eLife

Cell Biology
Physics of Living Systems

A differentiable Gillespie algorithm for simulating chemical kinetics, parameter estimation, and designing synthetic biological circuits

Reviewed Preprint
v2 • February 26, 2025
Revised by authors

Reviewed Preprint
v1 • January 7, 2025

Krishna Rijal ✉, **Pankaj Mehta**

Department of Physics, Boston University, Boston, United States

https://en.wikipedia.org/wiki/Open_access
Copyright information

eLife Assessment

This **important** study introduces a fully differentiable variant of the Gillespie algorithm as an approximate stochastic simulation scheme for complex chemical reaction networks, allowing kinetic parameters to be inferred from empirical measurements of network outputs using gradient descent. The concept and algorithm design are **convincing** and innovative. While the proofs of concept are promising, some questions are left open about implications for more complex systems that cannot be addressed by existing methods. This work has the potential to be of significant interest to a broad audience of quantitative and synthetic biologists.

https://doi.org/10.7554/eLife.103877.2.sa4

Abstract

The Gillespie algorithm is commonly used to simulate and analyze complex chemical reaction networks. Here, we leverage recent breakthroughs in deep learning to develop a fully differentiable variant of the Gillespie algorithm. The differentiable Gillespie algorithm (DGA) approximates discontinuous operations in the exact Gillespie algorithm using smooth functions, allowing for the calculation of gradients using backpropagation. The DGA can be used to quickly and accurately learn kinetic parameters using gradient descent and design biochemical networks with desired properties. As an illustration, we apply the DGA to study stochastic models of gene promoters. We show that the DGA can be used to: (i) successfully learn kinetic parameters from experimental measurements of mRNA expression levels from two distinct *E. coli* promoters and (ii) design nonequilibrium promoter architectures with desired input-output relationships. These examples illustrate the utility of the DGA for analyzing stochastic chemical kinetics, including a wide variety of problems of interest to synthetic and systems biology.

Randomness is a defining feature of our world. Stock market fluctuations, the movement of particles in fluids, and even the change of allele frequencies in organismal populations can all be described using the language of stochastic processes. For this reason, disciplines as diverse as physics, biology, ecology, evolution, finance, and engineering have all developed tools to mathematically model stochastic processes [1–4]. In the context of biology, an especially fruitful area of research has been the study of stochastic gene expression in single cells [5–8]. The small number of molecules involved in gene expression make stochasticity an inherent feature of protein production and numerous mathematical and computational techniques have been developed to model gene expression and relate mathematical models to experimental observations [9, 10].

One prominent computational algorithm for understanding stochasticity in gene expression is the Gillespie algorithm, with its Direct Stochastic Simulation Algorithm variant being the most commonly used method [11, 12]. The Gillespie algorithm is an extremely efficient computational technique used to simulate the time evolution of a system in which events occur randomly and discretely [12]. Beyond gene expression, the Gillespie algorithm is widely employed across numerous disciplines to model stochastic systems characterized by discrete, randomly occurring events including epidemiology [13], ecology [14, 15], neuroscience [16, 17], and chemical kinetics [18, 19].

Here, we revisit the Gillespie algorithm in light of the recent progress in deep learning and differentiable programming by presenting a "fully-differentiable" variant of the Gillespie algorithm we dub the Differentiable Gillespie Algorithm (DGA). The DGA modifies the traditional Gillespie algorithm to take advantage of powerful automatic differentiation libraries (for example, PyTorch [20], Jax [21], and Julia [22]) and gradient-based optimization. The DGA allows us to quickly fit kinetic parameters to data and design discrete stochastic systems with a desired behavior. Our work is similar in spirit to other recent work that seeks to harness the power of differentiable programming to accelerate scientific simulations [23–29]. The DGA's use of differential programming tools also complements more specialized numerical methods designed for performing parameter sensitivity analysis on Gillespie simulations such as finite-difference methods [30–32], the likelihood ratio method [33–35] and pathwise derivative methods [36].

One of the difficulties in formulating a differentiable version of the Gillespie algorithm is that the stochastic systems it treats are inherently discrete. For this reason, there is no obvious way to take derivatives with respect to kinetic parameters without making approximations. As shown in **Fig. 1**, in the traditional Gillespie algorithm both the selection of the index for the next reaction and the updates of chemical species are both discontinuous functions of the kinetic parameters. To circumnavigate these difficulties, the DGA modifies the traditional Gillespie algorithm by approximating discrete operations with continuous, differentiable functions, smoothing out abrupt transitions to facilitate gradient computation via automatic differentiation (**Fig. 1**). This significant modification preserves the core characteristics of the original algorithm while enabling integration with modern deep learning techniques.

One natural setting for exploring the efficacy of the DGA is recent experimental and theoretical works exploring stochastic gene expression. Here, we focus on a set of beautiful experiments that explore the effect of promoter architecture on steady-state gene expression [37]. An especially appealing aspect of [37] is that the authors independently measured the kinetic parameters for these promoter architectures using orthogonal experiments. This allows us to directly compare the predictions of DGA to ground truth measurements of kinetic parameters. We then extend our considerations to more complex promoter architectures [38] and illustrate how the DGA can be used to design circuits with a desired input-output relation.

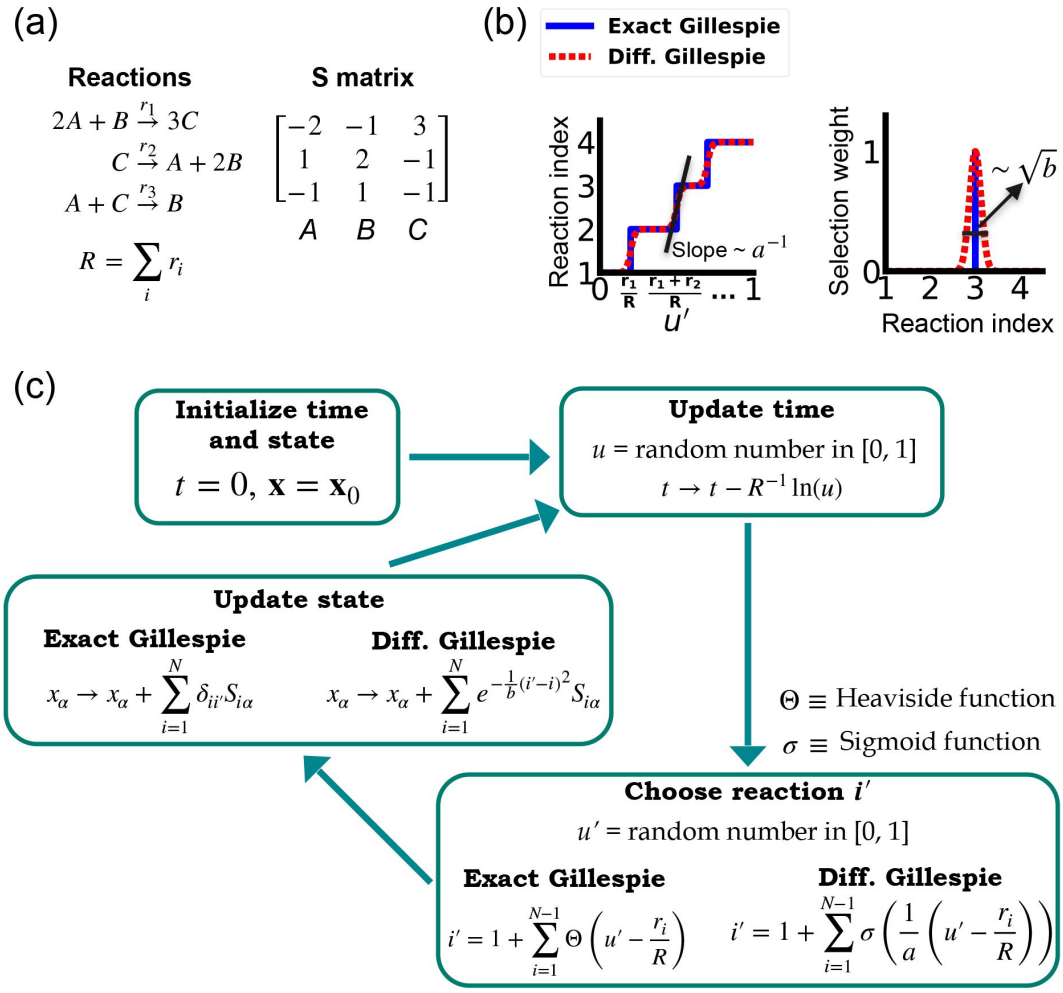


Fig. 1.

Comparison between the exact Gillespie algorithm and the DGA for simulating chemical kinetics. (a) Example of kinetics with N = 3 reactions with rates r_i(i = 1, 2, 3). (b) Illustration of the DGA's approximations: replacing the non-differentiable Heaviside and Kronecker delta functions with smooth sigmoid and Gaussian functions, respectively. (c) Flow chart comparing exact and differentiable Gillespie simulations.

I. A differentiable approximation to the Gillespie algorithm

Before proceeding to discussing the DGA, we start by briefly reviewing how the traditional Gillespie algorithm simulates discrete stochastic processes. For concreteness, in our exposition, we focus on the chemical system shown in **Fig. 1** consisting of three species, A, B, and C, whose abundances are described by a state vector $\mathbf{x} = (x_1, x_2, x_3)$. These chemical species can undergo $N = 3$ chemical reactions, characterized by rate constants, $r_i(\mathbf{x})$ where $i = 1, \ldots, 3$, and a stoichiometric matrix $S_{i\alpha}$ whose i-th row encodes how the abundance x_α of species α changes due to reaction i. Note that in what follows, we will often supress the dependence of the rates $r_i(\mathbf{x})$ on $\mathbf{x}$ and simply write r_i.

In order to simulate such a system, it is helpful to discretize time into small intervals of size $\Delta t \ll 1$. The probability that a reaction i with rate r_i occurs during such an interval is simply $r_i \Delta t$. By construction, we choose Δt to be small enough that $r_i \Delta t \ll 1$ and that the probability that a reaction occurs in any interval Δt is extremely small and well described by a Poisson process. This means that naively simulating such a process is extremely inefficient because, in most intervals, no reactions will occur.

A. Gillespie algorithm

The Gillespie algorithm circumnavigates this problem by: (i) exploiting the fact that the reactions are independent so that the rate at which *any* reaction occurs is also described by an independent Poisson process with rate $R = \Sigma_i r_i$ and (ii) the waiting time distribution $p(\tau)$ of a Poisson process with rate R is the exponential distribution $p(\tau) = Re^{-R\tau}$. The basic steps of the Gillespie algorithm are illustrated in **Fig. 1**.

The simulation begins with the initialization of time and state variables:

$$t = 0,$$
$$\mathbf{x} = \mathbf{x}_0,$$

where t is the simulation time. One then samples the waiting time distribution $p(\tau)$ for a reaction to occur to determine when the next the reaction occurs. This is done by drawing a random number u from a uniform distribution over [0, 1] and updating

$$t \to t - R^{-1} \ln(u). \tag{1}$$

Note that this time update is a fully differentiable function of the rates r_i.

In order to determine which of the reactions i' occurs after a time τ, we note that probability that reaction i occurs is simply given by $q_i = r_i/R$. Thus, we can simply draw another random number u' and choose i' such that i' equals the smallest integer satisfying

$$\sum_{i=1}^{i'} r_i/R > u'. \tag{2}$$

The reaction abundances $\mathbf{x}$ are then updated using the stoichiometric matrix

$$x_\alpha \to x_\alpha + S_{i'\alpha}. \tag{3}$$

Unlike the time update, both the choice of the reaction i' and the abundance updates are not differentiable since the choice of the reaction i' is a discontinuous function of the parameters r_i.

B. Approximating updates in the Gillespie with differentiable functions

In order to make use of modern deep learning techniques and modern automatic differentiation packages, it is necessary to modify the Gillespie algorithm in such as way as to make the choice of reaction index (Eq. (2)) and abundance updates (Eq. (3)) differentiable functions of the kinetic parameters. To do so, we rewrite Eq. (2) using a sum of Heaviside step function $\Theta(y)$ (recall $\Theta(y) = 0$ if $y < 0$ and $\Theta(y) = 1$ if $y > 0$):

$$i' = 1 + \sum_{i=1}^{N-1} \Theta\left(u' - \frac{r_i}{R}\right). \tag{4}$$

This formulation of index selection makes clear the source of non-differentiability. The derivative of the i' with respect to r_i does not exist at the transition points where the Heaviside function jumps (see **Fig. 1b**).

This suggests a natural modification of the Gillespie algorithm to make it differentiable – replacing the Heaviside function $\Theta(y)$ by a sigmoid function of the form

$$\sigma(y) = \frac{1}{1 + e^{-\frac{y}{a}}}, \tag{5}$$

where we have introduced a "hyper-parameter" a that controls the steepness of the sigmoid and plays an analogous role to temperature in a Fermi function in statistical mechanics. A larger value of a^{-1} results in a steeper slope for the sigmoid functions, thereby more closely approximating the true Heaviside functions which is recovered in the limit $a \to 0$ (see **Fig. 1**(b)). With this replacement, the index selection equation becomes

$$i' = 1 + \sum_{i=1}^{N-1} \sigma\left(\frac{1}{a}\left(u' - \frac{r_i}{R}\right)\right). \tag{6}$$

Note that in making this approximation, our index is no longer an integer, but instead can take on all real values between 0 and N. However, by making a sufficiently small, **Eq. (6)** still serves as a good approximation to the discrete jumps in **Eq. (4)**. In general, a is a hyperparameter that is chosen to be as small as possible while still ensuring that the gradient of i' with respect to the kinetic parameters r_i can be calculated numerically with high accuracy. For a detailed discussion, please see **Fig. 9** and **Appendix A**.

Since the index i' is no longer an integer but a real number, we must also modify the abundance update in **Eq. (3)** to make it fully differentiable. To do this, we start by rewriting **Eq. (3)** using the Kronecker delta δ_{ij}

(where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} \neq 0$ if $i \, j$) as

$$x_\alpha \to x_\alpha + \sum_{i=1}^{N} \delta_{ii'} S_{i\alpha} \tag{7}$$

Since i' is no longer an integer, we can approximate the Kronecker delta $\delta_{ii'}$ by a Gaussian function, to arrive at the approximate update equation

$$x_\alpha \to x_\alpha + \sum_{i=1}^{N} e^{-\frac{1}{b}(i'-i)^2} S_{i\alpha}. \qquad (8)$$

The hyperparameter b is generally chosen to be as small as possible while still ensuring numerical stability of gradients (**Fig. 9**). Note by using an abundance update of the form **Eq. (8)**, the species abundances $\mathbf{x}$ are now real numbers. This is in stark contrast with the exact Gillespie algorithm where the abundance update (**Eq. (7)**) ensures that the x_α are all integers.

C. Combining the DGA with gradient-based optimization

The goal of making Gillespie simulations differentiable is to enable the computation of the gradient of a *loss function*, $L(\boldsymbol{\theta})$, with respect to the kinetics parameters $\boldsymbol{\theta}$. A loss function quantifies the difference between simulated and desired values for quantities of interest. For example, when employing the DGA in the context of fitting noisy gene expression models, a natural choice for $L(\boldsymbol{\theta})$ is the difference between the simulated and experimentally measured moments of mRNA/protein expression (or alternatively, the Kullback-Leibler divergence between the experimental and simulated mRNA/protein expression distributions if full distributions can be measured). When using the DGA to design gene circuits the loss function can be any function that characterizes the difference between the simulated and desired values of the input-output relation.

The goal of the optimization using the DGA is to find parameters $\boldsymbol{\theta}$ that minimize the loss. The basic workflow of a DGA-based optimization is shown in **Fig. 2**. One starts with an initial guess for the parameters $\boldsymbol{\theta}_0$. One then uses DGA algorithm to simulate the systems and calculate the gradient of the loss function $\nabla_\theta L\ (\boldsymbol{\theta})$. One then updates the parameters, moving in the direction of the gradient using gradient descent or more advanced methods such as ADAM [39, 40], which uses adaptive estimates of the first and second moments of the gradients to speed up convergence to a local minimum of the loss function.

D. The price of differentiability

A summary of the DGA is shown in **Fig. 1**. Unsurprisingly, differentiability comes at a price. The foremost of these is that unlike the Gillespie algorithm, the DGA is no longer exact. The DGA replaces the exact discrete stochastic system by an approximate differentiable stochastic system. This is done by allowing both the reaction index and the species abundances to be continuous numbers. Though in theory, the errors introduced by these approximations can be made arbitrarily small by choosing the hyper-parameters a and b small enough (see **Fig. 1**), in practice, gradients become numerically unstable when a and b are sufficiently small (see **Appendix A** and **Fig. 9**).

In what follows, we focus almost exclusively on steadystate properties that probe the “bulk”, steady-state properties of the stochastic system of interest. We find the DGA works well in this setting. However, we note that the effect of the approximations introduced by the DGA may be pronounced in more complex settings such as the calculation of rare events, modeling of tail-driven processes, or dealing with non-stationary time series.

E. Implementation

A detailed explanation of how the DGA is implemented using PyTorch is given in the Appendix. In addition, all code for the DGA is available on Github at our Github repository *https://github.com/Emergent-Behaviors-in-Biology/Differentiable-Gillespie-Algorithm*.

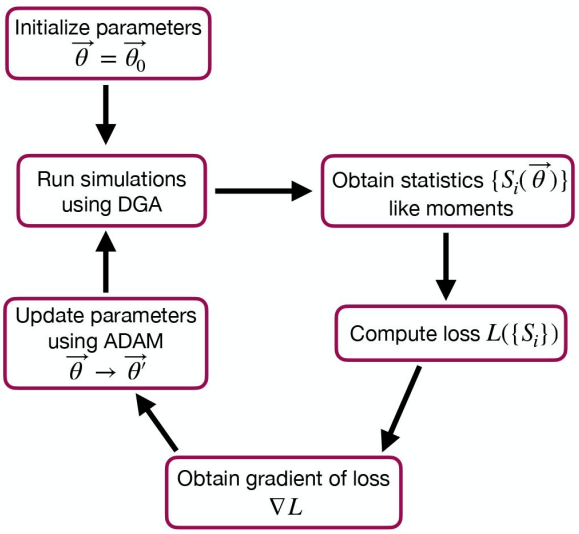


Fig. 2.

Flowchart of the parameter optimization process using the DGA. The process begins by initializing the parameters $\vec{\theta} = \vec{\theta_0}$. Simulations are then run using the DGA to obtain statistics $\{S_i(\vec{\theta})\}$ like moments. These statistics are used to compute the loss $L(\{S_i\})$, and the gradient of the loss ∇L is obtained. Finally, parameters are updated using the ADAM optimizer, and the process iterates to minimize the loss.

II. Benchmarking the dga on a simple model for stochastic gene expression

In order to better understand the DGA in the context of stochastic gene expression, we benchmarked the, DGA on a simple two-state promoter model inspired by experiments in *E. coli* [37]. This simple model had several advantages that make it well suited for exploring the performance of DGA. These include the ability to analytically calculate mRNA expression distributions and independent experimental measurements of kinetic parameters.

A. Two-state promoter model

Gene regulation is tightly regulated at the transcriptional level to ensure that genes are expressed at the right time, place, and in the right amount [41]. Transcriptional regulation involves various mechanisms, including the binding of transcription factors to specific DNA sequences, the modification of chromatin structure, and the influence of non-coding RNAs, which collectively control the initiation and rate of transcription [41–43]. By orchestrating these regulatory mechanisms, cells can respond to internal signals and external environmental changes, maintaining homeostasis and enabling proper development and function.

Here, we focus on a classic two-state promoter gene regulation [37]. Two-state promoter systems are commonly studied because they provide a simplified yet powerful model for understanding gene regulation dynamics. These systems, characterized by promoters toggling between active and inactive states, offer insights into how genes are turned on or off in response to various stimuli (see **Fig. 3(a)**). The two-state gene regulation circuit involves the promoter region, where RNA polymerase (RNAP) binds to initiate transcription and synthesize mRNA molecules at a rate r. A repressor protein can also bind to the operator site at a rate k_{on}^{R} and unbind at a rate $k_{\text{off}}^{\text{R}}$. When the repressor is bound to the operator, it prevents RNAP from accessing the promoter, effectively turning off transcription. mRNA is also degraded at a rate γ. An appealing feature of this model is that both mean mRNA expression and the Fano factor can be calculated analytically and there exist beautiful quantitative measurements of both these quantities (**Fig. 3(b)**). For this reason, we use this two-state promoters to benchmark the efficacy of DGA below.

B. Characterizing errors due to approximations in the DGA

We begin by testing the DGA to do forward simulations on the two-state promoter system described above and comparing the results to simulations performed with the exact Gillespie algorithm (see Appendix B for simulation details). **Fig. 4(a)** compares the probability distribution function (PDF) for the steady-state mRNA levels obtained from the DGA (in red) and the exact Gillespie simulation (in blue). The close overlap of these distributions demonstrates that the DGA can accurately replicate the results of the exact Gillespie simulation. This is also shown by the very close match of the first four moments $\langle m^n \rangle$ of the mRNA count between the exact Gillespie and the DGA in **Fig. 4(b)**, though the DGA systematically overestimates these moments. As observed in **Fig. 4(a)**, the DGA also fails to accurately capture the tails of the underlying PDF. This discrepancy arises because rare events result from very frequent lowprobability reaction events where the sigmoid approximation used in the DGA significantly impacts the reaction selection process and, consequently, the final simulation results.

Next, we compare the accuracy of the DGA in simulating mRNA abundance distributions across a range of simulation times (see **Fig. 4(c)**). The accuracy is quantified by the ratio of the Jensen-Shannon divergence JSD $(p_{\text{DGA}} || p_{\text{exact}}^{\text{ss}})$ between the differentiable Gillespie PDF p_{DGA} and the

Fig. 3.

Two-state gene regulation architecture. (a) Schematic of gene regulatory circuit for transcriptional repression. RNA polymerase (RNAP) binds to the promoter region to initiate transcription at a rate r, leading to the synthesis of mRNA molecules (red curvy lines). mRNA is degraded at a rate γ. A repressor protein can bind to the operator site, with association and dissociation rates k_{on}^{R} and $k_{\text{off}}^{\text{R}}$, respectively. (b) Experimental data from Ref. [37], showing the relationship between the mean mRNA level and the Fano factor for two different promoter constructs: *lac*UD5 (green squares) and 5DL1 (red squares).

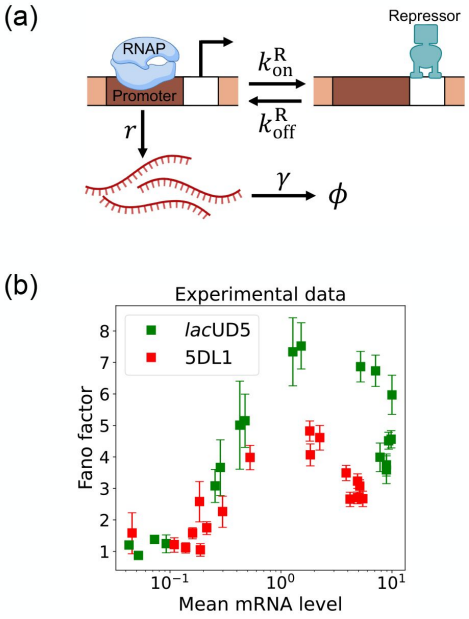
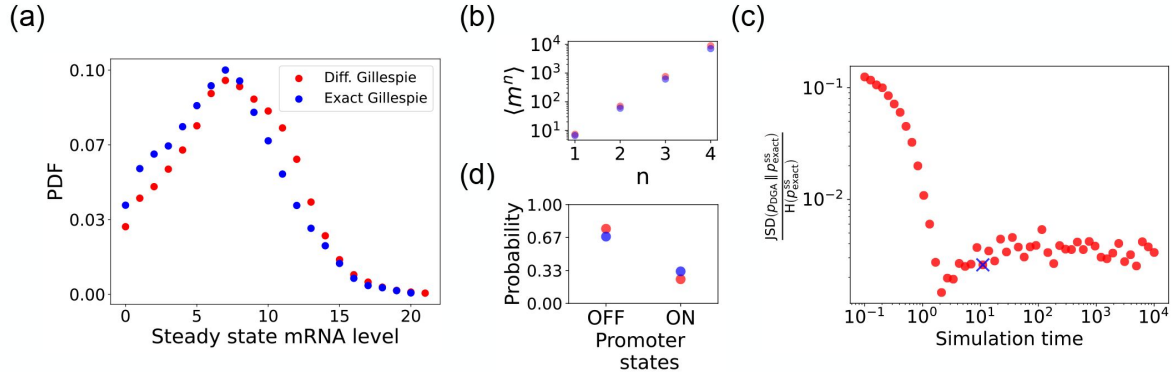


Fig. 4.

Accuracy of the DGA in simulating the two-state promoter architecture in **Fig. 3(a)**. Comparison between the DGA and exact simulations for (a) steady-state mRNA distribution, (b) moments of the steady-state mRNA distribution, and (d) the probability for the promoter to be in the "ON" or "OFF" state. (c) Ratio of the Jensen-Shannon divergence JSD $(p_{\text{DGA}}||p_{\text{exact}}^{\text{ss}})$ between the differentiable Gillespie PDF pDGA and the exact steady-state PDF $\text{PDF}\ p_{\text{exact}}^{\text{ss}}$, and the Shannon entropy H $(p_{\text{exact}}^{\text{ss}})$ of the exact steady-state PDF. In all of the plots, 2000 trajectories are used. The simulation time used in panels (a),(b), and (d) is marked by blue 'x'. Parameter values: $k_{\text{on}}^{\text{R}} = 0.5,\ k_{\text{off}}^{\text{R}} = 1.0,\ r = 10,\ \gamma = 1,\ 1/a = 200,\ \text{and}\ 1/b = 20$.



exact steady-state PDF $p^{\text{ss}}_{\text{exact}}$, and the entropy H $(p^{\text{ss}}_{\text{exact}})$ of the exact steady-state PDF. For probability distributions P and Q over the same discrete space $\mathcal{X}$, the JSD and H are defined as:

$$\begin{aligned} \text{JSD}(P \parallel Q) &= \frac{1}{2} D_{\text{KL}}(P \parallel M) + \frac{1}{2} D_{\text{KL}}(Q \parallel M) \\ \text{H}(P) &= -\sum_{x \in \mathcal{X}} P(x) \log P(x) \end{aligned} \tag{9}$$

where $M = \frac{1}{2}(P + Q)$ and D_{KL} denotes the Kullback-Leibler divergence

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \tag{10}$$

The ratio $\frac{\text{JSD}}{H}$ normalizes divergence by entropy, enabling meaningful comparison across systems. As expected, the $\frac{\text{JSD}}{\text{H}}$ ratio decreases with increasing simulation time, indicating convergence towards the steady-state distribution of the exact Gillespie simulation. By "steady-state distribution", we mean the long-term probability distribution of states that the exact Gillespie algorithm approaches after a simulation time of 10^4. The saturation of the $\frac{\text{JSD}}{\text{H}}$ ratio at approximately 0.003 for long simulation times is due to the finite values of a^{-1} and b^{-1}. In percentage terms, this ratio represents a 0.3% divergence, meaning that the DGA's approximation introduces only a 0.3% deviation from the exact distribution, relative to the total uncertainty (entropy) in the exact system.

Finally, the bar plot in **Fig. 4(d)** shows simulation results for the probability of the promoter being in the "OFF" and "ON" states as predicted by the DGA (in red) and the exact Gillespie simulation (in blue). The differentiable Gillespie over-estimates the probability of being in the "OFF" state and underestimates the probability of being in the "ON" state. Nonetheless, given the discrete nature of this system, the DGA does a reasonable job of matching the results of the exact simulations.

As we will see below, despite these errors the DGA is able to accurately capture gradient information and hence works remarkably well at gradient-based optimization of loss functions.

III. Parameter estimation using the dga

In many applications, one often wants to estimate kinetic parameters from experimental measurements of a stochastic system [44–47]. For example, in the context of gene expression, biologists are often interested in understanding biophysical parameters such as the rate at which promoters switch between states or a transcription factor unbinds from DNA. However, estimating kinetic parameters in stochastic systems poses numerous challenges because the vast majority of methods for parameter estimation are designed with deterministic systems in mind. Moreover, it is often difficult to analytically calculate likelihood functions making it difficult to perform statistical inference. One attractive method for addressing these difficulties is to combine differentiable Gillespie simulations with gradient-based optimization methods. By choosing kinetic parameters that minimize the difference between simulations and experiments as measured by a loss function, one can quickly and efficiently estimate kinetic parameters and error bars.

A. Loss Function for parameter estimation

To use the DGA for parameter estimation, we start by defining a loss function $L(\boldsymbol{\theta})$ that measures the discrepancy between simulations and experiments. In the context of the two-state promoter model (**Fig. 3**), a natural choice of loss function is the square error between the simulated and

experimentally measured mean and standard deviations of the steady-state mRNA distributions:

$$L(\boldsymbol{\theta}) = (\langle \hat{m} \rangle - \langle m \rangle)^2 + (\hat{\sigma}_m - \sigma_m)^2, \qquad (11)$$

where $\langle \hat{m} \rangle$ and $\hat{\sigma}_m$ denote the mean and standard deviation obtained from DGA simulations, and $\langle m \rangle$ and σ_m are the experimentally measured values of the same quantities. Having specified the loss function and parameters, we then use the gradient-based optimization to minimize the loss and find the optimal parameters $\hat{\boldsymbol{\theta}}$ (see **Fig. 2**). Note that in general the solution to the optimization problem need not be unique (see below).

B. Confidence intervals and visualizing loss landscapes

Given a set of learned parameters $\hat{\boldsymbol{\theta}}$ that minimize $L(\boldsymbol{\theta})$, one would also ideally like to assign a confidence interval (CI) to this estimate that reflect how constrained these parameters are. One natural way to achieve this is by examining the curvature of the loss function as the parameter θ_i varies around its minimum value, $\theta_i^{\min}$. Motivated by this, we define the 95% CIs for parameter θ_i by:

$$CI_{\theta_i} = [\theta_i^{\min} - \delta, \theta_i^{\min} + 1.96\delta_{\theta_i}] \qquad (12)$$

where

$$\delta_{\theta_i} = \left(\sqrt{\frac{\partial^2 L}{\partial \theta_i^2}} \right)^{-1} \Bigg|_{\theta_i = \theta_i^{\min}} \qquad (13)$$

and L. A detailed explanation of how to numerically estimate the CIs is given in **Appendix C**.

One shortcoming of **Eq. (13)** is that it treats each parameter in isolation and ignores correlations between parameters. On a technical level, this is reflected in the observation that the confidence intervals only know about the diagonal elements of the full Hessian $\partial_{ij}^2 L(\boldsymbol{\theta})$. This shortcoming is especially glaring when there are many sets of parameters that all optimize the loss function [48, 49]. As discuss below, this is often the case in many stochastic systems including the two-state promoter architecture in **Fig. 3**. For this reason, it is often useful to make two dimensional plots of the loss function $L(\boldsymbol{\theta})$. To do so, for each pair of parameters, we simply sample the parameters around their optimal value and forward simulate to calculate the loss function $L(\boldsymbol{\theta})$. We then use this simulations to create two-dimensional heat maps of the loss function. This allows us to identify "soft directions" in parameter space, where the loss function $L(\boldsymbol{\theta})$ changes slowly, indicating weak sensitivity to specific parameter combinations.

C. Parameter estimation on synthetic data

Before proceeding to experiments, we start by benchmarking the DGA's ability to perform parameter estimation on synthetic data generated using the two-state promoter model shown in **Fig. 3**. This model nominally has four independent kinetic parameters: the rate at which repressors bind the promoter, k_{on}^{R}; the rate at which the repressor unbinds from the promoter, $k_{\text{off}}^{\text{R}}$; the rate at which mRNA is produced, r; and the rate at which mRNA degrades, γ. Since we are only concerned with steady-state properties of the mRNA distribution, we choose to measure time in units of the off rate and set $k_{\text{off}}^{\text{R}} = 1$ in everything that follows. In **Appendix D**, we make use of exact analytical results for $\langle m \rangle$ and σ_m to show that the solution to the optimization problem specified by loss function in **Eq. (11)** is degenerate – there are many combinations of the three parameters $\{k_{\text{on}}^{\text{R}}, r, \gamma\}$ that all optimize $L(\boldsymbol{\theta})$. On the other hand, if one fixes the mRNA degradation rate γ, this degeneracy is lifted and there is a unique solution to the optimization problem for the two parameters $\{k_{\text{on}}^{\text{R}}, r\}$. We discuss both these cases below.

1. Generating synthetic data

To generate synthetic data, we randomly sample the three parameters: k_{on}^{R}, and γ within the range [0.1, 10], while keeping $k_{\text{off}}^{\text{R}}$ fixed at 1. In total, we generate 20 different sets of random parameters. We then perform exact Gillespie simulations for each set of parameters. Using these simulations, we obtain the mean $\langle m \rangle$ and standard deviation σ_m of the mRNA levels, which are then used as input to the loss function in **Eq. (11)**. We then use the DGA to estimate the parameters using the procedure outlined above and compare the resulting predictions with ground truth values for simulations.

2. Estimating parameters in the non-degenerate case

We begin by considering the case where the mRNA degradation rate γ is known and the goal is to estimate the two other parameters: the repressor binding rate k_{on}^{R} and the mRNA production rate r. As discussed above, in this case, the loss function in **Eq. (11)** has a unique minima, considerably simplifying the inference task. **Fig. 5(a)** shows a scatter plot of the learned and the true parameter values for wide variety of choices of γ. As can be seen, there is very good agreement between the true parameters and learned parameters. **Fig. 5(c)** shows that even when the true and learned parameters differ, the DGA can predict the mean $\langle m \rangle$ and standard deviation σ_m of the steady-state mRNA distribution almost perfectly (see **Appendix E** for discussion of how error bars were estimated). To better understand this, we selected a set of learned parameters: $k_{\text{on}}^{\text{R}} = 0.87,\ r = 3.83$, and γ = 2.43. We then plotted the loss function in the neigh-borhood of these parameters (**Fig. 5(b)**). As can be seen, the loss function around the true parameters is quite flat and the learned parameters live at the edge of this flat region. The flatness of the loss function reflects the fact that the mean and standard deviation of the mRNA distribution depend weakly on the kinetic parameters.

3. Estimating parameters for the degenerate case

We now estimate parameters for the two-state promoter model when all three parameters k_{on}^{R}, r, and γ are unknown. As discussed above, in this case, there are many sets of parameters that all minimize the loss function in **Eq. (11)**. **Fig. 6(a)** shows a comparison between the learned and true parameters along with a heat map of the loss function for one set of synthetic parameters (**Fig. 6(b)**). As can be seen in the plots, though the true parameters and learned parameter values differ significantly, they do so along "sloppy" directions where loss function is flat. Consistent with this intuition, we performed simulations comparing the mean $\langle \hat{m} \rangle$ and standard deviation $\hat{\sigma}_m$ of the steady-state mRNA levels using the true and learned parameters and found near-perfect agreement across all of the synthetic data (**Fig. 6(c)**).

D. Parameter estimation on experimental data

In the previous section, we demonstrated that our DGA can effectively obtain parameters for synthetic data. However, real experimental data often contains noise and variability, which can complicate the parameter estimation process. To test the DGA in this more difficult setting, we reanalyze experiments by Jones *et al.* [37] which measured how mRNA expression changes in a system well described by the two-state gene expression model in **Fig. 3**. In these experiments, two constitutive promoters *lac*UD5 and 5DL1 (with different transcription rates r) were placed under the control of a LacI repressor through the insertion of a LacI binding site. By systematically varying LacI concentrations, the authors were able to adjust the repressor binding rate k_{on}^{R}. mRNA fluorescence in situ hybridization (FISH) was employed to measure mRNA expression, providing data on both mean expression levels $\langle m \rangle$ and the variability as quantified by the Fano factor $f = \sigma_m^2/\langle m \rangle$ for both promoters (see **Fig. 3(b)**).

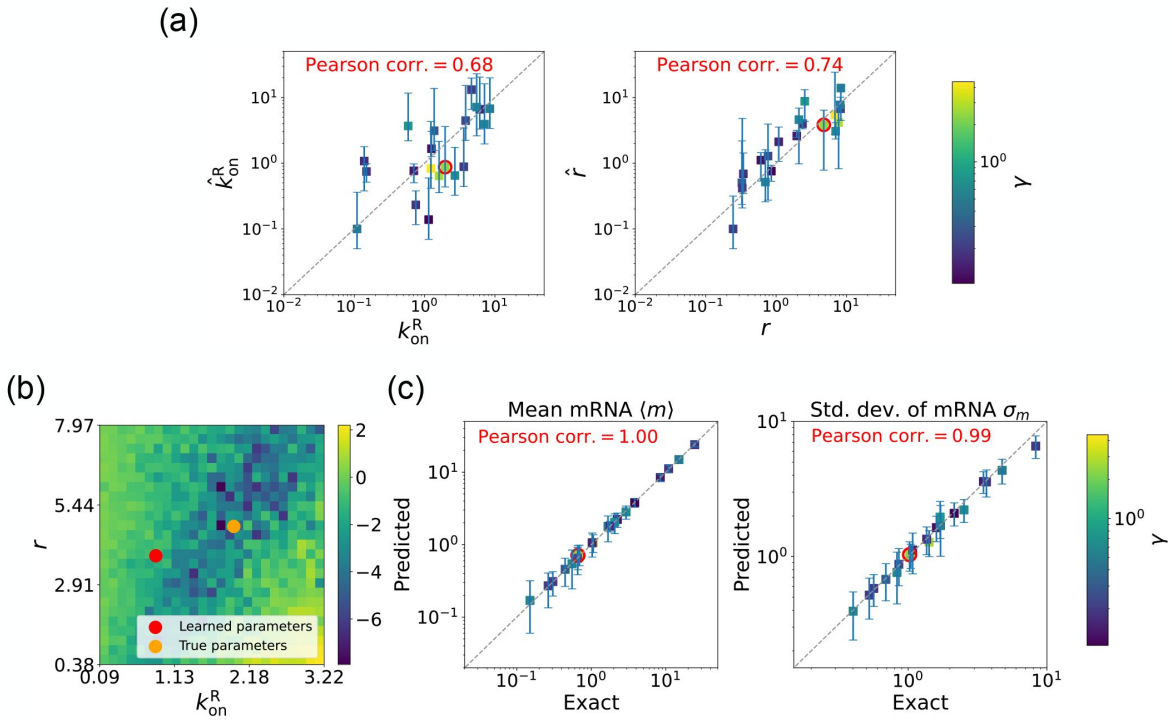


Fig. 5.

Gradient-based learning via DGA is applied to the synthetic data for the gene expression model in **Fig. 3(a)**. Parameters $k_{\text{off}}^{\text{R}}$ are fixed at 1, with 1/a = 200 and 1/b = 20 for a simulation time of 10. (a) Scatter plot of true vs. inferred parameters ($\hat{k}_{\text{on}}^{\text{R}}$ and $\hat{r}$) with γ constant. Error bars are 95% CIs. Panel (b) plots the logarithm of the loss function near a learned parameter set (shown in red circles in (a)), showing insensitivity regions. Panel (c) compares true and predicted mRNA mean and standard deviation with 95% CIs.

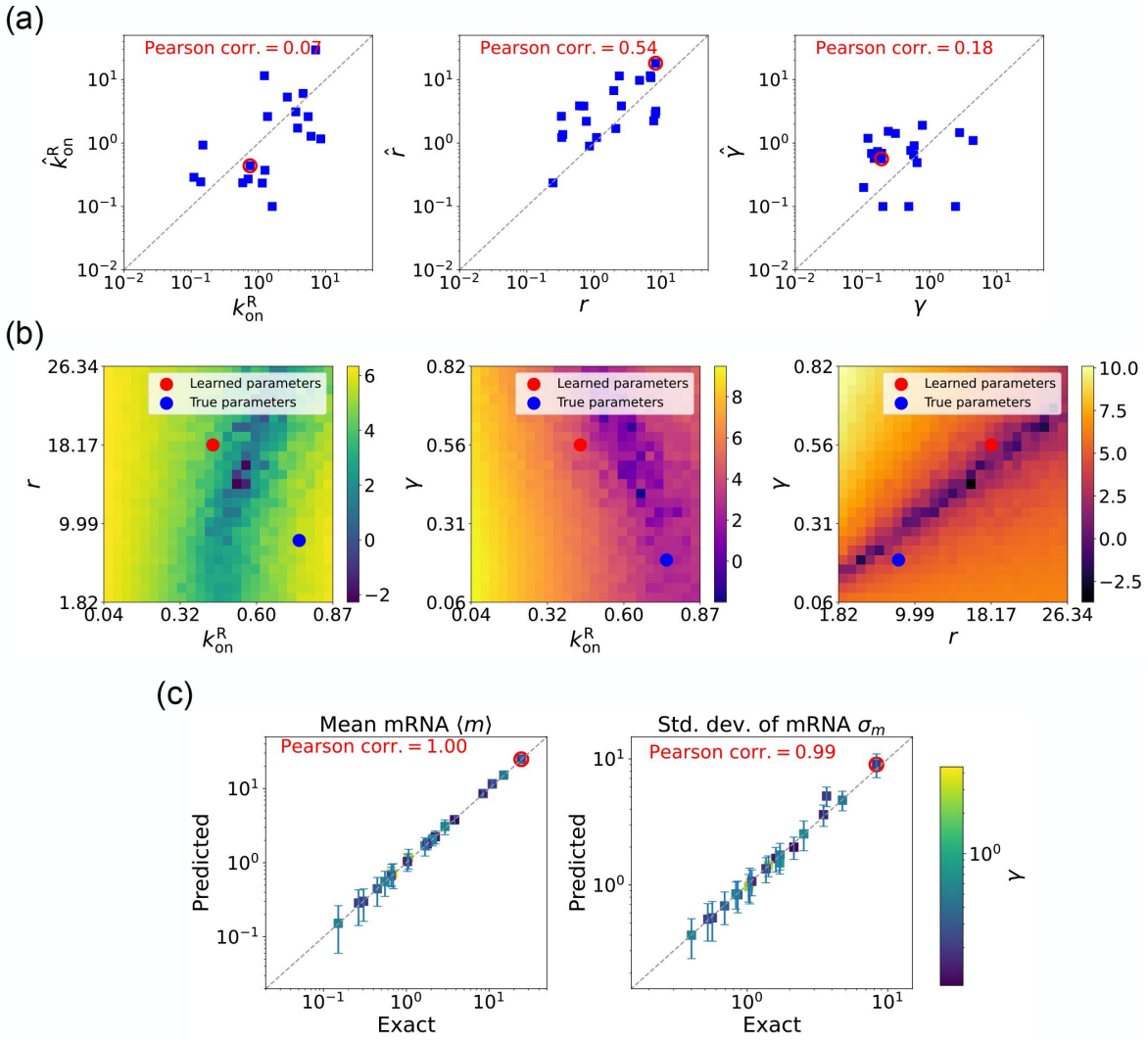


Fig. 6.

Gradient-based learning via DGA is applied to the synthetic data for the gene expression model in **Fig. 3(a)**. Parameters $k_{\text{off}}^{\text{R}}$ are fixed at 1, with 1/a = 200 and 1/b = 20 for a simulation time of 10. (a) Scatter plot of true vs. inferred parameters ($\hat{k}_{\text{on}}^{\text{R}}$, $\hat{r}$, and γ). Error bars are 95% CIs. Panel (b) plots the logarithm of the loss function near a learned parameter set (shown in red circles in (a)), showing insensitivity regions. Panel (c) compares true and predicted mRNA mean and standard deviation with 95% CIs.

Given a set of measurements of the mean and Fano factor $\{\langle m\rangle_i, f_i^m\}$ for a promoter (*lac*UD5 and 5DL1), we construct a loss function of the form:

$$L = \sum_{i=1}^{N}(\langle\hat{m}\rangle_i - \langle m\rangle_i)^2 + \sum_{i=1}^{N}(\hat{\sigma}_i^m - \sqrt{f_i^m\langle m\rangle_i})^2, \quad (14)$$

where *i* runs over data points (each with a different lac repressor concentration) and $\langle\hat{m}\rangle_i$ and $\hat{\sigma}_i^m$ are the mean and standard deviation obtained from a sample of DGA simulations. This loss function is chosen because, at its minimum, $\langle\hat{m}\rangle_i = \langle m\rangle_i$ and $\hat{\sigma}_i^m = \sqrt{f_i^m\langle m\rangle_i}$ for all *i*.

As above, we set $k_{\text{off}}^{\text{R}} = 1$, and focus on estimating the other three parameters $\{r, \gamma, k_{\text{on}}^{\text{R}}\}$. When performing our gradient-based optimization, we assume that the transcription rate *r* and the mRNA degradation rate *γ* are the same for all data points *i*, while allowing k_{on}^{R} to vary across data points *i*. This reflects the fact that k_{on}^{R} is a function of the lac repressor concentration which, by design, is varied across data points (see **Appendix F** for details on how this optimization is implemented and calculation of error bars).

The results of this procedure are summarized in **Fig. 7**. We find that for the *lac*UD5 promoter $\hat{r} = 90.25, \hat{\gamma} = 6.20$, and that $\hat{k}_{\text{on}}^{\text{R}}$ varies from a minimum value of 0.18 to a maximum value of 99.0. For the 5DL1 promoters $\hat{r} = 87.48$ and $\hat{\gamma} = 9.80$ and $\hat{k}_{\text{on}}^{\text{R}}$ varies between 3.64 and 99.0. Recall that we have normalized all rates to the repressor unbinding rate $k_{\text{off}}^{\text{R}} = 1$. These values indicate that mRNA transcription occurs much faster compared to the unbinding of the repressor, suggesting that once the promoter is in an active state, it produces mRNA rapidly. The relatively high mRNA degradation rates indicate a mechanism for fine-tuning gene expression levels, ensuring that mRNA does not persist too long in the cell, which could otherwise lead to prolonged expression even after promoter deactivation.

As expected, the repressor binding rates decrease with the mean mRNA level (see **Fig. 7(c)**). The broad range of repressor binding rates shows that the system can adjust its sensitivity to repressor concentration, allowing for both tight repression and rapid activation depending on the cellular context.

Fig. 7(a) shows a comparison between the predictions of the DGA (solid curves) and the experimental data (squares) for mean mRNA levels ⟨*m*⟩ and the Fano factor *f*. The theoretical curves are obtained by using analytical expression for ⟨*m*⟩ and *f* from [37] with parameters estimated from the DGA. We find that for the *lac*UD5 and the 5DL1 promoters, the mean percentage errors for predictions of the Fano factor are 25% and 28% respectively (see Appendix F).

An appealing feature of [37] is that the authors performed independent experiments to directly measure the normalized transcription rate *r*/*γ* (namely the ratio of the transcription rate and the mRNA degradation rate). This allows us to compare the DGA predictions for these parameters to ground truth measurements of kinetic parameters. z. 7(b), the predictions of the DGA agree remarkably well for both the *lac*UD5 and 5DL1 promoters.

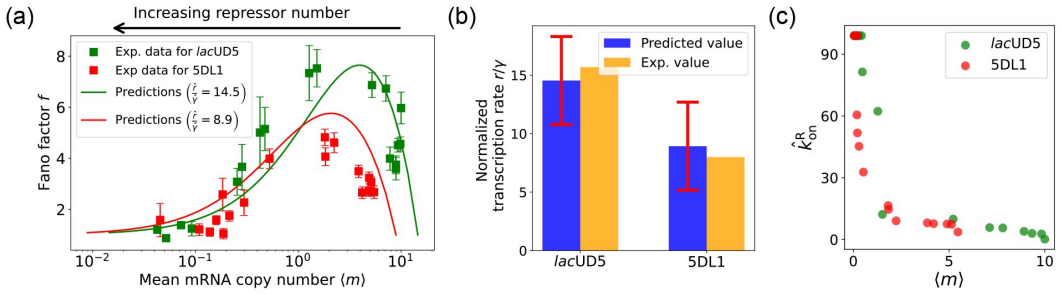


Fig. 7.

Fitting of experimental data from Ref. [37] using the DGA. (a) Comparison between theoretical predictions from the DGA (solid curves) and experimental values of mean and the Fano factor for the steady-state mRNA levels are represented by square markers, along with the error bars, for two different promoters, *lac*UD5 and 5DL1. Solid curves are generated by using DGA to estimate $\hat{r}$, $\hat{\gamma}$, and $\{\hat{k}_{on}^{R}\}$ and using this as input to exact analytical formulas. (b) Comparison between the inferred values of $\frac{\hat{r}}{\hat{\gamma}}$ using DGA with experimentally measured values of this parameter from Ref. [37]. (c) Inferred $\hat{k}_{on}^{R}$ values as a function of the mean mRNA level.

IV. Designing gene regulatory circuits with desired behaviors

Another interesting application of the DGA is to design stochastic chemical or biological networks that exhibit a particular behavior. In many cases, this design problem can be reformulated as identifying choices of parameter that give rise to a desired behavior. Here, we show that the DGA is ideally suited for such a task. We focus on designing the input-output relation of a four state promoter model of gene regulation [38]. We have chosen this more complex promoter architecture because, unlike the two-state promoter model analyzed above, it allows for nonequilibrium currents. In making this choice, we are inspired by numerous recent works have investigated how cells can tune kinetic parameters to operate out of equilibrium in order to achieve increased sharpness/sensitivity [38, 50–52].

A. Model of nonequilibrium promoter

We focus on designing the steady-state input-output relationship of the four-state promoter model of gene regulation model shown in **Fig. 8(a)** [38]. The locus can be in either an "ON" state where mRNA is transcribed at a rate r or an "OFF" state where the locus is closed and there is no transcription. In addition, a transcription factor (assumed to be an activator) with concentration $[c]$ can bind to the locus with a concentration dependent rate $[c]k_b$ in the "OFF" state and a rate $[c]\eta_{ba}k_b$ in the "ON" rate. The activator can also unbind at a rate k_u in the "OFF" state and a rate $\eta_{ua}k_u$ in the "ON" state. The average mRNA production rate (averaged over many samples) in this model is given by

$$\langle \bar{r} \rangle = r(\pi_2 + \pi_3) \qquad (15)$$

where π_s (s = 2, 3) is the steady-state probability of finding the system in each of the "ON" states (see **Fig. 8(a)**).

Such promoter architectures are often studied in the context of protein gradient-based development [38, 53, 54]. One well-known example of such a gradient is the dorsal protein gradient in Drosophila, which plays a crucial role in determining the spatial boundaries of gene expression domains during early embryonic development. In this context, the sharpness of the response as a function of activator concentration is a critical aspect. High sharpness ensures that the transition between different gene expression domains occurs over a very narrow region, leading to well-defined and precise boundaries. Inspired by this, our objective is to determine the parameters such that the variation in $\langle \bar{r} \rangle$ as a function of the activator concentration $[c]$ follows a desired response. We consider the two target responses (shown in **Fig. 8(b)**) of differing sharpness, which following [38] we quantify as max $\left(\frac{\partial \langle \bar{r} \rangle}{\partial [c]}[c]\right)$. For simplicity, we use 6th-degree polynomials to model the input-output functions, with the axis plotted on a logarithmic scale. We note that our results do not depend on this choice and any other functional form works equally well.

B. Loss function

In order to use the DGA to learn a desired input-output relation, we must specify a loss function that quantifies the discrepancy between the desired and actual responses of the promoter network. To construct such a loss function, we begin by discretizing the activator concentration into N = 10 logarithmically spaced points, $[c]_i$, where i = 1, 2, ..., N. For each $[c]_i$, we denote the

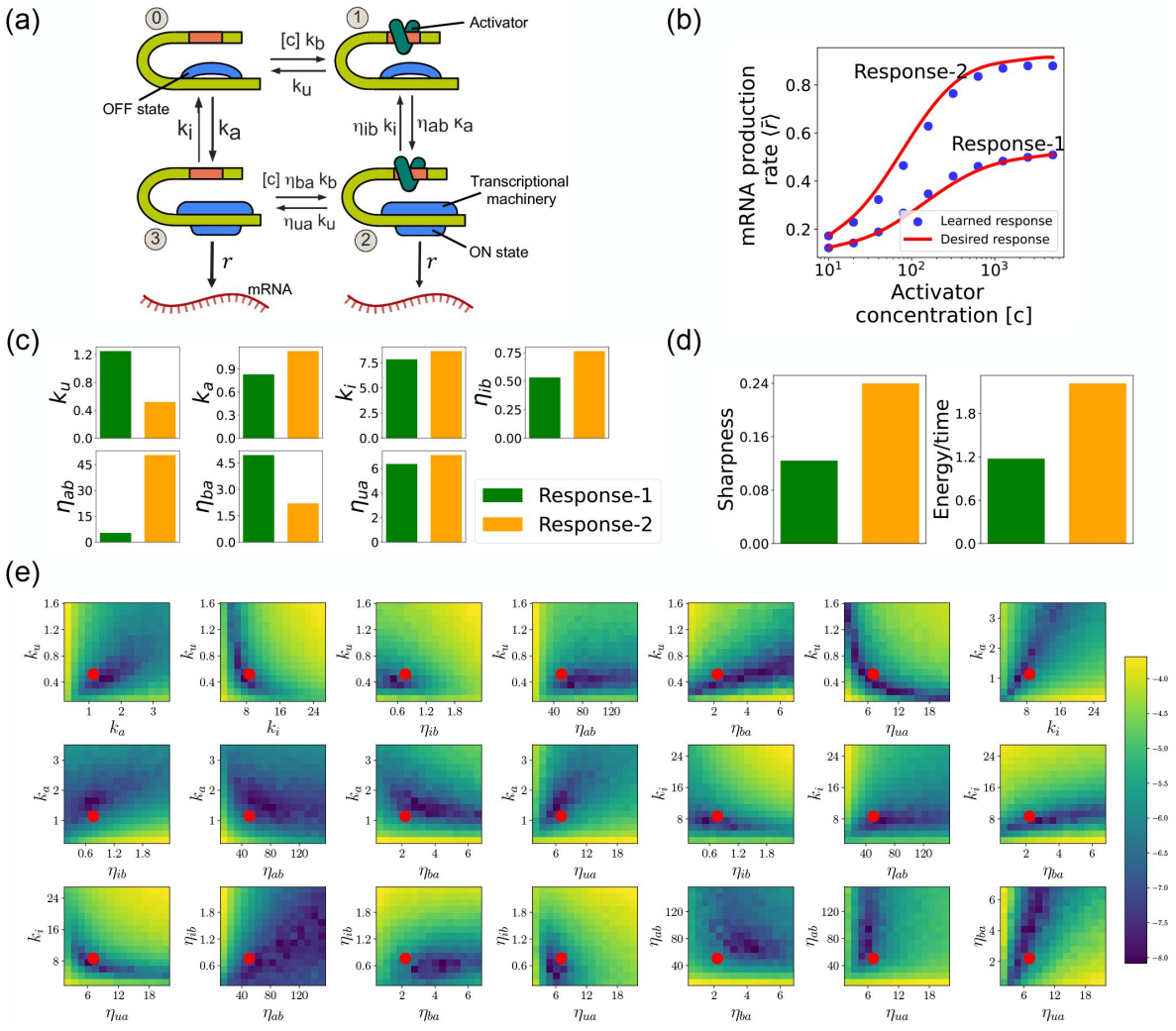


Fig. 8.

Design of the four-state promoter architecture using the DGA. (a) Schematic of four-state promoter model. (b) Target input-output relationships (solid curves) and learned input-output relationships (blue dots) between activator concentration [c] and average mRNA production rate. (c) Parameters learned by DGA for the two responses in (b). (d) The sharpness of the response $\frac{d\langle \bar{r} \rangle}{d[c]}[c]$ [c], and the energy dissipated per unit time for two responses in (b). (e) Logarithm of the loss function for the learned parameter set for Response-2, revealing directions (or curves) of insensitivity in the model's parameter space. The red circles are the learned parameter values.

corresponding average mRNA production rate $\langle \bar{r} \rangle_i$ (see Eq. (15)). After discretization, the loss function is simply the square error between the desired response, $\langle \bar{r} \rangle_i$, and the current response, $\langle \hat{\bar{r}}(\boldsymbol{\theta}) \rangle_i$, of the circuit

$$L = \sum_{i=1}^{N} (\langle \hat{\bar{r}}(\boldsymbol{\theta}) \rangle_i - \langle \bar{r} \rangle_i)^2, \qquad (16)$$

where $\langle \hat{\bar{r}}(\boldsymbol{\theta}) \rangle_i$ denotes the predicted average mRNA production rates obtained from the DGA simulations given the current parameters $\boldsymbol{\theta}$. To compute $\langle \hat{\bar{r}} \rangle_i$ for a concentration $[c]_i$, we perform n = 600 DGA simulations (indexed by capital letters A = 1, ..., n) using the DGA and use these simulations to calculate the fraction of time spent in transcriptionally active states (states s = 2 and s = 3 in **Fig. 8(a)**). If we denote the fraction of time spent in state s in simulation A by w_s^A, then we can calculate the probability π_s of being in state s by

$$\pi_s = \frac{1}{n} \sum_{A=1}^{n} w_s^A \qquad (17)$$

and use **Eq. (15)** to calculate $\langle \hat{\bar{r}}(\boldsymbol{\theta}) \rangle_i$

As before, we optimize this loss using gradient descent (see **Fig. 2**). We assume that the transcription rate r is known (this just corresponds to an overall scaling of mRNA numbers). Since we are concerned only with steady-state properties, we fix the activator binding rate to a constant value, k_b = 0.02. This is equivalent to measuring time in units of k_b^{-1}. We then use gradient descent to optimize the remaining seven parameters governing transitions between promoter states.

C. Assessing circuits found by the DGA

Fig. 8(b) shows a comparison between the desired and learned input-output relations. This is good agreement between the learned and desired responses, showing that the DGA is able to design dose-response curves with different sensitivities and maximal values. **Fig. 8(c)** shows the learned parameters for both response curves. Notably, the degree of activation resulting from transcription factor binding, denoted by η_{ab}, is substantially higher for the sharper response (Response-2). In contrast, the influence on transcription factor binding due to activation, represented by η_{ba}, is reduced for the sharper response curve. Additionally, the unbinding rate k_u is observed to be lower for the sharper response. However, it is essential to approach these findings with caution, as the parameters are highly interdependent. These interdependencies can be visualized by plotting the loss function around the optimized parameter values. **Fig. 8(e)** shows two dimensional heat maps of the loss function for Response-2. There are seven free parameters, resulting in a total of 21 possible 2D slices of the loss function within the 7-dimensional loss landscape.

The most striking feature of these plots is the central role played by the parameters η_{ab} and η_{ua} which must both be high, suggesting that the sharpness in Response-2 may result from creating a high-flux nonequilibrium cycle through the four promoter states (see **Fig. 8(a)**). This observation is consistent with recent works suggesting that creating such nonequilibrium kinetics represents a general design principle for engineering sharp responses [38, 50–52]. To better understand if this is indeed what is happening, we quantified the energy dissipation per unit time (power consumption), Φ, in the nonequilibrium circuit. The energetic cost of operating biochemical networks can be quantified using ideas from nonequilibrium thermodynamics using a generalized Ohm's law of the form [38, 55–59]

$$\Phi = J \Delta \mu \qquad (18)$$

where we have defined a nonequilibrium drive

$$\Delta\mu = \ln\left(\frac{\eta_{ab}\eta_{ua}}{\eta_{ib}\eta_{ba}}\right) \tag{19}$$

and the nonequilibrium flux

$$J = \pi_0 k_b[c] - \pi_1 k_u, \tag{20}$$

where π_0 and π_1 are the probabilities of finding the system in state 0 and 1, respectively. **Fig. 8(d)** shows a comparison between energy consumption and sharpness of the two learned circuits. Consistent with the results of [38], we find that the sharper response curve is achieved by consuming more energy.

V. Conclusion

In this paper, we introduced a fully differentiable variant of the Gillespie algorithm, the DGA. By integrating differentiable components into the traditional Gillespie algorithm, the DGA facilitates the use of gradient-based optimization techniques, such as gradient descent, for parameter estimation and network design. The ability to smoothly approximate the discrete operations of the traditional Gillespie algorithm with continuous functions facilitates the computation of gradients via both forwardmode and reverse-mode automatic differentiation, foundational techniques in machine learning, and has the potential to significantly expand the utility of stochastic simulations. Our work demonstrates the efficacy of the DGA through various applications, including parameter learning and the design of simple gene regulatory networks.

We benchmarked the DGA's ability to accurately replicate the results of the exact Gillespie algorithm through simulations on a two-state promoter architecture. We found the DGA could accurately approximate the moments of the steady-state distribution and other major qualitative features. Unsurprisingly, it was less accurate at capturing information about the tails of distributions. We then demonstrated that the DGA could be accurately used for parameter estimation on both simulated and real experimental data. This capability to infer kinetic parameters from noisy experimental data underscores the robustness of the DGA, making it a potentially powerful computation tool for real-world applications in quantitative biology. Furthermore, we showcased the DGA's application in designing biological networks. Specifically, for a complex four-state promoter architecture, we learned parameters that enable the gene regulation network to produce desired input-output relationships. This demonstrates how the DGA can be used to rapidly design complex biological systems with specific behaviors. We expect computational design of synthetic circuits with differentiable simulations to become an increasingly important tool in synthetic biology.

There remains much work still to be done. In this paper, we focused almost entirely on properties of the steady-states. However, a powerful aspect of the traditional Gillespie algorithm is that it can be used to simulate dynamical trajectories. How to adopt the DGA to utilize dynamical data remains an extremely important open question. In addition, it will be interesting to see if the DGA can be adapted to understand the kinetic of rare events. It will also be interesting to compare the DGA with other recently developed approximation methods such as those based on tensor networks [60, 61]. Beyond the gene regulatory networks, extending the DGA to handle larger and more diverse datasets will be crucial for applications in epidemiology, evolution, ecology, and neuroscience. On a technical level, this may be facilitated by developing more sophisticated smoothing functions and adaptive algorithms to improve numerical stability and convergence.

The DGA could also be extended to stochastic spatial systems by incorporating reaction-diffusion master equations or lattice-based models. Its differentiability may enable efficient optimization of spatially heterogeneous reaction parameters. However, such extensions may need to address computational scalability and stability in high-dimensional spaces, especially in processes such as diffusion-driven pattern formation or spatial gene regulation.

Acknowledgements

This work was supported by NIH NIGMS R35GM119461 to P.M. and Chan-Zuckerburg Institute Investigator grant to P.M. The authors also acknowledge support from the Shared Computing Cluster (SCC) administered by Boston University Research Computing Services. We would also like to thank the Mehta and Kondev groups for useful discussions.

Appendix A: Balancing accuracy and stability: Hyperparameter tradeoffs

In this section, we explore the tradeoffs involved in tuning the hyperparameters a and b in the DGA. These hyperparameters are crucial for balancing the accuracy and numerical stability of the DGA in approximating the exact Gillespie algorithm.

The hyperparameter a^{-1} controls the steepness of the sigmoid function used to approximate the Heaviside step function in reaction selection. Similarly, b^{-1} determines the sharpness of the Gaussian function used to approximate the Kronecker delta function in abundance updates. A larger value of a^{-1} or b^{-1} results in a steeper sigmoid or Gaussian function, thus more closely approximating the discrete functions in the exact Gillespie algorithm.

1. Accuracy of the forward DGA simulations

To assess the impact of these hyperparameters on the accuracy of the DGA, we measure the ratio of the Jensen–Shannon divergence between the DGA-generated PDF p_{DGA} and the exact PDF p_{exact}, normalized by the entropy $\mathrm{H}(p_{\mathrm{exact}})$ of the exact steady-state PDF (see Eq. (9)). This ratio, $\frac{\mathrm{JSD}(p_{\mathrm{DGA}}\|p_{\mathrm{exact}})}{\mathrm{H}(p_{\mathrm{exact}})}$, provides a measure of how closely the DGA approximates the exact Gillespie algorithm.

Fig. 9(a) and **9(b)** show the ratio $\frac{\mathrm{JSD}}{\mathrm{H}}$ as a function of the sharpness parameters a^{-1} and b^{-1}. In panel (a), b^{-1} is fixed at 20, and a^{-1} is varied. In panel (b), a^{-1} is fixed at 200, and b^{-1} is varied. Some key insights can be drawn from these plots.

First, as a^{-1} or b^{-1} increases, the ratio $\frac{\mathrm{JSD}}{\mathrm{H}}$ decreases, indicating that the DGA's approximation becomes more accurate. This is because steeper sigmoid and Gaussian functions better mimics the discrete steps of the exact Gillespie algorithm. Interestingly, while the ratio decreases for both parameters, a^{-1} plateaus at high values, whereas b^{-1} rises at high values. This plateau occurs because the sigmoid function used for reaction selection becomes so steep that it effectively becomes a step function, beyond which further steepening has negligible impact. As b^{-1} becomes very large, the width of the Gaussian function used for abundance updates becomes extremely narrow. In such a scenario, it becomes increasingly improbable for the chosen reaction index to fall within this narrow width, especially because the reaction indices are not exact integers but are instead near-integer continuous values. Therefore, as b^{-1} becomes very large, the discrepancy between the DGA-generated probabilities and the exact probabilities widens, causing the ratio $\frac{\mathrm{JSD}}{\mathrm{H}}$ to

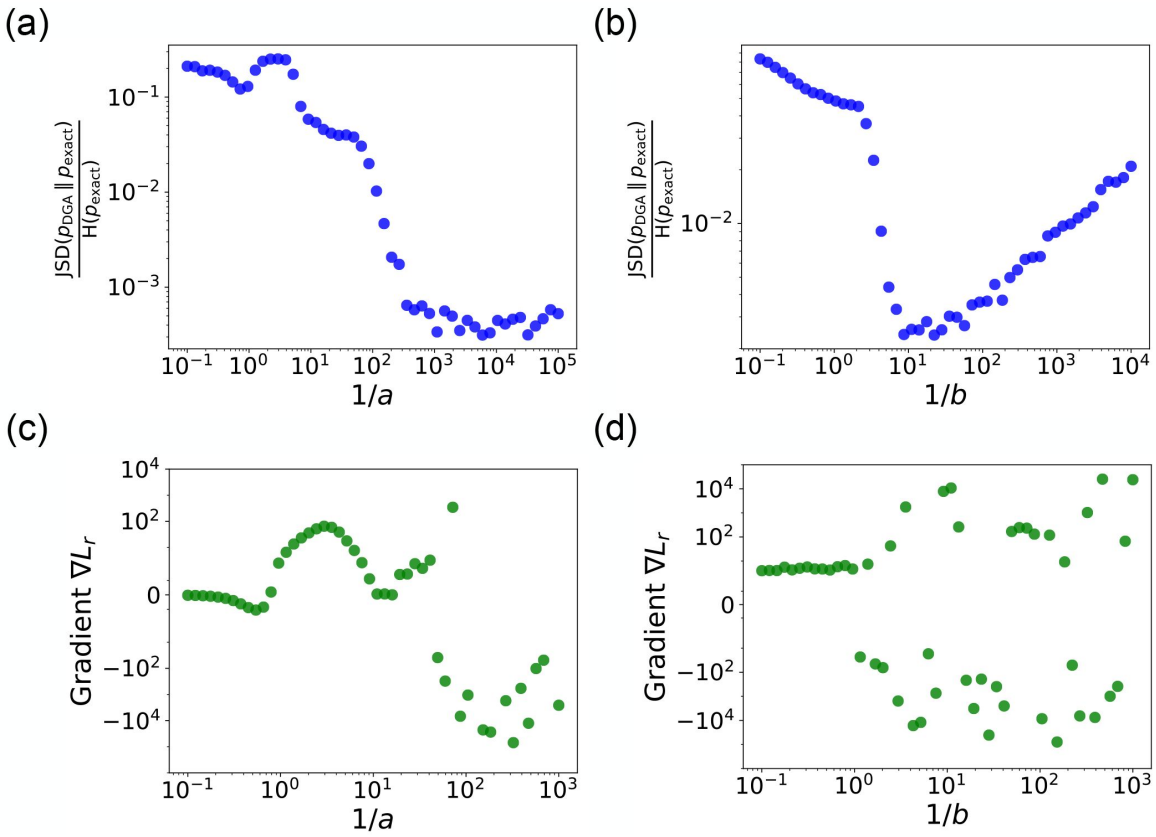


Fig. 9.

In panels (a) and (b), we plot the ratio of the Jensen-Shannon divergence JSD $(p_{\mathrm{DGA}}||p^{\mathrm{ss}}_{\mathrm{exact}})$ between the differentiable Gillespie PDF pDGA and the exact steady-state PDF $p^{\mathrm{ss}}_{\mathrm{exact}}$, and the Shannon entropy $\mathrm{H}(p^{\mathrm{ss}}_{\mathrm{exact}})$ of the exact steady-state PDF, as a function of the two sharpness parameters 1/a and 1/b. In panel (a), 1/b = 20; in panel (b), 1/a = 200. The simulation time is set to 10. In panels (c) and (d), for these same values, we show the gradient ∇L_r of the loss function L with respect to the parameter r near the true parameter values. In all the plots, the values of the rates are: $k^{\mathrm{R}}_{\mathrm{on}} = 0.5,\ k^{\mathrm{R}}_{\mathrm{off}} = 1,\ r = 10$, and γ = 1. 5000 trajectories are used to obtain the PDFs, while 200 trajectories are used to obtain the gradients.

2. Stability of the backpropagation of DGA simulations

Numerical stability for gradient computation is crucial. Therefore, it is important to examine how the gradient behaves as a function of the hyperparameters. Panels (c) and (d) of **Fig. 9** provide insights into this behavior. The plots show the gradient ∇L_r of the loss function L with respect to the parameter r near the true parameter values. The gradients are computed for the loss function in **Eq. (11)** with $\langle m \rangle$ and σ_m equal to 8 and 2.5 respectively, for the parameter values $k_{\text{on}}^{\text{R}} = 0.5, k_{\text{off}}^{\text{R}} = 1$, r = 10, and γ = 1. With these parameter values, the true mean and standard deviation are equal to 6.67 and 3.94 respectively.

As a^{-1} or b^{-1} increases, the gradients become more accurate, but their numerical stability can be compromised. This is evidenced by the increased variability and erratic behavior in the gradients at very high sharpness values (see **Figs. 9(c)** and **9(d)**). Hence, very large values of a^{-1} or b^{-1} lead to oscillations and convergence issues, highlighting the need for a balance between accuracy and stability.

The tradeoff between accuracy and numerical stability is evident. This necessitates careful tuning of a and b to ensure stable and efficient optimization. In practice, this involves selecting values that provide sufficient approximation quality without compromising the stability of the gradients.

Appendix B: Implementation of the DGA in PyTorch

This section explains the implementation of the DGA used to simulate the two-state promoter model. The implementation can be adapted to any model where the stoichiometric matrix and propensities are known. The model involves promoter state switching and mRNA production/degradation (**Fig. 3(a)**), and the algorithm is designed to handle these sub-processes effectively.

Stoichiometric matrix

The stoichiometric matrix is a key component in modeling state changes due to reactions. Each row represents a reaction, and each column corresponds to a state variable (promoter state and mRNA level). The matrix for the two-state promoter model includes:

- **Reaction 1:** Promoter state transitions from the OFF state (−1) to the ON state (+1).
- **Reaction 2:** Production of mRNA.
- **Reaction 3:** Promoter state transitions from the ON state (+1) to the OFF state (−1).
- **Reaction 4:** Degradation of mRNA.

The stoichiometric matrix S for this model is:

$$S = \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ -2 & 0 \\ 0 & -1 \end{pmatrix}$$

In this matrix, the rows correspond to the reactions listed above, and the columns represent the promoter state and mRNA level, respectively.

Propensity calculations

The levels vector is a 2-dimensional vector where the first element represents the promoter state (−1 or +1) and the second element represents the mRNA number *m*. Propensities are the rates at which reactions occur, given the current state of the system. In our PyTorch implementation, the propensities are calculated using the following expressions:

propensities = torch.stack([kon * torch.sigmoid(−c * levels[0]), # Promoter state switching from −1 to +1 r * torch.sigmoid(−c * levels[0]), # mRNA production torch.sigmoid(c * levels[0]), # Promoter state switching from +1 to −1 g * levels[1] # mRNA degradation])

Each propensity corresponds to a different reaction:

- **Promoter state switching from −1 to +1:** The value of kon * torch.sigmoid(−c * levels[0]) is around $k_{\mathrm{on}}^{\mathrm{R}}$ when the levels[0] (promoter state) is around −1 and close to zero when levels[0] is around +1. The sigmoid function ensures a smooth transition, allowing differentiability and preventing abrupt changes in rates. The constant *c* controls the sharpness of the sigmoid function.
- **mRNA production:** The rate is proportional to *r* and modulated by the same sigmoid function, torch.sigmoid(−c * levels[0]), such that the rate is equal to *r* only when the promoter is in −1 state.
- **Promoter state switching from +1 to −1:** The rate is set to 1 and modulated by the sigmoid function, torch.sigmoid(c * levels[0]), such that the rate is equal to 1 only when the promoter is in +1 state.
- **mRNA degradation:** The rate is proportional to the current mRNA level, g * levels[1], reflecting the natural decay of mRNA with rate $m\gamma$.

Using the sigmoid function in the propensity calculations is crucial for ensuring smooth and differentiable transitions between states. This smoothness is essential for gradient-based optimization methods, which rely on continuous and differentiable functions to compute gradients effectively. Without the sigmoid function, the propensity rates could change abruptly, leading to numerical instability and difficulties in optimizing the model parameters.

Reaction selection function

We define a function reaction_selection that selects the next reaction to occur based on the transition points and a random number between [0,1]. The function basically implements using Eq. (6). The transition points are first calculated from the cumulative sum of reaction rates normalized to the total rate.

State jump function

We define another function state_jump that calculates the state change vector when a reaction occurs. It uses a Gaussian function to smoothly transition between states based on the selected reaction index and the stoichiometry matrix (see Eq. (8)).

Gillespie simulation function

The main gillespie_simulation function uses the previously described functions, each with specific roles, to perform the actual simulation step-by-step, as shown in **Fig. 1**. This function iterates through the number of simulations, updating the system's state and the propensities of each reaction at each step.

Appendix C: Estimating confidence intervals for parameters

In this section, we describe the methodology used to estimate the confidence intervals for the parameters using polynomial fitting and numerical techniques. This approach uses the results of multiple simulations to determine the range within which the parameter θ_i is likely to lie, based on the curvature of the loss function around its minimum value. Specifically, we perform the following steps:

1. **Parameter initialization**: Set up and initialize the necessary parameters. This includes defining the number of evaluation points, the number of simulations, the simulation time, and the hyper-parameters a^{-1}, b^{-1}, and c.
2. **Range generation**: For each set of learned parameters $\hat{\boldsymbol{\theta}}$, generate a range of values for the parameter of interest θ_i while keeping the other parameters fixed at their learned values. Let the learned value of the parameter θ_i be $\hat{\theta}_i$. Then the range of evaluation is approximately $[0.2\hat{\theta}_i, 2\hat{\theta}_i]$, depending on the flatness of the loss landscape around its minimum.
3. **Simulation and loss calculation**: For each value in this range, perform forward DGA simulations to calculate the mean and standard deviation of the results, using which the loss function is computed and stored.
4. **Polynomial fitting**: Fit a polynomial of degree 6 to the computed loss values across the range of θ_i. Compute the first and second derivatives of the fitted polynomial to identify the minimum and evaluate the curvature.
5. **Identifying minimum**: Identify the valid minimum $\theta_i^{\min}$ of the loss function by solving for the roots of the first derivative and filtering out the points where the second derivative is positive.
6. **Curvature and standard deviation calculation**: Calculate the curvature of the loss landscape at its minimum using its second derivative at the minimum. An estimation of the standard deviation δ_{θ_i} of the parameter θ_i is given by:

$$\delta_{\theta_i} = \left(\sqrt{\frac{\partial^2 L}{\partial \theta_i^2}} \right)^{-1} \Bigg|_{\theta_i = \theta_i^{\min}} \qquad \text{(C1)}$$

7. **Confidence interval calculation**: The loss function is typically asymmetric around its minimum, with the left side usually steeper than the right (see **Fig. 10**). To determine the right error bar, we use $\theta_i^{\min} + 1.96\delta_{\theta_i}$. For the left error bar, we find the point where $L(\theta_i^{\min} - \delta) = L(\theta_i^{\min} + 1.96\delta_{\theta_i})$ (see **Fig. 10**). Therefore, the balanced 95% CI for the parameter θ_i is given by:

$$CI_{\theta_i} = [\theta_i^{\min} - \delta, \theta_i^{\min} + 1.96\delta_{\theta_i}] \qquad \text{(C2)}$$

This methodology allows for a robust estimation of the confidence intervals, providing insights into the reliability and precision of the parameter estimates.

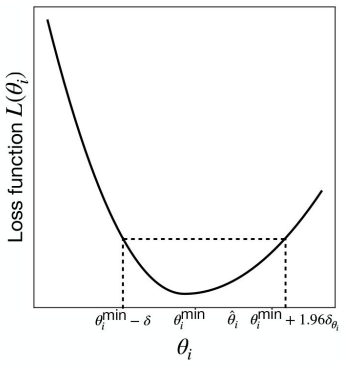


Fig. 10.

Error bars estimation for asymmetric loss function.

Appendix D: Demonstrating parameter degeneracy in the two-state promoter model

We will now demonstrate the existence of degeneracy in the two-state promoter architecture. Setting $k_{\text{off}}^{\text{R}} = 1$ in the analytical expressions for the mean m and the Fano factor f from Ref. [37], we have:

$$\begin{aligned} \langle m \rangle &= \frac{1}{k_{\text{on}}^{\text{R}} + 1} \cdot \frac{r}{\gamma}, \\ f &= 1 + \frac{k_{\text{on}}^{\text{R}}}{k_{\text{on}}^{\text{R}} + 1} \cdot \frac{r}{k_{\text{on}}^{\text{R}} + 1 + \gamma} \end{aligned} \tag{D1}$$

We want to solve Eqs. (D1) for r and γ. By isolating r in the expression for $\langle m \rangle$, we obtain:

$$r = \langle m \rangle \cdot \gamma \cdot (k_{\text{on}}^{\text{R}} + 1) \tag{D2}$$

Next, we substitute the expression for r from Eq. (D2) into the expression for the f in Eq. (D1):

$$\begin{aligned} f =& 1 + \frac{k_{\text{on}}^{\text{R}}}{k_{\text{on}}^{\text{R}} + 1} \cdot \frac{\langle m \rangle \cdot \gamma \cdot (k_{\text{on}}^{\text{R}} + 1)}{k_{\text{on}}^{\text{R}} + 1 + \gamma} \\ =& 1 + \frac{k_{\text{on}}^{\text{R}} \cdot \langle m \rangle \cdot \gamma}{k_{\text{on}}^{\text{R}} + 1 + \gamma} \\ \Rightarrow& (f-1) \cdot (k_{\text{on}}^{\text{R}} + 1 + \gamma) = k_{\text{on}}^{\text{R}} \cdot \langle m \rangle \cdot \gamma \end{aligned}$$

Expanding and isolating terms involving γ:

$$(f-1) \cdot k_{\text{on}}^{\text{R}} + (f-1) + (f-1) \cdot \gamma = k_{\text{on}}^{\text{R}} \cdot \langle m \rangle \cdot \gamma$$

Rearranging to solve for γ:

$$(f-1) + (f-1) \cdot k_{\text{on}}^{\text{R}} = \gamma \cdot (k_{\text{on}}^{\text{R}} \cdot \langle m \rangle - f + 1)$$

Thus, we obtain:

$$\gamma = \frac{(f-1) \cdot (k_{\text{on}}^{\text{R}} + 1)}{k_{\text{on}}^{\text{R}} \cdot \langle m \rangle - f + 1} \tag{D3}$$

We substitute back Eq. (D3) in Eq. (D1) to obtain r:

$$r = \frac{\langle m \rangle (f-1) \cdot (k_{\text{on}}^{\text{R}} + 1)^2}{k_{\text{on}}^{\text{R}} \cdot \langle m \rangle - f + 1} \tag{D4}$$

Eqs. (D3) and (D4) indicate that for any given k^{R}, there exists a corresponding pair of values for r and γ that satisfy the equations. Therefore, the solutions are not unique, demonstrating a high degree of degeneracy in the parameter space. This degeneracy arises because multiple combinations of $\{r, \gamma, k_{\text{on}}^{\text{R}}\}$ can produce the same observable $\langle m \rangle$ and f. The lack of unique solutions is a common issue in parameter estimation for complex systems, where different parameter sets can lead to similar system behaviors.

Resolving degeneracy with known γ

To resolve the degeneracy, we can fix the value of γ. Now, if the rate γ is known, we can solve Eqs. (D1) for the other parameters. Starting by rearranging the mean expression from Eqs. (D1):

$$k_{\text{on}}^{\text{R}} = \frac{r}{\gamma \cdot \langle m \rangle} - 1 \qquad \text{(D5)}$$

Substituting Eq. (D5) in the expression of f in Eq. (D1), we have

$$\begin{aligned} f &= 1 + \frac{r - \gamma\langle m\rangle}{r} \cdot \frac{r\gamma\langle m\rangle}{r + \gamma^2\langle m\rangle} \\ \Rightarrow &(f-1)(r + \gamma^2\langle m\rangle) = (r - \gamma\langle m\rangle)(\gamma\langle m\rangle) \\ \Rightarrow &r(f - 1 - \gamma\langle m\rangle) = -\gamma^2\langle m\rangle^2 - (f-1)\gamma^2\langle m\rangle \\ \Rightarrow &r = \frac{\langle m\rangle\gamma^2(\langle m\rangle + f - 1)}{\gamma\langle m\rangle - f + 1} \end{aligned} \qquad \text{(D6)}$$

Substituting Eq. (D6) into Eq. (D5), we obtain:

$$k_{\text{on}}^{\text{R}} = \frac{\gamma(\langle m\rangle + f - 1)}{\gamma\langle m\rangle - f + 1} - 1 \qquad \text{(D7)}$$

Eqs. (D6) and (D7) provide unique solutions for r and k_{on}^{R} given a known value of γ. By fixing γ, the degeneracy is resolved, and the remaining parameters can be accurately determined.

Appendix E: Estimating confidence intervals for ⟨ m ⟩ and σ

To assess the reliability of the statistics predicted through DGA-based optimization, we calculate error bars using the following procedure. For the non-degenerate situation, we use the learned parameter values $\hat{k}_{\text{on}}^{\text{R}}$ and $\hat{r}$, and perform forward DGA simulations with many different random seeds. This generates multiple samples of the mean mRNA level $\langle m\rangle$ and the standard deviation σ_m. These samples provide us with the variability due to the stochastic nature of the simulations. The 95% CIs are then determined using their standard deviations, denoted as $\delta_{\langle m\rangle}$ and δ_{σ_m}. For the mean mRNA level, the CI is calculated as:

$$\text{CI}_{\langle m\rangle} = [\langle\hat{m}\rangle - 1.96 \times \delta_{\langle m\rangle}, \langle\hat{m}\rangle + 1.96 \times \delta_{\langle m\rangle}] \qquad \text{(E1)}$$

For the standard deviation of mRNA levels, the CI is calculated as:

$$\text{CI}_{\sigma_m} = [\hat{\sigma}_m - 1.96 \times \delta_{\sigma_m}, \hat{\sigma}_m + 1.96 \times \delta_{\sigma_m}] \qquad \text{(E2)}$$

Appendix F: DGA-based optimization for experimental data and estimation of errors

1. Optimization Procedure

Given a set of measurements of the mean mRNA expression levels ($\langle m \rangle_i$) and the Fano factor (f^m) for promoters (*lac*UD5 and 5DL1), we construct a loss function as follows:

$$L = \sum_{i=1}^{N} (\langle \hat{m} \rangle_i - \langle m \rangle_i)^2 + \sum_{i=1}^{N} (\hat{\sigma}_i^m - \sqrt{f_i^m \langle m \rangle_i})^2, \quad \text{(F1)}$$

where *i* runs over data points (each with a different lac repressor concentration), and $\langle \hat{m} \rangle_i$ and $\hat{\sigma}_i^m$ are the mean and standard deviation obtained from a sample of DGA simulations. This loss function is chosen because, at its minimum, $\langle \hat{m} \rangle_i = \langle m \rangle_i$ and $\hat{\sigma}_i^m = \sqrt{f_i^m \langle m \rangle_i}$ for all *i*.

For the optimization, we set $k_{\text{off}}^{\text{R}} = 1$ and focus on estimating the parameters $\{r, \gamma, k_{\text{on}}^{\text{R}}\}$. During the gradientbased optimization, the transcription rate *r* and the mRNA degradation rate *γ* are assumed to be the same for all data points *i*, while allowing k_{on}^{R} to vary across data points *i*. This reflects the fact that k_{on}^{R} is a function of the lac repressor concentration, which is varied across data points.

Instead of k_{on}^{R}, we actually learn a transformed parameter $p_{\text{off}} = 1/(1 + k_{\text{on}}^{\text{R}})$, which is the probability for the promoter to be in the OFF state. This approach is based on our observation that the gradient of the loss function with respect to p_{off} is more numerically stable compared to the gradient with respect to k_{on}^{R}.

The parameters are initialized randomly as follows:

- *r* is initialized to a random number in [0, 100].
- *γ* is initialized to a random number between [0, 10].
- The p_{off} values, which depend on the index *i* of the data points, are initially set as linearly spaced points within the range [0.03, 0.97].

The hyperparameters used for the simulations are as follows:

- Number of simulations: 200
- Simulation time: 0.2
- Steepness of the sigmoid function: a^{-1} = 200.0
- Sharpness of the Gaussian function: b^{-1} = 20.0
- Steepness of the sigmoid in propensities: *c* = 20.0

The parameters are iteratively updated to minimize the loss function. During each iteration of the optimization, the following steps are performed:

1. **Forward simulation**: The DGA is used to simulate the system, generating predictions for the mean mRNA levels and their standard deviations.
2. **Loss calculation**: The loss function is computed based on the differences between the simulated and experimentally measured values of the mean mRNA levels and standard deviations (see Eq. (F1)).
3. **Gradient calculation**: The gradients of the loss function with respect to the parameters *r*, *γ*, and k_{on}^{R} are calculated using backpropagation.
4. **Parameter update**: The ADAM optimizer updates the parameters in the direction that reduces the loss function. ADAM adjusts the learning rates based on the history of gradients and their moments. The learning rate used is 0.1.

The values of the parameters and the loss value are saved after each iteration. The parameter values corresponding to the minimum loss after convergence are picked at the end.

2. Goodness of fit

To quantitatively assess the goodness of the fit, we define the mean percentage error (MPE) as follows:

$$\mathrm{MPE} = \frac{100\%}{N} \sum_{i=1}^{N} \frac{\left| f_i^m - \hat{f}_i^m \right|}{f_i^m}, \qquad \text{(F2)}$$

where $\hat{f}_i^m$ is the predicted Fano factor for the *i*-th data point. This metric provides a measure of the average discrepancy between the predicted and experimental Fano factors, expressed as a percentage of the experimental values.

3. Error bars

The error bars in the ratio $\hat{r}/\hat{\gamma}$ are obtained by applying error propagation to the standard deviations δ_r and δ_γ of the individual values $\hat{r}$ and $\hat{\gamma}$. The propagated error is given by:

$$\delta_{\frac{r}{\gamma}} = \frac{\hat{r}}{\hat{\gamma}} \sqrt{\left(\frac{\delta_r}{\hat{r}}\right)^2 + \left(\frac{\delta_\gamma}{\hat{\gamma}}\right)^2}, \qquad \text{(F3)}$$

where δ_r and δ_γ are the standard deviations of $\hat{r}$ and $\hat{\gamma}$, respectively. These standard deviations are obtained using the curvature of the loss function, as discussed earlier.

References

[1] Van Kampen N. G. 1992) **Stochastic processes in physics and chemistry** Elsevier

[2] Gardiner C. 2009) **Stochastic methods** Springer Berlin

[3] Rolski T., Schmidli H., Schmidt V., Teugels J. L. 2009) **Stochastic processes for insurance and finance** John Wiley & Sons

[4] Wong E., Hajek B. 2012) **Stochastic processes in engineering systems** Springer Science & Business Media

[5] McAdams H. H., Arkin A. 1997) **Proceedings of the National Academy of Sciences 94**:814

[6] Elowitz M. B., Levine A. J., Siggia E. D., Swain P. S. 2002) **Science 297**:1183

[7] Raj A., Van Oudenaarden A. 2008) **Cell 135**:216

[8] Sanchez A., Golding I. 2013) **Science 342**:1188

[9] Paulsson J. 2005) **Physics of life reviews 2**:157

[10] Wilkinson D. J. 2018) **Stochastic modelling for systems biology** Chapman and Hall/CRC

[11] Doob J. L. 1945) **Transactions of the American Mathematical Society 58**:455

[12] Gillespie D. T. 1977) **The journal of physical chemistry 81**:2340

[13] Pineda-Krch M. 2008) **Journal of Statistical Software 25**:1

[14] Parker M., Kamenev A. 2009) **Physical Review E 80**:021129

[15] Dobramysl U., Mobilia M., Pleimling M., Täuber U. C. 2018) **Journal of Physics A: Mathematical and The-oretical 51**:063001

[16] Benayoun M., Cowan J. D., van Drongelen W., Wallace E. 2010) **PLoS computational biology 6**:e1000846

[17] Rijal K., Müller N. I., Friauf E., Singh A., Prasad A., Das D. 2024) **Physical Review Letters 132**:228401

[18] Gillespie D. T. 1976) **Journal of computational physics 22**:403

[19] Gillespie D. T. 2007) **Annu. Rev. Phys. Chem 58**:35

[20] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., et al. 2019) **Advances in neural information processing systems 32**

[21] Bradbury J., Frostig R., Hawkins P., Johnson M. J., Leary C., Maclaurin D., Necula G., Paszke A., VanderPlas J., Wanderman-Milne S., Zhang Q. 2018) **JAX: composable transformations of**

Python+NumPy programs *GitHub*

[22] Bezanson J., Edelman A., Karpinski S., Shah V. B. 2017) **SIAM review 59**:65

[23] Liao H.-J., Liu J.-G., Wang L., Xiang T. 2019) **Physical Review X 9**:031041

[24] Schoenholz S., Cubuk E. D. 2020) **Advances in Neural Information Processing Systems 33**:11428

[25] Wei J., Chu X., Sun X.-Y., Xu K., Deng H.-X., Chen J., Wei Z., Lei M. 2019) **InfoMat 1**:338

[26] Degrave J., Hermans M., Dambre J., et al. 2019) **Frontiers in neurorobotics 13**:406386

[27] Arya G., Schauer M., Schäfer F., Rackauckas C. 2022) **Advances in Neural Information Processing Systems 35**:10435

[28] Arya G., Seyer R., Schäfer F., Lew A., Huot M., Mansinghka V. K., Rackauckas C., Chandra K., Schauer M. 2023) **arXiv**

[29] Bezgin D. A., Buhendwa A. B., Adams N. A. 2023) **Computer Physics Communications 282**:108527

[30] Anderson D. F. 2012) **SIAM Journal on Numerical Analysis 50**:2237

[31] Srivastava R., Anderson D. F., Rawlings J. B. 2013) **The Journal of chemical physics 138**

[32] Thanh V. H., Zunino R., Priami C. 2018) **Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474**:20180303

[33] Glynn P. W. 1990) **Communications of the ACM 33**:75

[34] McGill J. A., Ogunnaike B. A., Vlachos D. G. 2012) **Journal of Computational Physics 231**:7170

[35] Núñez M., Vlachos D. 2015) **The Journal of chemical physics 142**

[36] Sheppard P. W., Rathinam M., Khammash M. 2012) **The Journal of chemical physics 136**

[37] Jones D. L., Brewster R. C., Phillips R. 2014) **Science 346**:1533

[38] Lammers N. C., Flamholz A. I., Garcia H. G. 2023) **Proceedings of the National Academy of Sciences 120**:e2211203120

[39] Kingma D. P., Ba J. 2014) **arXiv preprint**

[40] Mehta P., Bukov M., Wang C.-H., Day A. G., Richardson C., Fisher C. K., Schwab D. J. 2019) **Physics reports 810**:1

[41] Phillips R., Kondev J., Theriot J., Garcia H. 2012) **Physical biology of the cell** Garland Science

[42] Sanchez A., Choubey S., Kondev J. 2013) **Annual review of biophysics 42**:469

[43] Phillips R., Belliveau N. M., Chure G., Garcia H. G., Razo-Mejia M., Scholes C. 2019) **Annual review of biophysics 48**:121

[44] Tian T., Xu S., Gao J., Burrage K. 2007) **Bioinformatics 23**:84

[45] Munsky B., Trinh B., Khammash M. 2009) **Molecular systems biology 5**:318

[46] Komorowski M., Finkenstädt B., Harper C. V., Rand D. A. 2009) **BMC bioinformatics 10**:1

[47] Villaverde A. F., Fröhlich F., Weindl D., Hasenauer J., Banga J. R. 2019) **Bioinformatics 35**:830

[48] Einav T., Duque J., Phillips R. 2018) **PLoS One 13**:e0204275

[49] Razo-Mejia M., Barnes S. L., Belliveau N. M., Chure G., Einav T., Lewis M., Phillips R. 2018) **Cell systems 6**:456

[50] Zoller B., Gregor T., Tkačik G. 2022) **Current opinion in systems biology 31**:100435

[51] Wong F., Gunawardena J. 2020) **Annual review of biophysics 49**:199

[52] Dixit S., Middelkoop T. C., Choubey S. 2024) **Biophysical Journal 123**:1015

[53] Estrada J., Wong F., DePace A., Gunawardena J. 2016) **Cell 166**:234

[54] Owen J. A., Horowitz J. M. 2023) **Nature Communications 14**:1280

[55] Qian H. 2007) **Annu. Rev. Phys. Chem 58**:113

[56] Mehta P., Schwab D. J. 2012) **Proceedings of the National Academy of Sciences 109**:17978

[57] Lan G., Sartori P., Neumann S., Sourjik V., Tu Y. 2012) **Nature physics 8**:422

[58] Lang A. H., Fisher C. K., Mora T., Mehta P. 2014) **Physical review letters 113**:148103

[59] Mehta P., Lang A. H., Schwab D. J. 2016) **Journal of Statistical Physics 162**:1153

[60] Strand N. E., Vroylandt H., Gingrich T. R. 2022) **The Journal of Chemical Physics 157**

[61] Nicholson S. B., Gingrich T. R. 2023) **Physical Review X 13**:041006

Author information

Krishna Rijal
Department of Physics, Boston University, Boston, United States
ORCID iD: 0000-0001-7236-7387

For correspondence: krishnarijal331@gmail.com

Pankaj Mehta
Department of Physics, Boston University, Boston, United States

Editors

Reviewing Editor
Anne-Florence Bitbol
Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland

Senior Editor
Aleksandra Walczak
École Normale Supérieure - PSL, Paris, France

Reviewer #1 (Public review):

Summary:

This work introduces the differentiable Gillespie algorithm, DGA, which is a differentiable variant of the celebrated (and exact) Gillespie algorithm commonly used to perform stochastic simulations across numerous fields, notably in the life sciences. The proposed DGA approximates the exact Gillespie algorithm using smooth functions yielding a suitable approximate differentiable stochastic system as a proxy for the underlying discrete stochastic system, where DGA stochastic reactions have continuous reaction index and the species abundances. To illustrate their methodology, the authors specifically consider in detail the case of a well-studied two-state promoter gene regulation system that they analyze using a machine learning approach, and by combining simulation data with analytical results. For the two-state promoter gene system, the DGA is benchmarked by accurately reproducing the results of the exact Gillespie algorithm. For this same simple system, the authors also show how the DGA can be used for estimating kinetic parameters of both simulated and real noisy experimental data. This lets them argue convincingly that the DGA can become a powerful computation tool for applications in quantitative and synthetic biology. In order to argue that the DGA can be employed to design circuits with ad-hoc input-output relations, these considerations are then extended to a more complex four-state promoter model of gene regulation. The main strength of the paper is its clarity and its pedagogical presentation of the simulation methods.

Strengths:

The main strength of the paper is its clarity and its pedagogical presentation of the simulation methods.

Weaknesses:

It would have been useful to have a brief discussion, based on a concrete example, of what can be achieved with the DGA and is totally beyond the reach of the Gillespie algorithm and the numerous existing stochastic simulation methods. A more comprehensive and quantitative analysis of the limitations of the DGA, e.g. for rare events, and how it might be used for stochastic spatial systems would have also been helpful. However, this is arguably beyond the scope of this study whose primary goal is to introduce the DGA and demonstrate that it can achieve tasks like parameter estimation and network design.

Comments on revisions:

The authors have made a sound effort to address many of the comments raised in the previous reports. This has helped improve the clarity of the discussion.

https://doi.org/10.7554/eLife.103877.2.sa3

Reviewer #2 (Public review):

Summary:

In this work, the authors present a differentiable version of the widely-used Gillespie Algorithm. The Gillespie Algorithm has been used for decades to simulate the behavior of stochastic biochemical reaction networks. But while the Gillespie Algorithm is a powerful tool for the forward simulation of biochemical systems given some set of known reaction parameters, it cannot be used for reverse process, i.e. inferring reaction parameters given a set of measured system characteristics. The Differentiable Gillespie Algorithm ("DGA") overcomes this limitation by approximating two discontinuous steps in the Gillespie Algorithm with continuous functions. This makes it possible to calculate of gradients for each step in the simulation process which, in turn, allows the reaction parameters to be optimized via powerful backpropagation techniques. In addition to describing the theoretical underpinnings of DGA, the authors demonstrate different potential use-cases for the algorithm in the context of simple models of stochastic gene expression.

Overall, the DGA represents an important conceptual step forward for the field and should lay the groundwork for exciting innovations in the analysis and design of stochastic reaction networks. At the same time, significantly more work is needed to establish when the approximations made by DGA are valid and to demonstrate the viability of the algorithm in the context of complicated reaction networks.

Strengths:

This work makes an important conceptual leap by introducing a version of the Gillespie Algorithm that is end-to-end differentiable. This idea alone has the potential to drive a number of exciting innovations in the analysis, inference, and design of biochemical reaction networks. Beyond the theoretical adjustments, the authors also implement their algorithm in a Python-based codebase that combines DGA powerful optimization libraries like PyTorch. This codebase has the potential to be of interest to a wide range of researchers, even if the true scope of the method's applicability remains to be fully determined.

The authors also demonstrate how DGA can be used in practice both to infer reaction parameters from real experimental data (Figure 7) and to design networks with user-specified input-output characteristics (Figure 8). These illustrations should provide a nice roadmap for researchers interested in applying DGA to their own projects/systems.

Finally, although it does not stem directly from DGA, the exploration of pairwise parameter dependencies in different network architectures provides an interesting window into the design constraints (or lack thereof) that shape the architecture of biochemical reaction networks.

Weaknesses:

While it is clear that the DGA represents an important conceptual advancement, the authors do not do enough in the present manuscript to (i) validate the robustness of DGA inference and (ii) demonstrate that DGA inference works in the kinds of complex biochemical networks where it would actually be of legitimate use.

It is to the authors' credit that they are open and explicit about the potential limitations of DGA due to breakdowns in its continuous approximations. However they do not provide the reader with nearly enough empirical (i.e. simulation-based) or theoretical context to assess when, why, and to what extent DGA will fail in different situations. In Figure 2, they compare DGA to GA (i.e. ground-truth) in the context of a simple two state model of a stochastic transcription. Even in this minimal system, we see that DGA deviates notably from ground-

truth both in the simulated mRNA distributions (Figure 2A) and in the ON/OFF state occupancy (Figure 2C). This begs the question of how DGA will scale to more complicated systems, or systems with non-steady state dynamics. Will the deviations become more severe? This is important because, in practice, there is really not much need for using DGA with a simple 2 state system-we have analytic solutions for this case. It is the more complex systems where DGA has the potential to move the needle.

A second concern is that the authors' present approach for parameter inference and error calculation does not seem to be reliable. For example, in Figure 5A, they show DGA inference results for the ON rate of a two-state system. We see substantial inference errors in this case, even though the inference problem should be non-degenerate in this case. One reason for this seems to be that the inference algorithm does not reliably find the global minimum of the loss function (Figure 2B). To turn DGA into a viable approach, it is paramount that the authors find some way to improve this behavior, perhaps by using multiple random initializations to better search the loss space.

Finally, the authors do a good job of illustrating how DGA might be used to infer biological parameters (Figure 7) and design reaction networks with desired input-output characteristics (Figure 8). However, analytic solutions exist for both of the systems they select for examples. This means that, in practice, there would be no need for DGA in these contexts, since one could directly optimize, e.g., the expressions for the mean and Fano Factor of the system in Figure 7A. I still believe that it is useful to have these examples, but it seems critical to add a use-case where DGA is the only option.

Comments on revisions:

I am concerned that the results in Figure 8D may not be correct, or that the authors may be mis-interpreting them. From my reading of the paper they cite (Lammers & Flamholz 2023), the equilibrium sharpness limit for the network they consider in Figure 8 should be 0.25. But both solutions shown in Figure 8D fall below this limit, which means that they have sharpness levels that could have been achieved with no energy expenditure. If this is the case, then it would imply that while both systems do dissipate energy, they are not doing so productively; meaning that the same results could be achieved while holding Phi=0.

I acknowledge that this could be due to a difference in how they measure sharpness, but wanted to raise it here in case it is, in fact, a genuine issue with the analysis.

There should be an easy fix for this: just set the sharper "desired response" curve in 8b to be such that it demands non-equilibrium sharpness levels (0.25)

https://doi.org/10.7554/eLife.103877.2.sa2

Reviewer #3 (Public review):

Summary:

This manuscript introduces a differentiable variant of the Gillespie algorithm (DGA) that allows gradient calculation using backpropagation. The most significant contribution of this work is the development of the DGA itself, a novel approach to making stochastic simulations differentiable. This is achieved by replacing discontinuous operations in the traditional Gillespie algorithm with smooth, differentiable approximations using sigmoid and Gaussian functions. This conceptual advance opens up new avenues for applying powerful gradient-based optimization techniques, prevalent in machine learning, to studying stochastic biological systems.

The method was tested on a simple two-state promoter model of gene expression. The authors found that the DGA accurately captured the moments of the steady-state distribution and other major qualitative features. However, it was less accurate at capturing information about the distribution's tails, potentially because rare events result from frequent low-probability reaction events where the approximations made by the DGA have a greater impact. The authors also used the DGA to design a four-state promoter model of gene regulation that exhibited a desired input-output relationship. The DGA could learn parameters that produced a sharper response curve, which was achieved by consuming more energy.

The authors conclude that the DGA is a powerful tool for analyzing and designing stochastic systems. The discussion lays several open questions in the field and constructively addresses shortcomings of the proposed method as well as potential ways forward.

Strengths:

The DGA allows gradient-based optimization techniques to estimate parameters and design networks with desired properties.

The DGA efficacy in estimating kinetic parameters from both synthetic and experimental data. This capability highlights the DGA's potential to extract meaningful biophysical parameters from noisy biological data.

The DGA's ability to design a four-state promoter architecture exhibits a desired input-output relationship. This success indicates the potential of the DGA as a valuable tool for synthetic biology, enabling researchers to engineer biological circuits with predefined behaviours.

Weaknesses:

The study primarily focuses on analysing the steady-state properties of stochastic systems.

Comments on revisions:

Thank you for addressing all the points raised. I am looking forward to seeing the next steps in DGAs development and performance!

https://doi.org/10.7554/eLife.103877.2.sa1

Author response:

The following is the authors' response to the current reviews.

> ***Response to Reviewer 2's comments:***
>
> *I am concerned that the results in Figure 8D may not be correct, or that the authors may be mis-interpreting them. From my reading of the paper they cite (Lammers & Flamholz 2023), the equilibrium sharpness limit for the network they consider in Figure 8 should be 0.25. But both solutions shown in Figure 8D fall below this limit, which means that they have sharpness levels that could have been achieved with no energy expenditure. If this is the case, then it would imply that while both systems do dissipate energy, they are not doing so productively; meaning that the same results could be achieved while holding Phi=0.*
>
> *I acknowledge that this could be due to a difference in how they measure sharpness, but wanted to raise it here in case it is, in fact, a genuine issue with the analysis.There should*

> *be an easy fix for this: just set the sharper "desired response" curve in 8b to be such that it demands non-equilibrium sharpness levels (0.25<S<0.5).*

Thank you for raising this point regarding the interpretation of our results in Figure 8D. We agree that if the equilibrium sharpness limit for this particular network is around 0.25 (as shown by Lammers & Flamholz 2023), then achieving a sharpness below this threshold could, in principle, be accomplished without any energy expenditure. However, in our current design approach, the loss function is solely designed to enforce agreement with a target mean mRNA level at different input concentrations; it does not explicitly constrain energy dissipation, noise, or other metrics. Consequently, the DGA has no built-in incentive to minimize or optimize energy consumption, which means the resulting solutions may dissipate energy without exceeding the equilibrium sharpness limit.

In other words, the same input–output relationship could theoretically be achieved with \Phi =0 if an explicit constraint or regularization term penalizing energy usage had been included. As noted, adding such a term (e.g., penalizing \Phi^2) is conceptually straightforward but falls outside the scope of this study. Our primary goal is to demonstrate the flexibility of the DGA in designing a desired response, rather than to delve into energy–sharpness trade-offs or other biological considerations

While we appreciate the suggestion to set a higher target sharpness that exceeds the equilibrium limit, we believe the current example effectively demonstrates the DGA's ability to design circuits with desired input-output relationships, which is the primary focus of this study. Researchers interested in optimizing energy efficiency, burst size, burst frequency, noise, response time, mutual information, or other system properties can easily extend our approach by incorporating additional terms into the loss function to target these specific objectives.

We hope this explanation addresses your concern and clarifies that the manuscript provides sufficient context for readers to interpret the results in Figure 8D correctly.

The following is the authors' response to the original reviews.

> ***Reviewer #1 (Public review):***

We thank Reviewer #1 for their thoughtful feedback and appreciation of the manuscript's clarity. Our primary goal is to introduce the DGA as a foundational tool for integrating stochastic simulations with gradient-based optimization. While we recognize the value of providing detailed comparisons with existing methods and a deeper analysis of the DGA's limitations (such as rare event handling), these topics are beyond the scope of this initial work. Our focus is on presenting the core concept and demonstrating its potential, leaving more extensive evaluations for future research.

> ***Reviewer #2 (Public review):***

We thank Reviewer #2 for their detailed and constructive feedback. We appreciate the recognition of the DGA as a significant conceptual advancement for stochastic biochemical network analysis and design.

> *Weaknesses:*
>
> *(1) Validation of DGA robustness in complex systems:*

Our primary goal is to introduce the DGA framework and demonstrate its feasibility. While validation on high-dimensional and non-steady-state systems is important, it is beyond the

scope of this initial work. Future studies may improve scalability by employing techniques such as dynamically adjusting the smoothness of the DGA's approximations during simulation or using surrogate models that remain differentiable but more accurately capture discrete behaviors in critical regions, thus preserving gradient computation while improving accuracy.

> *(2) Inference accuracy and optimization:*

We acknowledge that the non-convex loss landscape in the DGA can hinder parameter inference and convergence to global minima, as seen in Figure 5A. While techniques like multi-start optimization or second-order methods (e.g., L-BFGS) could improve performance, our focus here is on establishing the DGA framework. We plan to explore better optimization methods in future work to improve the accuracy of parameter inference in complex systems.

> *(3) Use of simple models for demonstration:*

We selected well-understood systems to clearly illustrate the capabilities of the DGA. These examples were intended to demonstrate how the DGA can be applied, rather than to solve problems better addressed by analytical methods. Applying DGA to more complex, analytically intractable systems is an exciting avenue for future work, but introducing the method was our main objective in this study.

> ***Reviewer #3 (Public review):***

We thank the reviewer for their detailed and insightful feedback. We appreciate the recognition of the DGA as a significant advancement for enabling gradient-based optimization in stochastic systems.

> *Weaknesses:*
>
> *(1) Application beyond steady-state analysis*

We acknowledge the limitation of focusing solely on steady-state properties. To extend the DGA for analyzing transient dynamics, time-dependent loss functions can be incorporated to capture system evolution over time. This could involve aligning simulated trajectories with experimental time-series data or using moment-matching across multiple time points.

> *(2) Numerical instability in gradient computation*

The reviewer correctly highlights that large sharpness parameters (a and b) in the sigmoid and Gaussian approximations can induce numerical instability due to vanishing or exploding gradients. To address this, adaptive tuning of a and b during optimization could balance smoothness and accuracy. Additionally, alternative smoothing functions (e.g., softmax-based reaction selection) and gradient regularization techniques (such as gradient clipping and trust-region methods) could improve stability and convergence.

> ***Reviewer #1 (recommendations):***

We thank the reviewer for their thoughtful and constructive feedback on our manuscript. Below, we address each of the comments and suggestions raised.

> *Main points:*
>
> *(1) It would have been useful to have a brief discussion, based on a concrete example, of what can be achieved with the DGA and is totally beyond the reach of the Gillespie*

> *algorithm and the numerous existing stochastic simulation methods.*

Thank you for your comment. We would like to clarify that the primary aim of this work is to introduce the DGA and demonstrate its feasibility for tasks such as parameter estimation and network design. Unlike traditional stochastic simulation methods, the DGA's differentiable nature enables gradient-based optimization, which is not possible with the classical Gillespie algorithm or its variants.

> *(2) As often with machine learning techniques, there is a sense of black box, with a lack of mathematical details of the proposed method: as opposite to the exact Gillespie algorithm, whose foundations lie on solid mathematical results (exponentially-distributed waiting times of continuous-time Markov processes), the DGA involves uncontrolled approximations, that are only briefly mentioned in the paper. For instance, it is currently simply noted that "the approximations introduced by the DGA may be pronounced in more complex settings such as the calculation of rare events", without specifying how limiting these errors are. It would be useful to include a clearer and more comprehensive discussion of the limitations of the DGA: When does it work accurately? What are the approximations/errors and can they be controlled? When is it worth paying the price for those approximations/errors, and when is it better to stick to the Gillespie algorithm? Is this notably the case for problems involving rare events? Clearly, these are difficult questions, and the answers are problem specific. However, it would be important to draw the readers' attention on the issues, especially if the DGA is presented as a potentially significant tool in computational and synthetic biology.*

We acknowledge the importance of discussing the limitations of the DGA in more detail. While we have noted that the approximations introduced by the DGA may impact its accuracy in certain scenarios, such as rare-event problems, a deeper exploration of these trade-offs is outside the scope of this work. Instead, we provide sufficient context in the manuscript to guide readers on when the DGA is appropriate.

> *(3) The DGA is here introduced and discussed in the context of non-spatial problems (simple gene regulatory networks). However, numerous problems in the life sciences and computational/synthetic biology, involve stochasticity and spatial degrees of freedom (e.g. for problems involving diffusion, migration, etc). It is notoriously challenging to use the Gillespie algorithm to efficiently simulate stochastic spatial systems, especially in the context of rare events (e.g., extinction or fixation problems). It would be useful to comment on whether, and possibly how, the DGA can be used to efficiently simulate stochastic spatial systems, and if it would be better suited than the Gillespie algorithm for this purpose.*

Thank you for pointing this out. Although our current work centers on non-spatial systems, we agree that many biological contexts incorporate both stochasticity and spatial degrees of freedom. Extending the DGA to efficiently simulate such systems would indeed require substantial modifications—for instance, coupling it with reaction-diffusion frameworks or spatial master equations. We believe this is an exciting direction for future research and mention it briefly in the discussion as a potential extension.

> *Minor suggestions:*
>
> *(1) After Eq.(10): it would be useful to explain and motivate the choice of the ratio JSD/H.*

Done.

> *(2) On page 6, just below the caption of Fig.4: it would be useful to clarify what is actually meant by "... convergence towards the steady-state distribution of the exact Gillespie simulation, which is obtained at a simulation time of 10^4".*

Done.

> *(3) At the end of Section B on page 7: please clarify what is meant here by "soft directions".*

Done.

> ***Reviewer #2 (recommendations):***

We thank the reviewer for their thoughtful comments and constructive feedback. Below, we address each of the comments/suggestions.

> *Main points:*
>
> *(1) Enumerate the conditions under which DGA assumptions hold (and when they do not). There is currently not enough information for the interested reader to know whether DGA would work for their system of interest. Without this information, it is difficult to assess what the true scope of DGA's impact will be. One simple idea would be to test DGA performance along two axes: (i) increasing number of model states and (ii) presence/absence of non-steady state dynamics. I acknowledge that these are very open-ended directions, but looking at even a single instance of each would greatly strengthen this work. Alternatively, if this is not feasible, then the authors should provide more discussion of the attendant difficulties in the main text.*

We agree that a detailed exploration of the conditions under which the DGA assumptions hold would be a valuable addition to the field. However, this paper primarily aims to introduce the DGA methodology and demonstrate its proof-of-concept applications. A comprehensive analysis along axes such as increasing model states or non-steady-state dynamics, while important, would require significant additional simulations and is beyond the scope of this work. In Appendix A, we have discussed the trade-off between accuracy and numerical stability. Additionally, we encourage future users to tune the hyperparameters a and b for their specific systems.

> *(2) Demonstrate DGA performance in a more complex biochemical system. Clearly the authors were aware that analytic solutions exist for the 2-state system in Figure 7, but it this is actually also the case (I think) for mean mRNA production rate of the non-equilibrium system in Figure 8. To really demonstrate that DGA is practically viable, I encourage the authors to seek out an interesting application that is not analytically tractable.*

We appreciate the suggestion to validate DGA on a more complex biochemical system. However, the goal of this study is not to provide an exhaustive demonstration of all possible applications but to introduce the DGA and validate it in systems where ground-truth comparisons are available. While the non-equilibrium system in Figure 8 might be analytically tractable, its complexity already provides a meaningful demonstration of DGA's ability to optimize parameters and design systems. Extending this work to analytically intractable systems is an exciting direction for future studies, and we hope this paper will inspire others to explore these applications.

> *(3) Take steps to improve the robustness of parameter optimization and error bar calculations. (3a) When the loss landscape is degenerate, shallow, or otherwise "difficult," a common solution is to perform multiple (e.g. 25-100) inference runs starting from different random positions in parameter space. Doing this, and then taking the parameter set that minimizes the loss should, in theory, lead to a more robust recovery of the optimal parameter set.*
>
> *(3b) It seems clear that the Hessian approximation is underestimating the true error in your inference results. One alternative is to use a "brute force" approach like bootstrap resampling to get a better estimate for the statistical dispersion in parameter estimates. But I recognize that this is only viable if the inference is relatively fast. Simply recovering the true minimum will, of course, also help.*

(3a) We acknowledge the challenge posed by degenerate or shallow loss landscapes during parameter optimization. While performing multiple inference runs from different initializations is a common strategy, this approach is computationally intensive. Instead, we rely on standard optimization techniques (e.g., ADAM) to find a robust local minimum.

(3b) Thank you for your comment. We agree that Hessian-based error bars can underestimate uncertainty, particularly in degenerate or poorly conditioned loss landscapes. While methods like bootstrap and Monte Carlo can provide more robust estimates, they can be computationally prohibitive for larger-scale simulations. A simpler reason for not using them is the high resource demand from repeated simulations, which quickly becomes infeasible for complex or high-dimensional models. We note these trade-offs between robust estimation and practicality as an important area for further exploration.

> *Moderate comments:*
>
> *(1) Figure 7: is it possible to also show the inferred kon values? Specifically, it would be of interest to see how kon varies with repressor concentration.*

Thank you for the suggestion. We have updated Figure 7 to include the inferred kon values, showing their variation with the mean mRNA copy number. However, we could not plot them against repressor concentration due to the lack of available data.

> *(2) Figure 8B & D: the authors claim that the sharper system dissipates more energy, but doesn't 8D show the opposite of this? More importantly, it does not look like either network drives sharpness levels that exceed the upper equilibrium limit cited in [36]. So it is not clear that it is appropriate to look at energy dissipation here. In fact, it is likely that equilibrium networks could produce the curves in 8B, and might be worth checking.*

Thank you for pointing this out. We realized that the plotted values in Figure 8D were incorrect, as we had mistakenly plotted noise instead of energy dissipation. The plot has now been corrected.

> *(3) Figure 8: I really like this idea of using DGA to "design" networks with desired input-output properties, but I wonder if you could explore more a biologically compelling use-case. Specifically, what about some kind of switch-like logic where, as the activator concentration increases, you have first 0 genes on, then 1 promoter on, then 2 promoters on. This would achieve interesting regulatory logic, and having DGA try to produce step functions would ensure that you force the networks to be maximally sharp (i.e. about double what you're currently achieving).*

Thank you for this intriguing suggestion. While the proposed switch-like logic use case is indeed compelling, implementing such a system would require significant work. This goes beyond the scope of the current study, which focuses on demonstrating the feasibility of DGA for network design with simple input-output properties.

> *Minor comments:*
>
> *(1) Figure 4B & C: the bar plots do not do a good job conveying the points made by the authors. Consider alternatives, such as scatter plots or box plots that could convey inference uncertainty.*

Done.

> *(2) Figure 4B: consider using a log y-axis.*

The y-axis in Figure 4B is already plotted on a log scale.

> *(3) Figure 4D is mentioned prior to 4C in the text. Consider reordering.*

Done.

> *(4) Figure 5B: it is difficult to assess from this plot whether or not the landscape is truly "flat," as the authors claim. Flat relative to what? Consider alternative ways to convey your point.*

Thank you for highlighting this ambiguity. By describing the loss landscape as “flat,” we intend to convey its relative insensitivity to parameter variations in certain regions, rather than implying a completely level surface. While we believe Figure 5B still provides a useful qualitative depiction of this behavior, we acknowledge that it does not quantitatively establish “flatness.” In future work, we plan to incorporate more rigorous measures—such as gradient magnitudes or Hessian eigenvalues—to more accurately characterize and communicate the geometry of the loss landscape.

> ***Reviewer #3 (recommendations):***

We sincerely thank the reviewer for their thoughtful feedback and constructive suggestions, which have helped us improve the clarity and rigor of our manuscript. Below, we address each of the comments.

> *(1) Precision is lacking in the introduction section. Do the authors mean the Direct SSA, sorted SSA, which is usually faster, and how about rejection sampling methods?*

Thank you for pointing this out. We have updated the introduction to explicitly mention the Direct SSA.

> *(2) When mentioning PyTorch and Jax, would be good to also talk about Julia, as they have fast stochastic simulators.*

We have now mentioned Julia alongside PyTorch and Jax.

> *(3) Mentioned references 22-27. Reference 26 is an odd choice; a better reference is from the same author the Automatic Differentiation of Programs with Discrete Randomness, G Arya, M Schauer, F Schäfer, C Rackauckas, Advances in Neural Information Processing Systems, NeurIPS 2022*

We have now cited the suggested reference.

> *(4) Page 1, Section: 'To circumnavigate these difficulties, the DGA modifies....' Have you thought about how you would deal with the bias that will be introduced by doing this?*

Thank you for your insightful comment. We acknowledge the potential for bias due to the differentiable approximations in the DGA; however, our analysis has not revealed any systematic bias compared to the exact Gillespie algorithm. Instead, we observe irregular deviations from the exact results as the smoothness of the approximations increases.

> *(5) Page 2, first sentence '... traditional Gillespie...' be more precise here - the direct algorithm.*

Thank you for your comment. We believe that the context of the paper, particularly the schematic in Figure 1, makes it clear that we are focusing on the Direct SSA.

> *(6) Page 2, second paragraph: ' In order to simulate such a system...' This doesn't fit here as this section is about tau-leaping. As this approach approximates discrete operations, it is unclear if it would work for large models, snap-shot data of larger scale and if it would be possible to extend it for time-lapse data*

Thank you for your comment. We respectfully disagree that this paragraph is misplaced. The purpose of this paragraph is to explain why the standard Gillespie algorithm does not use fixed time intervals for simulating stochastic processes. By highlighting the inefficiency of discretizing time into small intervals where reactions rarely occur, the paragraph provides necessary context for the Gillespie algorithm's event-driven approach, which avoids this inefficiency.

Regarding the applicability of the DGA to larger models, snapshot data, or time-lapse data, we acknowledge these are important directions and have noted them as potential extensions in the discussion section.

> *(7) Page 2 Section B: 'In order to make use of modern deep-learning techniques...' It doesn't appear from the paper that any modern deep learning is used.*

Thank you for your comment. Although the DGA does not utilize deep learning architectures such as neural networks, it employs automatic differentiation techniques provided by frameworks like PyTorch and Jax. These tools allow efficient gradient computations, making the DGA compatible with modern optimization workflows.

> *(8) Page 3, Fig 1(a). S matrix last row, B and C should swap places: B should be 1 and C is -1.*

Corrected the typo.

> *(9) Fig1 needs a more detailed caption.*

Expanded the caption slightly for clarity.

> *(10) Page 3 last paragraph: 'The hyperparameter b...' Consequences of this are relevant, for example can we now go below zero. Also, we lose more efficient algorithms here. It would be good to discuss this in more detail that this is an approx.. algorithm that is good for our case study, but for other to use it more tests are needed.*

Thank you for the comment. Appendix A discusses the trade-offs related to a and b, but we agree that more detailed analysis is needed. The hyperparameters are tailored to our case study and must be tuned for specific systems.

> *(11) Page 4, Section C, first paragraph, 'The goal of making...' This is snapshot data. Would the framework also translate to time-lapse data? Also, it would be better to make it clearer earlier which type of data are the target of this study.*

Thank you for your suggestion. While the current study focuses on snapshot data and steady-state properties, we believe the DGA could be extended to handle time-lapse data by incorporating multiple recorded time points into its inference objective. Specifically, one could modify the loss function to penalize discrepancies across observed transitions between these time points, effectively capturing dynamic trajectories. We consider this an exciting area for future development, but it lies beyond our present scope.

> *(12) Page 4 Section C, sentence '...experimentally measured moments'. Should later be mentioned as error, as moments are imperfect*

Thank you for your comment. We agree that experimentally measured moments are inherently noisy and may not perfectly represent the true system. However, within the context of the DGA, these moments serve as target quantities, and the discrepancy between simulated and measured moments is already accounted for in the loss function.

> *(13) Page 4 Section C, last sentence '...second-order...such as ADAM'. Another formulation would be better as second order can be confusing, especially in the context of parameter estimation*

We have revised the language to avoid confusion regarding “second-order” methods.

> *(14) Fig 4(a) a density plot would fit better here*

Fig. 4(a) has been updated to a scatter density plot as suggested.

> *(15) Fig 4(c) Would be interesting to see closer analysis of trade of between gradient and accuracy when changing a and b parameters*

Thank you for this suggestion. We acknowledge that an in-depth exploration of these trade-offs could provide deeper insights into the method’s performance. However, for now, we believe the current analysis suffices to highlight the utility of the DGA in the contexts examined.

> *(16) Page 6 Section III, first sentence: This fits more to intro. Further the reference list is severely lacking here, with no comparison to other methods for actually fitting stochastic models.*

Thank you for the suggestion. We have added a few references there.

> *(17) Page 6, Section A, sentence: '....experimental measured mean...' Why is it a good measure here (moment matching is not perfect), also do you have distribution data, would that not be better? How about accounting for measurement error?*

Thank you for the comment. While we do not have full distribution data, we acknowledge that incorporating experimental measurement error could enhance the framework. A

weighted loss function could model uncertainty explicitly, but this is beyond the scope of the current study.

> *(18) Page 7, section B, first paragraph: 'Motivated by this, we defined the...'Why using Fisher-Information when profile-likelihood have proven to be better, especially for systems with few parameters like this.*

Thank you for the suggestion. While profile-likelihood is indeed a powerful tool for parameter uncertainty analysis, we chose Fisher Information due to its computational efficiency and compatibility with the differentiable nature of the DGA framework.

> *(19) Page 7, section C, sentence '...set kR/off=1..'. In this case, we cannot infer this parameter.*

Thank you for the comment. You are correct that setting kR/off = 1 effectively normalizes the rates, making this parameter unidentifiable. In steady-state analyses, not all parameters can be independently inferred because observable quantities depend on relative—rather than absolute—rate values (as evident when setting the time derivative to zero in the master equation). To infer all parameters, one would need additional information, such as time-series data or moments at finite time.

> *(20) Page 7 Section 2. Estimating parameters .... Sentence: '....as can be seen, there is very good agreement..' How many times the true value falls within the CI (because corr 0.68 is not great).*

Thank you for your comment. While a correlation coefficient of 0.68 indicates moderate agreement, the primary goal was to demonstrate the feasibility of parameter estimation using the DGA rather than achieving perfect accuracy. The coverage of the CI was not explicitly calculated, as the focus was on the overall trends and relative agreement.

> *(21) Page 7 Section 2. Estimating parameters .... Sentence: 'Fig5(c) shows....' Is this when using exact simulator?*

Thank you for your question. Yes, the exact values in x-axis of Fig. 5(c) are obtained using the exact Gillespie simulation.

> *(22) Page 7 Section 3 Estimating parameters for the... Sentence: 'Fig6(a) shows...' Why Cis are not shown?*

Thank you for your comment. CIs are not shown in Fig. 6(a) because this particular case is degenerate, making the calculation and meaningful representation of CIs challenging.

> *(23) Page 10, Sentence: 'As can be seen in Fig 7(b)...' Can you show uncertainty in measured value? It would be good to see something of a comparison against an exact method, at least on simulated synthetic data*

Thank you for the comment. Fig. 7(a) already includes error bars for the experimental data, which account for measurement uncertainty. However, in Fig. 7(b), we do not include error bars for the experimental values due to limitations in the available data.

> *(24) Page 12, Section B Loss function '...n=600...' This is on a lower range. Have you tested with n=1000?*

Yes, we have tested with n=1000 and observed no significant difference in the results. This indicates that n=600 is sufficient for the purposes of this study.

> *(25) Fig 8(c) why there are no CI shown?*

Thank you for your comment. CIs were not included in Fig. 8(c) due to degeneracy, which makes meaningful confidence intervals difficult to compute.

> *(26) Page 12 Conclusion, sentence: '..gradients via backpropagation...' Actually, by making the function continuous, both forward and reverse mode might be used. And in this case, forward-mode would actually be the fastest by quite a margin*

Thank you for your insightful comment. You are correct that by making the function continuous, both forward-mode and reverse-mode automatic differentiation can be used. We have now mentioned this point in the discussion.

> *(27) Overall comment for the Conclusion section: It would be good to discuss how this framework compares to other model-fitting frameworks for models with stochastic dynamics. The authors mention dynamic data and more discussion on this would be very welcomed. Why use ADAM and not something established like BFGS for model fitting? It would be interesting to discuss how this can fit with other SSA algorithms (e.g. in practice sorting SSA is used when models get larger). Also, inference comparison against exact approaches would be very nice. As it is now, the authors truly only check the accuracy of the SSA on 1 model -it would be interesting to see for other models.*

Thank you for your detailed comments. While this study focuses on introducing the DGA and demonstrating its feasibility, we agree that comparisons with other model-fitting frameworks, testing on additional models, and integrating with other SSA variants like sorted SSA are important directions for future work. Similarly, extending the DGA to handle transient dynamics and exploring alternatives to ADAM, such as BFGS, are promising areas to investigate further.

https://doi.org/10.7554/eLife.103877.2.sa0