. The solid vertical line denotes entry into nighttime. We transform the molecular number (blue curve) into a cellular state by a commonly used Hill function (Frank, 2013; Zhang et al., 2013), yielding the green curve that describes the cellular circadian rhythm.

We measure success by the total Euclidean distance between the gold curve that describes the environmental circadian pattern and the green curve that describes the internal cellular circadian rhythm. We search for good parameters by optimization algorithms that use the gradient of the success measure relative to the parameters of the differential equation, calculating the gradient by autodifferentiation through the numerical algorithm that calculates the temporal trajectories.

In this example, the stochastic differential equation has random perturbations of the molecular numbers per cell. This example also imposes environmental randomness, shown in Figure 3b. The external daylight signal (gold curve) is initially off, then comes on in the middle of day 3 and stays on for the remainder of the period shown. In general, the external signal turns on and off randomly, such that cells may pass many days entirely in the dark. The blue curve of that panel tracks the abundance of TF 2. The external daylight signal strongly stimulates production of TF 2, providing an internal signal within the cell that could be used to entrain the circadian dynamics.

The computational challenge starts with random parameters. One then searches for those parameters that lead to an internal circadian rhythm that buffers the internal stochastic molecular perturbations, uses the external daylight signal for entrainment when it is present, and maintains a good internal rhythm when the external signal is absent. The particular example illustrated in Figure 3 handles all of those challenges (Frank, 2022a).

This model has 164 parameters. It would be difficult to find a good parameter combination without the remarkable computational advantages of automatic differentiation. In general, one can use this approach to generate and analyze hypotheses about how natural processes design biological systems to produce dynamic traits.

I finish my summary of examples with a few comments about using automatic differentiation in various applications. For the hare-lynx problem, fitting ODE models typically took several hours or less on a 2022 Apple M1 Ultra Mac Studio computer. In that study, I also fit neural ODE (NODE) models that use artificial neural networks (Frank, 2022b). Those NODE models have many more parameters and take longer to optimize but typically finish within many hours or $<$1 day.

For the TF problem, optimization time was typically several days, sometimes a week or more per run. These stochastic models are much more challenging problems for optimization through the differential equation solvers.

Although I optimized most individual algorithms for both problems, I made no attempt to minimize overall runtimes for any of these problems. So the times quoted here are only meant as very rough guidelines for those who might want to compare with their own approaches.

A question sometimes arises whether common derivative-free optimization methods could solve such problems as well or better than automatic differentiation. A full study of that question would require tuning the computer code to match various alternative methods. That has not been done for these problems. However, to make a quick test, I compared runs of the TF problem using automatic differentiation and the commonly used Nelder-Mead algorithm, which is a derivative-free optimization method. Using the same code with the alternative optimization methods, in three independent runs Nelder-Mead failed completely to track to circadian pattern. A single comparison run with automatic differentiation converged to a good solution.

6. Discussion

Many aspects of model interpretation depend on sensitivity (Mester et al., 2022). How much do model predictions change with a change in a parameter? For example, if one wishes to change dynamics, altering a parameter with a strong effect on outcome would be better than altering a parameter with little effect.

Confidence in parameter estimates also relates to sensitivity. If large changes in a parameter have little effect on outcome, then estimates for that parameter will vary widely and confidence in a particular estimate is low.

In evolutionary models of biological traits, sensitivity may relate to genetic variability. For a parameter with low sensitivity, changes in that parameter have relatively little effect on performance. Such parameters are likely to accumulate much associated genetic variability. By contrast, changes in sensitive parameters strongly affect performance, suggesting relatively little genetic variability associated with such parameters.

Because sensitivity often means the change in performance with respect to a change in parameters, one often evaluates sensitivity by the derivative of performance with respect to parameters. For models with many parameters, automatic differentiation provides computational benefits (Mester et al., 2022). In evolutionary analyses, the performance surface that defines sensitivity is the adaptive or fitness landscape (Stadler, 2002; Malan, 2021). Many conceptual aspects of evolutionary dynamics and of optimizing artificial neural networks come down to understanding how various factors alter the geometry of performance surfaces in models with large numbers of parameters (Yang, 2019).

For studies of sensitivity and performance surface geometry, Bayesian posterior distributions of parameters provide a complementary approach. A narrow distribution typically means that small changes in a parameter provide much information, which also typically means that small changes strongly alter model outcome. In biology, narrow Bayesian posteriors may associate with less genetic variability than broad posteriors, because narrowness associates with large fitness effects for small changes in trait values.

The approximate Bayesian method to generate the predicted trajectories in Figure 2 provides estimates for the posterior distribution of parameters. That method depends on automatic differentiation to calculate the gradient of performance with respect to the parameters. Roughly speaking, the gradient gives the directional change of the parameters favored to improve performance in the same way that biological fitness favors changes in traits to enhance fitness.

Balanced against that directional change in improved performance, the method introduces random fluctuations in parameters, similar to the way that mutation causes random changes in traits. The balance between directional selection and random mutation creates a distribution of parameter values that approximates the Bayesian posterior distribution. Those opposing forces also match the notion of genetic variability maintained by a balance between selection and mutation. The similarity between mutation-selection genetics and computational Bayesian analysis hints at the broad conceptual relations between evolutionary dynamics and the study of optimizing large systems in artificial neural networks and other domains.

As computational techniques for automatic differentiation improve, many new opportunities for theoretical advances will arise in domains for which optimization provides an important tool. Such opportunities for theoretical application will be matched by opportunities for greater conceptual understanding of processes that improve performance.

Data availability statement

Publicly available software and computer output were used in this study. The software code and output can be found at: https://doi.org/10.5281/zenodo.6463624; https://doi.org/10.5281/zenodo.6798421.

Author contributions

The author confirms being the sole contributor of this work and has approved it for publication.

Funding

The Donald Bren Foundation, National Science Foundation grant DEB-1939423 and DoD grant W911NF2010227 support my research.

Acknowledgments

A preprint of this manuscript is on arXiv (Frank, 2022b).

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.* 18, 1–43. Available online at: http://jmlr.org/papers/v18/17-468.html

Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J., et al. (2005a). Transcriptional regulation by the numbers: applications. *Curr. Opin. Genet. Dev.* 15, 125–135. doi: 10.1016/j.gde.2005.02.006

Bintu, L., Buchler, N. E., Garcia, H. G., Gerland, U., Hwa, T., Kondev, J., et al. (2005b). Transcriptional regulation by the numbers: models. *Curr. Opin. Genet. Dev.* 15, 116–124. doi: 10.1016/j.gde.2005.02.007

Bonnaffé, W., Sheldon, B. C., and Coulson, T. (2021). Neural ordinary differential equations for ecological and evolutionary time-series analysis. *Methods Ecol. Evol.* 12, 1301–1315. doi: 10.1111/2041-210X.13606

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv:1806.07366*. doi: 10.48550/arXiv.1806.07366

Edelstein-Keshet, L. (1988). *Mathematical Models in Biology*. New York, NY: Random House.

Ellner, S. P., and Guckenheimer, J. (2006). *Dynamic Models in Biology*. Princeton, NJ: Princeton University Press. doi: 10.1515/9781400840960

Floudas, C. A., and Pardalos, P. M., editors (2008). *Encyclopedia of Optimization*. New York, NY: Springer Science & Business Media.

Frank, S. A. (2009). Natural selection maximizes Fisher information. *J. Evol. Biol.* 22, 231–244. doi: 10.1111/j.1420-9101.2008.01647.x

Frank, S. A. (2013). Input-output relations in biological systems: measurement, information and the Hill equation. *Biol. Direct* 8:31. doi: 10.1186/1745-6150-8-31

Frank, S. A. (2022a). Optimization of transcription factor genetic circuits. *Biology* 11:1294. doi: 10.3390/biology11091294

Frank, S. A. (2022b). Optimizing differential equations to fit data and predict outcomes. *arXiv.2204.07833*. doi: 10.48550/arXiv.2204.07833

Frank, S. A., and Fox, G. A. (2020). "The inductive theory of natural selection," in *The Theory of Evolution*, eds S. M. Scheiner and D. P. Mindell (Chicago: University of Chicago Press), 171–193.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Griewank, A., and Walther, A. (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Philadelphia, PA: Society for Industrial and Applied Mathematics. doi: 10.1137/1.9780898717761

LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-dependent Problems*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). doi: 10.1137/1.9780898717839

Li, C., Chen, C., Carlson, D., and Carin, L. (2015). Preconditioned stochastic gradient Langevin dynamics for deep neural networks. *arXiv:1512.07666*. doi: 10.1609/aaai.v30i1.10200

Malan, K. M. (2021). A survey of advances in landscape analysis for optimisation. *Algorithms* 14:40. doi: 10.3390/a14020040

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. *Proc. Natl. Acad. Sci. U.S.A.* 107, 6286–6291. doi: 10.1073/pnas.0913357107

Margossian, C. C. (2019). A review of automatic differentiation and its efficient implementation. *WIREs Data Mining Knowl. Discov.* 9:e1305. doi: 10.1002/widm.1305

McElreath, R. (2015). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan, 2nd Edn*. Boca Raton, FL: Chapman & Hall/CRC.

Mester, R., Landeros, A., Rackauckas, C., and Lange, K. (2022). Differential methods for assessing sensitivity in biological models. *PLoS Comput. Biol.* 18:e1009598. doi: 10.1371/journal.pcbi.1009598

Odum, E. P., and Barrett, G. W. (1971). *Fundamentals of Ecology, 3rd Edn*. Philadelphia, PA: W. B. Saunders.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., et al. (2020). Universal differential equations for scientific machine learning. *arXiv:2001.04385*. doi: 10.21203/rs.3.rs-55125/v1

Reddi, S. J., Kale, S., and Kumar, S. (2019). On the convergence of Adam and beyond. *arXiv:1904.09237*. doi: 10.48550/arXiv.1904.09237

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *arXiv:1609.04747*. doi: 10.48550/arXiv.1609.04747

Stadler, P. F. (2002). "Fitness landscapes," in *Biological Evolution and Statistical Physics*, eds M. Lässig and A. Valleriani (Berlin; Heidelberg: Springer), 183–204. doi: 10.1007/3-540-45692-9_10

Yang, G. (2019). Wide feedforward or recurrent neural networks of any architecture are Gaussian processes. *Adv. Neural Inform. Process. Syst.* 32.

Zhang, Q., Bhattacharya, S., and Andersen, M. E. (2013). Ultrasensitive response motifs: basic amplifiers in molecular signalling networks. *Open Biol.* 3:130031. doi: 10.1098/rsob.130031